OPTIMAL LEARNING

OPTIMAL LEARNING

2nd Edition (Draft)

Warren B. Powell
Operations Research and Financial Engineering
Princeton University

Ilya O. Ryzhov
Robert H. Smith School of Business
University of Maryland

March 11, 2018

WILEY-INTERSCIENCE

A JOHN WILEY & SONS, INC., PUBLICATION

Published by John Wiley & Sons, Inc., Hoboken, New Jersey.
Published simultaneously in Canada.

Library of Congress Cataloging-in-Publication Data:

Optimal Learning / Warren B. Powell and Ilya O. Ryzhov
p. cm.—(Wiley series in survey methodology)
"Wiley-Interscience."
Includes bibliographical references and index.
ISBN 0-471-48348-6 (pbk.)
1. Surveys—Methodology. 2. Social
sciences—Research—Statistical methods. I. Groves, Robert M. II. Series.

HA31.2.S873 2007
001.4'33—dc22 2004044064
Printed in the United States of America.

10 9 8 7 6 5 4 3 2 1

To our families

CONTENTS

Preface xv

Acknowledgments xix

1 The challenges of learning 1

1.1 From optimization to learning 2
1.2 Areas of application 3
1.3 Major problem classes 10
1.4 Two applications 12
1.4.1 The newsvendor problem 13
1.4.2 Learning the best path 14
1.5 Learning from different communities 16
1.6 Information collection using decision trees 18
1.6.1 A basic decision tree 18
1.6.2 Decision tree for offline learning 19
1.6.3 Decision tree for online learning 21
1.6.4 Discussion 23
1.7 Website and downloadable software 24
1.8 Goals of this book 25
Problems 26

2 Adaptive learning 31

2.1 Belief models 32
2.2 The frequentist view 33
2.3 The Bayesian view 34
2.3.1 The updating equations for independent beliefs 35
2.3.2 Updating for correlated normal priors 37
2.3.3 Bayesian updating with an uninformative prior 40
2.4 Bayesian updating for sampled nonlinear models 41
2.5 The expected value of information 43
2.6 Updating for non-Gaussian priors 45
2.6.1 The gamma-exponential model 46
2.6.2 The gamma-Poisson model 48
2.6.3 The Pareto-uniform model 48
2.6.4 Models for learning probabilities* 49
2.6.5 Learning an unknown variance* 52
2.6.6 The normal as an approximation 54
2.7 Monte Carlo simulation 54
2.8 Why does it work?* 57
2.8.1 Derivation of $\tilde{\sigma}$ 57
2.8.2 Derivation of Bayesian updating equations for independent beliefs 58
2.9 Bibliographic notes 60
Problems 60

3 Offline learning 63

3.1 The model 64
3.2 Learning policies 66
3.2.1 Deterministic vs. sequential policies 66
3.2.2 Optimal learning policies 67
3.2.3 Heuristic policies 69
3.3 Simulating outcomes 73
3.3.1 Policy-independent representations 73
3.3.2 Policy-dependent representation 75
3.3.3 Discussion 75
3.4 Evaluating policies 76
3.5 Handling complex dynamics 77
3.6 Equivalence of using true means and sample estimates* 78
3.7 Tuning policies 79
3.8 Extensions of the ranking and selection problem 80
3.9 Bibliographic notes 83
Problems 83

4 Value of information policies 87

4.1 The value of information 88
4.2 The knowledge gradient for independent beliefs 90
4.2.1 Computation 91
4.2.2 Some properties of the knowledge gradient 93
4.3 The value of information and the S-curve effect 94
4.3.1 The S-curve 94
4.3.2 The KG(*) policy 96
4.3.3 A tunable lookahead policy 96
4.3.4 The problem of too many choices 97
4.4 The four distributions of learning 98
4.5 Knowledge gradient for correlated beliefs 98
4.5.1 Updating with correlated beliefs 99
4.5.2 Computing the knowledge gradient 101
4.5.3 The value of a KG policy with correlated beliefs 102
4.6 Anticipatory vs. experiential learning 104
4.7 The knowledge gradient for some non-Gaussian distributions 105
4.7.1 The beta-Bernoulli model 106
4.7.2 The gamma-exponential model 108
4.7.3 The gamma-Poisson model 111
4.7.4 The Pareto-uniform model 112
4.7.5 The normal distribution as an approximation 113
4.7.6 Discussion 114
4.8 Expected improvement 114
4.9 The problem of priors 115
4.10 Value of information with unknown variance* 117
4.11 Discussion 120
4.12 Why does it work?* 121
4.12.1 Derivation of the knowledge gradient formula 121
4.13 Bibliographic notes 125
Problems 126

5 Online learning 137

5.1 An excitation policy 140
5.2 Upper confidence bounding 141
5.3 Gittins indices 143
5.3.1 Gittins indices in the beta-Bernoulli model 144
5.3.2 Gittins indices in the normal-normal model 146
5.3.3 Approximating Gittins indices 149
5.4 The knowledge gradient for online learning 149
5.4.1 The basic idea 150
5.4.2 Tunable variations 152
5.4.3 Some experimental comparisons 153

5.4.4 Non-normal models 155
5.5 Variations of bandit problems 156
5.6 Evaluating policies 157
5.7 Bibliographic notes 160
Problems 160

6 Elements of a learning problem 165

6.1 The states of our system 166
6.2 Types of decisions 169
6.3 Exogenous information 170
6.4 Transition functions 171
6.5 Objective functions 172
6.5.1 Terminal vs. cumulative reward 172
6.5.2 Experimental costs 174
6.5.3 Objectives 175
6.6 Designing policies 181
6.6.1 Policy search 182
6.6.2 Lookahead policies 183
6.6.3 Classifying policies 183
6.7 The role of priors in policy evaluation 185
6.8 Policy search and the MOLTE testing environment 188
6.9 Discussion 191
6.10 Bibliographic notes 192
Problems 193

7 Linear belief models 195

7.1 Applications 196
7.1.1 Maximizing ad clicks 197
7.1.2 Dynamic pricing 198
7.1.3 Housing loans 198
7.1.4 Optimizing dose response 199
7.2 A brief review of linear regression 200
7.2.1 The normal equations 200
7.2.2 Recursive least squares 201
7.2.3 A Bayesian interpretation 202
7.2.4 Generating a prior 203
7.3 The knowledge gradient for a linear model 205
7.3.1 Calculations 205
7.3.2 Behavior 206
7.4 Application to drug discovery 208
7.5 Application to dynamic pricing 212

7.6 Bibliographic notes 216
Problems 216

8 Nonlinear belief models 219

8.1 An optimal bidding problem 220
8.1.1 Modeling customer demand 221
8.1.2 Some valuation models 222
8.1.3 The logit model 223
8.1.4 Bayesian modeling for dynamic pricing 225
8.1.5 An approximation for the logit model 230
8.1.6 Bidding strategies 232
8.1.7 Numerical illustrations 235
8.2 A sampled representation 238
8.2.1 The belief model 239
8.2.2 The experimental model 240
8.2.3 Sampling policies 240
8.2.4 Resampling 241
8.3 The knowledge gradient for sampled belief model 241
8.4 Locally quadratic approximations 243
8.5 Bibliographic notes 243
Problems 243

9 Large choice sets 245

9.1 Sampled experiments 246
9.2 Subset selection 247
9.2.1 Choosing a subset 249
9.2.2 Setting prior means and variances 249
9.2.3 Two strategies for setting prior covariances 250
9.2.4 Managing large sets 252
9.2.5 Using simulation to reduce the problem size 252
9.2.6 Computational issues 254
9.2.7 Experiments 255
9.3 Hierarchical learning for multiattribute choices 257
9.3.1 Laying the foundations 258
9.3.2 Combining multiple levels of aggregation 261
9.3.3 Hierarchical knowledge gradient 263
9.4 Why does it work? 267
9.4.1 Computing bias and variance 267
9.5 Bibliographic notes 269
Problems 269

10 Optimizing a scalar function 271

10.1 Deterministic experiments 271
10.1.1 Differentiable functions 272
10.1.2 Nondifferentiable functions, finite budget 272
10.1.3 Nondifferentiable functions, infinite budget 274
10.2 Noisy experiments 275
10.2.1 The model 276
10.2.2 Finding the posterior distribution 276
10.2.3 Choosing the experiment 279
10.2.4 Discussion 281
10.3 Bibliographic notes 281
Problems 282

11 Stopping problems 283

11.1 Sequential probability ratio test 284
11.2 The secretary problem 288
11.2.1 Setup 289
11.2.2 Solution 290
11.3 Bibliographic notes 293
Problems 294

12 Active learning in statistics 297

12.1 Deterministic policies 298
12.2 Sequential policies for classification 301
12.2.1 Uncertainty sampling 302
12.2.2 Query by committee 303
12.2.3 Expected error reduction 304
12.3 A variance minimizing policy 305
12.4 Mixtures of Gaussians 307
12.4.1 Estimating parameters 307
12.4.2 Active learning 309
12.5 Bibliographic notes 310
12.5.1 Optimal computing budget allocation 311

13 Learning in mathematical programming 313

13.1 Applications 315
13.1.1 Piloting a hot air balloon 315
13.1.2 Optimizing a portfolio 320
13.1.3 Network problems 321
13.1.4 Discussion 325
13.2 Learning on graphs 325

13.3 Alternative edge selection policies 328
13.4 Learning costs for linear programs* 329
13.5 Bibliographic notes 335

14 Optimizing over continuous alternatives* 337

14.1 The belief model 339
14.1.1 Updating equations 340
14.1.2 Parameter estimation 342
14.2 Sequential kriging optimization 343
14.3 Efficient global optimization 345
14.4 Experiments 346
14.5 Extension to higher dimensional problems 347
14.6 Bibliographic notes 348

15 Learning with a physical state 351

15.1 Introduction to dynamic programming 353
15.1.1 Approximate dynamic programming 354
15.1.2 The exploration vs. exploitation problem 356
15.1.3 Discussion 357
15.2 Some heuristic learning policies 358
15.3 The local bandit approximation 358
15.4 The knowledge gradient in dynamic programming 361
15.4.1 Generalized learning using basis functions 361
15.4.2 The knowledge gradient 364
15.4.3 Experiments 366
15.5 An expected improvement policy 368
15.6 Bibliographic notes 369

PREFACE

This book emerged from a stream of research conducted in CASTLE Laboratory at Princeton University in 2006-2011. Initially, the work was motivated by the "exploration vs. exploitation" problem that arises in the design of algorithms for approximate dynamic programming, where it may be necessary to visit a state to learn the value of being in the state. However, we quickly became aware that this basic question had many applications outside of dynamic programming.

The results of this research were made possible by the efforts and contributions of numerous colleagues. The work was conducted under the guidance and supervision of Warren Powell, founder and director of CASTLE Lab. Key contributors include Peter Frazier, Ilya Ryzhov, Warren Scott, and Emre Barut, all graduate students at the time; Martijn Mes, a post-doctoral associate from University Twente in the Netherlands; Diana Negoescu, Gerald van den Berg and Will Manning, then undergraduate students. The earliest work by Peter Frazier recognized the power of a one-step look-ahead policy, which he named the knowledge gradient, in offline (ranking and selection) problems. The true potential of this idea, however, was realized later with two developments. The first, by Peter Frazier, adapted the knowledge gradient concept to an offline problem with correlated beliefs about different alternatives. This result made it possible to learn about thousands of discrete alternatives with very small measurement budgets.

The second development, by Ilya Ryzhov, made a connection between the knowledge gradient for offline problems and the knowledge gradient for online problems. This relationship links two distinct communities: ranking and selection in statistics and simulation, and the multi-armed bandit problem in applied probability and computer science. This link provides a tractable approach to multi-armed bandit problems with correlated beliefs, a relatively new extension of the well-known bandit model.

This foundation led to a number of additional results. Peter Frazier and Diana Negoescu created a version of the algorithm for problems where the belief structure is given by a linear model, which made it possible to tackle a problem in drug discovery, sorting through 87,000 possible drug combinations with just a few hundred experiments. Martijn Mes adapted the knowledge gradient approach to create an algorithm for a non-parametric model where the value of an alternative was represented through a hierarchical aggregation model. Emre Barut adapted this result to derive the knowledge gradient for a non-parametric belief model using kernel regression. Warren Scott derived a very difficult but powerful algorithm for when the choice of what to measure is a multidimensional vector of continuous parameters, which was applied to calibrate an industrial simulator for airline operations. Ilya Ryzhov then used the knowledge gradient idea to connect the notion of optimal learning with classical mathematical programming, making it possible to incorporate learning issues into fundamental optimization models such as linear programs. As of this writing, we are developing (with Gerald van den Berg) a method to handle the exploration issue in approximate dynamic programming – the very problem that we originally set out to solve.

The work inspired an undergraduate course (junior- and senior-level) at Princeton University called "Optimal Learning." Over several years, it has repeatedly attracted talented and enthusiastic students who have produced a creative collection of projects. This book evolved out of lecture notes written the first year that the course was offered. Indeed, the course covers roughly the first seven chapters, with other topics selected from the second half as time permits. The book is designed to be accessible to an advanced undergraduate audience, and presents an overview of the extensive body of research we compiled around the idea of the knowledge gradient. However, we also kept another goal in mind: to recognize the important contributions that had been made by a number of different communities such as economics, computer science, applied probability, simulation optimization, stochastic search, and ranking and selection.

The languages of these different communities have posed a particular challenge. For example, we use the term "online learning" to refer to learning where we have to live with the rewards we receive while also learning to improve decisions in the future, while some use this same term to refer to any sequential learning policy. Different communities are guided by specific classes of applications with characteristics that guide the choice of algorithms. The application setting is rarely transparent in the mathematics, complicating the evaluation of competing methods which have been designed with specific issues in mind. In the multi-armed bandit literature, an alternative x is always referred to as a "bandit," even if x is continuous and vector-valued. Even within this literature, the preferred techniques for solving these problems are quite different in applied probability (index policies) and computer science (upper confidence bounding).

Additional material for the book is available at the website:

`http://optimallearning.princeton.edu/`

Further reading, software, sample projects and additional thoughts about the field will be made available here.

Audience

The book is aimed primarily at an advanced undergraduate audience with a course in statistics and a full course in probability. The core of each chapter focuses on a specific learning

problem, and presents practical, implementable algorithms. Downloadable software is provided for several of the most important algorithms, and a fairly elaborate implementation, the Optimal Learning Calculator, is available as a spreadsheet interface calling a sophisticated Java library.

The later chapters cover material that is more advanced, including learning on graphs and linear programs, and learning where the alternatives are continuous (and possibly vector-valued). We have provided chapters designed to bridge with communities such as simulation optimization and machine learning. This material is designed to help Ph.D. students and researchers to understand the many communities that have contributed to the general area of optimal learning.

While every effort has been made to make this material as accessible as possible, the theory supporting this field can be quite subtle. Material that is more suitable to a graduate level audience is indicated with an * in the section title.

Organization of the book

The book is roughly organized into three parts:

Part I: Fundamentals

- Chapter 1 - The challenges of learning
- Chapter 2 - Adaptive learning
- Chapter 3 - The economics of information
- Chapter 4 - Ranking and selection
- Chapter 5 - The knowledge gradient
- Chapter 6 - Bandit problems
- Chapter 7 - Elements of a learning problem

Part II: Extensions and Applications

- Chapter 8 - Linear belief models
- Chapter 9 - Subset selection models
- Chapter 10 - Optimizing a scalar function
- Chapter 11 - Optimal bidding
- Chapter 12 - Stopping problems

Part III: Advanced topics

- Chapter 13 - Active learning in statistics
- Chapter 14 - Simulation optimization
- Chapter 15 - Learning in mathematical programming
- Chapter 16 - Optimizing over continuous measurements
- Chapter 17 - Learning with a physical state

The book is used as a textbook for an undergraduate course at Princeton. In this setting, Part I covers the foundations of the course. This material is supplemented with traditional

weekly problem sets and a midterm exam (two hourly exams would also work well here). Each of these chapters have a relatively large number of exercises to help students develop their understanding of the material.

After this foundational material is covered, students have to come up with a project which involves the efficient collection of information. Students are encouraged to work in teams of two, and the project progresses in three stages: initial problem definition, design of the belief model and learning algorithms, and then submission of the final report which involves the testing of different policies in the context of their application. In the initial problem definition, it is important for students to clearly identify the information that is being collected, the implementation decision (which may be different, but not always), and the metric used to evaluate the quality of the implementation decision.

While the students work on their projects, the course continues to work through most of the topics in Part II. The material on linear belief models is particularly useful in many of the student projects, as is the subset selection chapter. Sometimes it is useful to prioritize the material being presented based on the topics that the students have chosen. The chapters in part II have a small number of exercises, many of which require the use of downloadable Matlab software to help with the implementation of these more difficult algorithms.

Part III of the book is advanced material, and is intended primarily for researchers and professionals interested in using the book as a reference volume. These chapters are not accompanied by exercises, since the material here is more difficult and would require the use of fairly sophisticated software packages.

WARREN B. POWELL

ILYA O. RYZHOV

Princeton University, University of Maryland
October, 2011

ACKNOWLEDGMENTS

This book reflects the contributions of a number of students at Princeton University. Special thanks go to Peter Frazier, who developed the initial framework that shaped our research in this area. Warren Scott, Diana Negoescu, Martijn Mes and Emre Barut all made important contributions, and their enthusiastic participation in all stages of this research is gratefully acknowledged.

The Optimal Learning Calculator was co-written by Gerald van den Berg and Will Manning, working as undergraduate interns. The spreadsheet-based interface has provided valuable insights into the behavior of different learning policies.

We are very grateful to the students of ORF 418, Optimal Learning, who put up with the formative development of these ideas, and who provided an extensive list of creative projects to highlight potential applications of information collection.

The research was supported primarily by the Air Force Office of Scientific Research from the discrete mathematics program headed by Don Hearn, with additional support from the Department of Homeland Security through the CICCADA Center at Rutgers under the leadership of Fred Roberts, and the National Science Foundation. Special thanks also goes to SAP and the enthusiastic support of Bill McDermott and Paul Hofmann, and the many corporate partners of CASTLE Laboratory and PENSA who have provided the challenging problems which motivated this research.

W. B. P. and I. O. R.

CHAPTER 1

THE CHALLENGES OF LEARNING

We are surrounded by situations where we need to make a decision or solve a problem, but where we do not know some or all of the relevant information for the problem perfectly. Will the path recommended by my navigation system get me to my appointment on time? Am I charging the right price for my product, and do I have the best set of features? Will a new material make batteries last longer? Will a drug cocktail help reduce a cancer tumor? If I turn my retirement fund over to this investment manager, will I be able to outperform the market?

Sometimes the decisions have a simple structure (what is the best drug to treat my diabetes), while others require complex planning (how do I deploy a team to assess the outbreak of a disease). There are settings where we have to learn while we are doing (the sales of a book at a particular price), while in other cases we may have a budget to collect information before making a final decision that is implemented in the field. In fact, learning problems come in a range of major problem classes that reflect the nature of choices being made (e.g. which drug), what we learn (success/failure, or estimates of varying accuracy), the size of our learning budget, and whether we are learning online (in the field) or offline (in the laboratory).

There are some decision problems that are hard even if we have access to perfectly accurate information about our environment: planning routes for aircraft and pilots, optimizing the movements of vehicles to pick up and deliver goods, or scheduling machines to finish a set of jobs on time. This is known as deterministic optimization. Then there are other situations where we have to make decisions under uncertainty, but where we assume we know the probability distributions of the uncertain quantities: how do I allocate investments

Alternative	Value
1	759
2	722
3	698
4	653
5	616

(a) The best of five known alternatives

Alternative	Mean	Std. dev.
1	759	120
2	722	142
3	698	133
4	653	90
5	616	102

(b) The best of five uncertain alternatives

Table 1.1 (a) A problem involving five known alternatives, and (b) a problem where the value of each alternative is normally distributed with known mean and standard deviation.

to minimize risk while maintaining a satisfactory return, or how do I optimize the storage of energy given uncertainties about demands from consumers? This is known as stochastic optimization.

In this book, we introduce problems where we have only a probabilistic understanding about a relationship or function, but where we have the opportunity to collect new information to improve our understanding of what they are. We are primarily interested in problems where there is a cost to learning about our unknown function, but this cost can vary widely. An oil company may need to spend $100 million to learn about the geology in an underwater formation, while an internet company worries about showing a user a set of movies to learn more about the choices the user may make.

Learning problems seem, on the surface, to be relatively simple. In most instances, we are choosing among a finite set of alternatives (prices, drugs, paths, materials) to find the one that works the best. Ultimately, our goal is to design rules, that we will call policies, for making these decisions. Again, in most (but not all) cases, these policies will be relatively simple to implement. Modeling the problem of designing and testing these policies, however, can be fairly difficult. Furthermore, there are problems, such as those that arise in health, or drilling oil wells, where the decisions have to be made quite carefully. The challenge is that we have to model knowledge, and how the flow of information affects our belief state, which can be quite subtle.

1.1 FROM OPTIMIZATION TO LEARNING

We are going to use the simple setting of finding the best alternative to illustrate the difference between deterministic optimization, stochastic optimization, and learning. Figure 1.1 depicts two optimization problem. The problem in Figure 1.1(a) shows five choices, each of which has a known value. The best choice is obviously the first one, with a value of 759. Of course, deterministic optimization problems can be quite hard, but this happens to be a trivial one.

A harder class of optimization problems arise when there is uncertainty in the parameters. Figure 1.1(b) depicts a problem with five choices where the reward we receive from a choice is normally distributed with known mean and standard deviation. Assume that we have to make a choice before the reward is received, and we want to make a choice that gives us the highest expected return. Again, we would select the first alternative, because it has the highest expected value.

Alternative	Initial mean and std. dev. Mean	Std dev	First obs.	Updated mean and std. dev. Mean	Std dev	Second obs.
1	759	102	702	712	92	
2	722	133		722	133	734
3	698	78		698	78	
4	653	90		653	90	
5	616	102		616	102	

Table 1.2 Learning where we update our beliefs based on observations, which changes our distribution of belief for future experiments.

The problems illustrated in Table 1.1 use either known values, or known distributions. This problem is fairly trivial (picking the best out of a list of five), but there are many problems in stochastic optimization that are quite hard. In all of these problems, there are uncertain quantities but we assume that we know the probability distribution describing the likelihood of different outcomes. Since the distributions are assumed known, when we observe an outcome we view it simply as a realization from a known probability distribution. We do not use the observation to update our belief about the probability distribution.

Now consider what happens when you are not only uncertain about the reward, you are uncertain about the probability distribution for the reward. The situation is illustrated in Table 1.2, where after choosing to measure the first alternative, we observe an outcome of 702, and then use this outcome to update our belief about the first alternative. Before we run our experiment, we thought the reward was normally distributed with mean 759 and standard deviation 102. After the experiment, we now believe the mean is 712 with standard deviation of 92. As a result, alternative 2 now seems to be the best.

Since we are willing to change our belief about an alternative, is it necessarily the case that we should try to evaluate what appears to be the best alternative? Later in this volume, we are going to refer to this as an *exploitation policy*. This means that we exploit our current belief state and choose the alternative that appears to be best. But it might be the case that if we observe an alternative that does not appear to be the best to use right now, we may collect information that allows us to make better decisions in the future. The central idea of optimal learning is to incorporate the value of information in the future to make better decisions now.

1.2 AREAS OF APPLICATION

The diversity of problems where we have to address information acquisition and learning is tremendous. Below, we try to provide a hint of the diversity.

Transportation

- Finding the best path during construction - Imagine that a road is closed due to major bridge repair, forcing people to find new routes. Travelers have to find not just the best path, but the best time to leave in the morning for work.

Figure 1.1 The operations center for NetJets, which manages over 500 aircraft. NetJets has to test different policies to strike the right balance of costs and service.

- Revenue management - Providers of transportation need to set a price that maximizes revenue (or profit), but since demand functions are unknown, it is often necessary to do a certain amount of trial and error.
- Evaluating airline passengers or cargo for dangerous items - Examining people or cargo to evaluate risk can be time consuming. There are different policies that can be used to determine who/what should be subjected to varying degrees of examination. Finding the best policy requires testing them in field settings.
- Finding the best heuristic to solve a difficult integer program for routing and scheduling - Vehicle routing and scheduling problems are typically solved with heuristics that go under names such as local search, tabu search, and large neighborhood search, all of which have tunable parameters that affect their behavior. We need to search for the best values of these parameters.
- Finding the best business rules - A transportation company needs to determine the best terms for serving customers, rules for guiding drivers, the best mix of equipment and the right drivers to hire (see figure 1.1). They may use a computer simulator to evaluate these options, requiring time consuming simulations to be run to evaluate different strategies.

Energy and the environment

- Finding the best energy-saving technologies for a building - Insulation, tinted windows, motion sensors and automated thermostats interact in a way that is unique to each building. It is necessary to test different combinations to determine the technologies that work the best.
- Finding the best material for a solar panel - It is necessary to test large numbers of molecular compounds to find new materials for converting sunlight to electricity.

Figure 1.2 Wind turbines are one form of alternative energy resources (from http://www.nrel.gov/data/pix/searchpix.cgi).

Testing and evaluating materials is time consuming and very expensive, and there are large numbers of molecular combinations that can be tested.

- Optimizing the best policy for storing energy in a battery - A policy is defined by one or more parameters that determine how much energy is stored and in what type of storage device. One example might be, "charge the battery when the spot price of energy drops below x_1, discharge when it goes above x_2." We can collect information in the field or a computer simulation that evaluates the performance of a policy over a period of time.

- Learning how lake pollution due to fertilizer run-off responds to farm policies - We can introduce new policies that encourage or discourage the use of fertilizer, but we do not fully understand the relationship between these policies and lake pollution, and these policies impose different costs on the farmers. We need to test different policies to learn their impact, but each test requires a year to run and there is some uncertainty in evaluating the results.

- On a larger scale, we need to identify the best policies for controlling CO2 emissions, striking a balance between the cost of these policies (tax incentives on renewables, a carbon tax, research and development costs in new technologies) and the impact on global warming, but we do not know the exact relationship between atmospheric CO2 and global temperatures.

Homeland security

- You would like to minimize the time to respond to an emergency over a congested urban network. You can run experiments to improve your understanding of the time

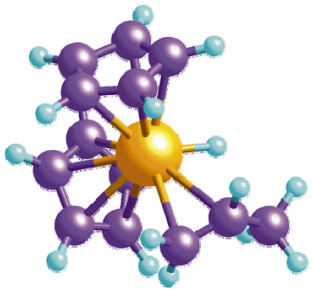


Figure 1.3 Drug discovery requires testing large numbers of molecules.

to traverse each region of the traffic network, but collecting these observations takes time. How should you structure your observations of links in the network to achieve the best time when you need to find the shortest path?

- You need to manage a group of inspectors to intercept potentially dangerous cargo being smuggled through ports and across borders. Since you do not know the frequency with which smugglers might try to use a port of entry, it is important to allocate inspectors not just to maximize the likelihood of an interception given current beliefs, but to also collect information so that we can improve our understanding of the truth. For example, we may believe that a particular entry point might have a low probability of being used, but we may be wrong.
- Radiation is detected in downtown Manhattan. Inspectors have to be managed around the city to find the source as quickly as possible. Where should we send them to maximize the likelihood of finding the source?

Science and engineering

- The National Ignition Facility uses large crystals to focus lasers into a very small region to perform nuclear research. The crystals become damaged over time and have to be repaired or replaced, but the process of examining each crystal is time consuming and reduces the productivity of the facility. NIF has to decide when to examine a crystal to determine its status.
- A company is trying to design an aerosol device whose performance is determined by a number of engineering parameters: the diameter of the tube that pulls liquid from a reservoir, the pressure, the angle of a plate used to direct the spray, the size of the portal used to project the spray and the angle of the departure portal. These have to be varied simultaneously to find the best design.

Health and medicine

- Drug discovery - Curing a disease often involves first finding a small family of base molecules, and then testing a large number of variations of a base molecule. Each test of a molecular variation can take a day, consumes costly materials, and the performance can be uncertain.

- Drug dosage - Each person responds to medication in a different way. It is often necessary to test different dosages of a medication to find the level that produces the best mix of effectiveness against a condition with minimum side effects.
- How should a doctor test different medications to treat diabetes, given that he will not know in advance how a particular patient might respond to each possible course of treatment?
- What is the best way to test a population for an emerging disease so that we can plan a response strategy?

Sports

- How do you find the best set of five basketball players to use as your starting lineup? Basketball players require complementary skills in defense, passing and shooting, and it is necessary to try different combinations of players to see which group works the best.
- What is the best combination of rowers for a four person rowing shell? Rowers require a certain synergy to work well together, making it necessary to try different combinations of rowers to see who turns in the best time.
- Who are the best hitters that you should choose for your baseball team? It is necessary to see how a player hits in game situations, and of course these are very noisy observations.
- What plays work the best for your football team? Specific plays draw on different combinations of talents, and a coach has to find out what works best for his team.

Business

- What are the best labor rules or terms in a customer contract to maximize profits? These can be tested in a computer simulation program, but it may require several hours (in some cases, several days) to run. How do we sequence our experiments to find the best rules as quickly as possible?
- What is the best price to charge for a product being sold over the Internet? It is necessary to use a certain amount of trial and error to find the price that maximizes revenue.
- We would like to find the best supplier for a component part. We know the price of the component, but we do not know about the reliability of the service or the quality of the product. We can collect information on service and product quality by placing small orders.
- We need to identify the best set of features to include in a new laptop we are manufacturing. We can estimate consumer response by running market tests, but these are time consuming and delay the product launch.
- A company needs to identify the best person to lead a division that is selling a new product. The company does not have time to interview all the candidates. How should a company identify a subset of potential candidates?

- Advertising for a new release of a movie - We can choose between TV ads, billboards, trailers on movies already showing, the Internet and promotions through restaurant chains. What works best? Does it help to do TV ads if you are also doing Internet advertising? How do different outlets interact? You have to try different combinations, evaluate their performance and use what you learn to guide future advertising strategies.

- Conference call or airline trip? Business people have to decide when to try to land a sale using teleconferencing, or when a personal visit is necessary. For companies that depend on numerous contacts, it is possible to experiment with different methods of landing a sale, but these experiments are potentially expensive, involving the time and expense of a personal trip, or the risk of not landing a sale.

E-commerce

- Which ads will produce the best consumer response when posted on a website? You need to test different ads, and then identify the ads that are the most promising based on the attributes of each ad.

- Netflix can display a small number of movies to you when you log into your account. The challenge is identifying the movies that are likely to be most interesting to a particular user. As new users sign up, Netflix has to learn as quickly as possible which types of movies are most likely to attract the attention of an individual user.

- You need to choose keywords to bid on to get Google to display your ad. What bid should you make for a particular keyword? You measure your performance by the number of clicks that you receive.

- YouTube has to decide which videos to feature on its website to maximize the number of times a video is viewed. The decision is the choice of video, and the information (and reward) is the number of times people click on the video.

- Amazon uses your past history of book purchases to make suggestions for potential new purchases. Which products should be suggested? How can Amazon use your response to past suggestions to guide new suggestions?

The service sector

- A university has to make specific offers of admission, after which it then observes which types of students actually matriculate. The university has to actually make an offer of admission to learn whether a student is willing to accept the offer. This information can be used to guide future offers in subsequent years. There is a hard constraint on total admissions.

- A political candidate has to decide in which states to invest his remaining time for campaigning. He decides which states would benefit the most through telephone polls, but has to allocate a fixed budget for polling. How should he allocate his polling budget?

- The Federal government would like to understand the risks associated with issuing small business loans based on the attributes of an applicant. A particular applicant

Figure 1.4 The air force has to design new technologies, and determine the best policies for operating them.

might not look attractive, but it is possible that the government's estimate of risk is inflated. The only way to learn more is to try granting some higher risk loans.

- The Internal Revenue Service has to decide which companies to subject to a tax audit. Should it be smaller companies or larger ones? Are some industries more aggressive than others (for example, due to the presence of lucrative tax write-offs)? The government's estimates of the likelihood of tax cheating may be incorrect, and the only way to improve its estimates is to conduct audits.

The military

- The military has to collect information on risks faced in a region using UAVs (unmanned aerial vehicles). The UAV collects information about a section of road, and then command determines how to deploy troops and equipment. How should the UAVs be deployed to produce the best deployment strategy?

- A fighter has to decide at what range to launch a missile. After firing a missile, we learn whether the missile hit its target or not, which can be related to factors such as range, weather, altitude and angle-of-attack. With each firing, the fighter learns more about the probability of success.

- The Air Force has to deploy tankers for mid-air refueling. There are different policies for handling the tankers, which include options such as shuttling tankers back and forth between locations, using one tanker to refuel another tanker, and trying different locations for tankers. A deployment policy can be evaluated by measuring how much time fighters spend waiting for refueling, and the number of times a fighter has to abort a mission from lack of fuel.

- The military has to decide how to equip a soldier. There is always a tradeoff between cost and the weight of the equipment, versus the likelihood that the soldier will survive. The military can experiment with different combinations of equipment to assess its effectiveness in terms of keeping a soldier alive.

Tuning models and algorithms

- There is a large community that models physical problems such as manufacturing systems using Monte Carlo simulation. For example, we may wish to simulate the manufacture of integrated circuits which have to progress through a series of stations. The progression from one station to another may be limited by the size of buffers which hold circuit boards waiting for a particular machine. We wish to determine the best size of these buffers, but we have to do this by sequential simulations which are time consuming and noisy.

- There are many problems in discrete optimization where we have to route people and equipment, or scheduling jobs to be served by a machine. These are exceptionally hard optimization problems that are typically solved using heuristic algorithms such as tabu search or genetic algorithms. These algorithms are controlled by a series of parameters which have to be tuned for specific problem classes. One run of an algorithm on a large problem can require several minutes to several hours (or more), and we have to find the best setting for perhaps five or ten parameters.

- Engineering models often have to be calibrated to replicate a physical process such as weather or the spread of a chemical through groundwater. These models can be especially expensive to run, often requiring the use of fast supercomputers to simulate the process in continuous space or time. At the same time, it is necessary to calibrate these models to produce the best possible prediction.

1.3 MAJOR PROBLEM CLASSES

Given the diversity of learning problems, it is useful to organize these problems into major problem classes. A brief summary of some of the major dimensions of learning problems is given below.

- Online vs. offline - Online problems involve learning from experiences as they occur. For example, we might observe the time on a path through a network by traveling the path, or adjust the price of a product on the Internet and observe the revenue. We can try a decision that looks bad in the hopes of learning something, but we have to incur the cost of the decision, and balance this cost against future benefits. In offline problems, we might be working in a lab with a budget for running experiments, or we might set aside several weeks to run computer simulations. If we experiment with a chemical or process that does not appear promising, all we care about is the information learned from the experiment; we do not incur any cost from running an unsuccessful experiment. When our budget has been exhausted, we have to use our observations to choose a design or a process that will then be put into production.

- Objectives - Problems differ in terms of what we are trying to achieve. Most of the time we will focus on minimizing the expected cost or maximizing the expected reward from some system. However, we may be simply interested in finding the best design, or ensuring that we find a design that is within five percent of the best.

- The experimental decision - In some settings, we have a small number of choices such as drilling test wells to learn about the potential for oil or natural gas. The number of choices may be small, but each test can cost millions of dollars. Alternatively, we

might have to find the best set of 30 proposals out of 100 that have been submitted, which means that we have to choose from 3×10^{25} possible portfolios. Or we may have to choose the best price, temperature or pressure (a scalar, continuous parameter). We might have to set a combination of 16 parameters to produce the best results for a business simulator. Each of these problems introduce different computational challenges because of the size of the search space.

- The implementation decision - Collecting the best information depends on what you are going to do with the information once you have it. Often, the choices of what to observe (the experimental decision) are the same as what you are going to implement (finding the choice with the best value). But you might measure a link in a graph in order to choose the best path. Or we might want to learn something about a new material to make a decision about new solar panels or batteries. In these problems, the implementation decision (the choice of path or technology) is different from the choice of what to measure.

- What we believe - We may start by knowing nothing about the best system. Typically, we know something (or at least we will know something after we make our first experiment). What assumptions can we reasonably make about different choices? Can we put a normal distribution of belief on an unknown quantity? Are the beliefs correlated (if a laptop with one set of features has higher sales than we expected, does this change our belief about other sets of features)? Are the beliefs stored as a lookup table (that is, a belief for each design), or are the beliefs expressed as some sort of statistical model?

- The cost of an observation or experiment - One of the most important distinguishing features of different problem classes is the cost of running an experiment. There are two important problem classes:

  Inexpensive experiments A popular source of problems arise in internet settings such as finding the best ad to maximize ad-clicks, or finding the best movies or products to advertise on a retail site. This problem class tends to involve a very large number of choices (there can be thousands of ads or movies), but different choices can be tested relatively easily. An important issue here is that the logic for deciding what to test needs to be easy to compute.

  Expensive experiments There are many settings where a single experiment can be quite expensive, whether it is a day in a laboratory, hours to run a simulation on a computer, or weeks of testing in the field. Here, experiments need to be carefully designed, taking advantage of any prior information.

- The nature of an observation - Closely related to what we believe is what we learn when we run an experiment. Is the observation normally distributed? Is it a binary random variable (success/failure)? Are experiments run with perfect accuracy? If not, do we know the distribution of the error from running an experiment?

- Switching costs - The simplest problems assume that you can switch from one choice to another without penalty, but this is not always the case. We may have equipment set up to run a particular set of experiments; switching incurs a cost.

- Stationary vs. nonstationary settings - There are three important settings that describe the behavior of our the observations we make when we run an experiment:

Stationary distributions - Observations come from an unknown distribution which is not changing over time.

Nonstationary distributions - Observations are coming from distributions that are evolving over time (we hope that they are not changing too quickly). For example, sales of a product at a certain price depends on other exogenous factors that may change over time.

Learning problems - The more we try something, the better it may get.

- The experimental budget - Primarily relevant in offline settings, experimental budgets may be quite small (we can only run 20 laboratory experiments in a summer), or much larger (I might have a robotic scientist that can run hundreds of experiments).

- Planning constraints - There are fast-paced settings where we may have 50 milliseconds to make a decision, to problems where a single experiment is both time consuming and expensive, justifying significant computation before making the next experiment.

- Sequential vs. parallel (or batch) experimentation - We may have to run each experiment in sequence, which means we know the results of one experiment before deciding the next, but there are many settings where experiments can be run in parallel or batch. For example, we may try out a new drug on a group of patients at the same time.

- Belief states and physical/information states - All learning problems include a belief state which captures what we believe about the system. There are many learning problems where the belief state is the only information we have available. However, there are problems where we have access to additional information that changes the behavior of our problem (we call this the "information state"), or we may be managing a physical system that changes the decisions we can make (we call this the "physical state"). For example

  Information state - We may learn about the pricing decisions of our competitors before choosing the price we charge for our product, or we might obtain the characteristics of a patient before prescribing medication.

  Physical state - We may be managing a technical team that is collecting information about the spread of a disease. The decision of where to send the team next depends on where it is located now.

Given the richness of these problems, it should perhaps not be surprising that they have arisen in different communities who have devised strategies that are suited to different environments. In fact, it is important to understand the problem setting to appreciate the diversity of solution strategies that have evolved.

1.4 TWO APPLICATIONS

In this section, we provide a little more detail using two very classic problems: the newsvendor problem and the shortest path problem.

1.4.1 The newsvendor problem

Now consider another popular optimization problem known as the newsvendor problem. In this problem, we wish to order a quantity (of newspapers, oil, money, energy) x to satisfy a random demand D (that is, D is not known when we have to choose x). We earn p dollars per unit of satisfied demand, which is to say $\min(x, D)$, and we have to pay c dollars per unit of x that we order. The total profit is given by

$$F(x, D) = p \min(x, D) - cx.$$

The optimization problem is to solve

$$\max_x \mathbb{E}F(x, D).$$

There are a number of ways to solve stochastic optimization problems such as this. If the distribution of D is known, we can characterize the optimal solution using

$$P_D[x^* \leq D] = \frac{c}{p}.$$

where $P_D()$ is the cumulative distribution function for D. So, as the purchase cost c is decreased, we should increase our order quantity so that the probability that the order quantity is less than demand also decreases.

In many applications, we do not know the distribution of D, but we are able to make observations of D (or, we can observe if we have ordered too much or too little). Let x^{n-1} be the order quantity we chose after observing D^{n-1} which was our best guess of the right order quantity to meet the demand on day n, and let D^n be resulting demand. Now let g^n be the derivative of $F(x, D)$ given that we ordered x^{n-1} and then observed D^n. This derivative is given by

$$g^n = \begin{cases} p - c & \text{if } x \leq D \\ -c & \text{if } x > D \end{cases}$$

A simple method for choosing x^n is a stochastic gradient algorithm which looks like

$$x^n = x^{n-1} + \alpha_{n-1} g^n. \tag{1.1}$$

Here, α_{n-1} is a stepsize that has to satisfy certain conditions that are not important here. If the stepsize is chosen appropriately, it is possible to show that in the limit, x^n approaches the optimal solution, even without knowing the distribution of D in advance.

What our algorithm in equation (1.1) ignores is that our choice of x^n allows us to learn something about the distribution of D. For example, it might be that the purchase cost c is fairly high compared to the sales price p, which would encourage us to choose smaller values of x, where we frequently do not satisfy demand. But we might benefit from making some larger orders just to learn more about the rest of the demand distribution. By ignoring our ability to learn, the algorithm may not converge to the right solution, or it may eventually find the right solution, but very slowly. When we use optimal learning, we explicitly capture the value of the information we learn now on future decisions.

The newsvendor problem can be used to illustrate a variety of settings, including

- The observation of "sales" from our supply of "newspapers" might take a day, but there are situations where we are observing sales over a week or more.

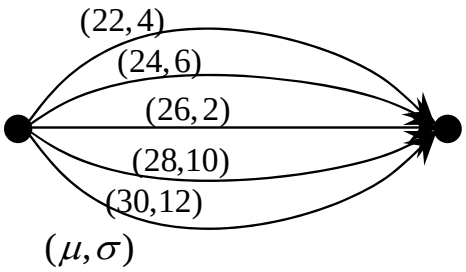


Figure 1.5 A simple shortest path problem, giving the current estimate of the mean and standard deviation (of the estimate) for each path.

- Before choosing the quantity x, we may be given information on the weather (that would affect the demand), or perhaps the price we might receive (imagine learning the spot price of frozen fish before deciding how much to allocate).

- The level of uncertainty - We may have a modest level of uncertainty in the demand D, or we may have to combine uncertainty about the demand (which could be high) and the price. If we are deciding the inventory for a new product to be sold over the Christmas season, we may have an exceptionally high level of uncertainty about the demand.

- Nonstationarity - The demand may change over time, reflecting seasonality or the effects of competition.

- A physical state - Now imagine that leftover newspapers are held until tomorrow. This means the decision today affects available inventory tomorrow, which then determines the set of feasible decisions.

1.4.2 Learning the best path

Consider the problem of finding the fastest way to get from your new apartment to your new job in Manhattan. We can find a set of routes from the Internet or from our GPS device, but we do not know anything about traffic congestion or subway delays. The only way we can get data to estimate actual delays on a path is to travel the path. We wish to devise a strategy that governs how we choose paths so that we strike a balance between experimenting with new paths and getting to work on time every day.

Assume that our network is as depicted in Figure 1.5. Let p be a specific path, and let $x_p = 1$ if we choose to take path p. After we traverse the path, we observe a cost $\hat{c}_p$. Let μ_p denote the true mean value of $\hat{c}_p$, which is of course unknown to us. After n trials, we can compute a sample mean $\bar{\theta}_p^n$ of the cost of traversing path p, and a sample variance $\hat{\sigma}_p^{2,n}$ using our observations of path p. Of course, we only observe path p if $x_p^n = 1$, so we might

compute these statistics using

$$N_p^n = \sum_{k=1}^{n} x_p^k, \tag{1.2}$$

$$\bar{\theta}_p^n = \frac{1}{N_p^n} \sum_{k=1}^{n} x_p^k \hat{c}_p^k, \tag{1.3}$$

$$\hat{\sigma}_p^{2,n} = \frac{1}{N_p^n - 1} \sum_{k=1}^{n} x_p^k (\hat{c}_p^k - \bar{\theta}_p^n)^2. \tag{1.4}$$

Note that $\hat{\sigma}_p^{2,n}$ is our estimate of the variance of $\hat{c}_p$ by iteration n (assuming we have visited path p $N_p^n > 1$ times). The variance of our estimate of the mean, $\bar{\theta}_p^n$, is given by

$$\bar{\sigma}_p^{2,n} = \frac{1}{N_p^n} \hat{\sigma}_p^{2,n}.$$

Now we face the challenge: which path should we try? Let's start by assuming that you just started a new job, you have been to the Internet to find different paths, but you have not tried any of them. If your job involves commuting from a New Jersey suburb into Manhattan, you have a mixture of options that include driving (various routes) and commuter train, with different transit options once you arrive in Manhattan. But you do have an idea of the length of each path, and you may have heard some stories about delays through the tunnel into Manhattan, and a few stories about delayed trains. From this, you construct a rough estimate of the travel time on each path, and we are going to assume that you have at least a rough idea of how far off these estimates may be. We denote these initial estimates using

$\bar{\theta}_p^0$ = initial estimate of the expected travel time on path p,

$\bar{\sigma}_p^0$ = initial estimate of the standard deviation of the difference between $\bar{\theta}_p^0$ and the truth.

If we believe that our estimation errors are normally distributed, then we think that the true mean, μ_p, is in the interval $(\mu_p - z_{\alpha/2}\bar{\sigma}_p^0, \mu_p + z_{\alpha/2}\bar{\sigma}_p^0)$ α percent of the time. If we assume that our errors are normally distributed, we would say that we have an estimate of μ_p that is normally distributed with parameters $(\bar{\theta}_p^0, (\bar{\sigma}_p^0)^2)$.

So which path do you try first? If our priors are as shown in Figure 1.5, presumably we would go with the first path, since it has a mean path time of 22 minutes, which is less than any of the other paths. But our standard deviation around this belief is 4, which means we believe this could possibly be as high as 30. At the same time, there are paths with times of 28 and 30 with standard deviations of 10 and 12. This means that we believe that these paths could have times that are even smaller than 20. Do we always go with the path that we think is the shortest? Or do we try paths that we think are longer, but where we are just not sure, and there is a chance that these paths may actually be better?

If we choose a path we think is best, we say that we are *exploiting* the information we have. If we try a path because it might be better, which would help us make better decisions in the future, we say that we are *exploring*. Exploring a new path, we may find that it is an unexpectedly superior option, but it is also possible that we will simply confirm what we already believed. We may even obtain misleading results – it may be that this one route was experiencing unusual delays on the one day we happened to choose it. Nonetheless, it

is often desirable to try something new to avoid becoming stuck on a suboptimal solution just because it "seems" good. Balancing the desire to explore versus exploit is referred to in some communities as the *exploration vs. exploitation* problem. Another name is the *learn vs. earn* problem. Regardless of the name, the point is the lack of information when we make a decision, and the value of new information in improving future decisions.

1.5 LEARNING FROM DIFFERENT COMMUNITIES

The challenge of efficiently collecting information is one that arises in a number of communities. The result is a lot of parallel discovery, although the questions and computational challenges posed by different communities can be quite different, and this has produced diversity in the strategies proposed for solving these problems. Below we provide a rough list of some of the communities that have become involved in this area.

- Simulation optimization - The simulation community often faces the problem of tuning parameters that influence the performance of a system that we are analyzing using Monte Carlo simulation. These parameters might be the size of a buffer for a manufacturing simulator, the location of ambulances and fire trucks, or the number of advance bookings for a fare class for an airline. Simulations can be time consuming, so the challenge is deciding how long to analyze a particular configuration or policy before switching to another one.

- The ranking and selection problem - This is a statistical problem that arises in many settings, including the simulation optimization community. It is most often approached using the language of classical frequentist statistics (but not always), and tends to be very practical in its orientation. In ranking and selection, we assume that for each experiment, we can choose from a set of alternatives (there is no cost for switching from one alternative to another). Although the ranking and selection framework is widely used in simulation optimization, the simulation community recognizes that it is easier to run the simulation for one configuration a little longer than it is to switch to the simulation of a new configuration.

- The bandit community - There are two subcommunities that work on learning problems known as the *multiarmed bandit problem*. It evolved originally within applied probability in the 1950's, which led to a breakthrough known as Gittins indices that represented the first computable, optimal policy for a particular learning problem. This work spawned the search for what became known as "index policies" for solving variations of learning problems. The difficulty was that while these were computable in principle, they were difficult to compute. In the 1980's, a new line of research emerged within computer science with the discovery that some simple policies, known as "upper confidence bounds," exhibited attractive bounds on their performance that suggested that they would work well.

- Global optimization of expensive functions - The engineering community often finds a need to optimize complex functions of continuous variables. The function is sometimes a complex piece of computer software that takes a long time to run, but the roots of the field come from geospatial applications. The function might be deterministic (but not always), and a single evaluation can take an hour to a week or more.

- Learning in economics - Economists have long studied the value of information in a variety of idealized settings. This community tends to focus on insights into the economic value of information, rather than the derivation of specific procedures for solving information collection problems.
- Active learning in computer science - A popular problem in computer science is classification: Is a news article fake? Does a patient have cancer? Will a customer purchase a product or movie? Whereas most machine learning is performed with a static (batch) dataset, we may be able to control our inputs (for example, by choosing which movies to show a user, or choosing a treatment regime for a patient). Having the ability to choose our data is known as active learning, and the goal is often to maximize classification success.
- Statistical design of experiments - A classical problem in statistics is deciding what experiments to run. For certain objective functions, it has long been known that experiments can be designed deterministically, in advance, rather than sequentially. Our focus is primarily on sequential information collection, but there are important problem classes where this is not necessary.
- Frequentist versus Bayesian communities - It is difficult to discuss research in optimal learning without addressing the sometimes contentious differences in styles and attitudes between frequentist and Bayesian statisticians. Frequentists look for the truth using nothing more than the data that we collect, while Bayesians would like to allow us to integrate expert judgment.
- Optimal stopping - There is a special problem class where we have the ability to observe a single stream of information such as the price of an asset. As long as we hold the asset, we get to observe the price. At some point, we have to make a decision whether we should sell the asset or continue to observe prices (a form of learning). Another variant is famously known as the "secretary problem" where we interview candidates for a position (or offers for an asset); after each candidate (or offer) we have to decide if we should accept and stop or reject and continue observing.
- Approximate dynamic programming/reinforcement learning - Approximate dynamic programming, widely known as reinforcement learning in the computer science community, addresses the problem of choosing an action given a state which generates a reward and takes us to a new state. We do not know the exact value of the downstream state, but we might decide to visit a state just to learn more about it. This is generally known as the "exploration vs. exploitation" problem, and this setting has motivated a considerable amount of research in optimal learning.
- Psychology - Not surprisingly, the tradeoff between exploration and exploitation is a problem that has to be solved by people (as well as other animals ranging from chimpanzees to ants) for problems ranging from finding foot to finding mates. Psychologists have been attracted to the problem of understanding how people make these decisions.
- Learning in the plant and animal sciences - Both plants and animals have to find food and mates, which requires that they also solve the exploration vs. exploitation problem. Two-armed slime molds have to decide which arm to extend; vines have to decide in which direction to grow along the ground to find a tree; predators on land and sea have to devise strategies for searching for food.

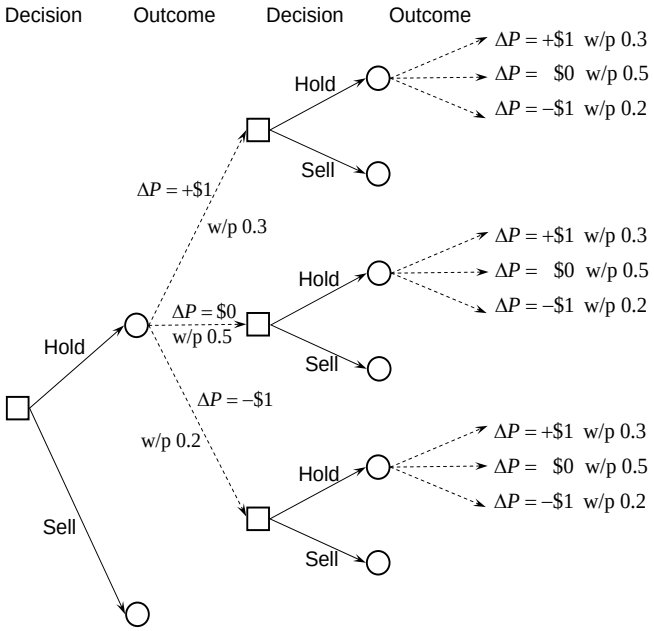


Figure 1.6 Illustration of a decision tree to determine if we should hold or sell a stock, where the stock might go up or down $1, or stay the same in each time period.

Readers who wish to study this field seriously will encounter the contributions of these (and perhaps other) communities. It is not possible to cover all the issues and perspectives of these communities in this volume, but we do provide a foundation that should make it possible for students and researchers to understand the issues and, in some cases, challenge conventional wisdom within specific communities.

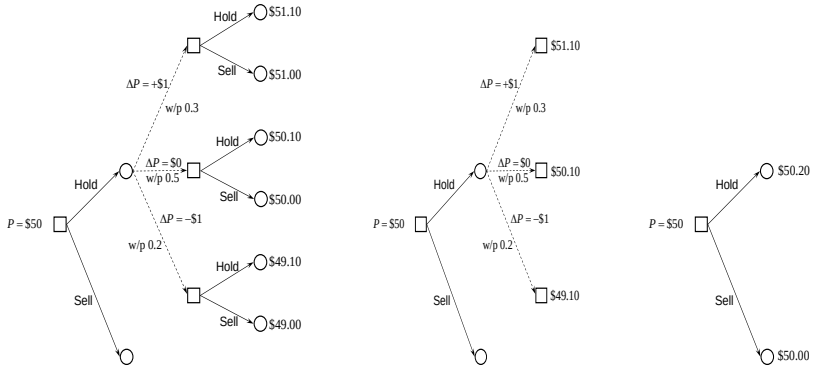
1.6 INFORMATION COLLECTION USING DECISION TREES

The simplest types of information collection problems arise when there is a small number of choices to collect information. Should you check the weather report before scheduling a baseball game? Should you purchase an analysis of geologic formulations before drilling for oil? Should you do a statistical analysis of a stock price before investing in the stock?

These are fairly simple problems that can be analyzed using a decision tree, which is a device that works well when the number of decisions, as well as the number of possible outcomes, is small and discrete. We begin by first presenting a small decision tree where collecting information is not an issue.

1.6.1 A basic decision tree

Decision trees are a popular device for solving problems that involve making decisions under uncertainty, because they illustrate the sequencing of decisions and information so clearly. Figure 1.6 illustrates the decision tree that we might construct to help with the decision of whether to hold or sell a stock. In this figure, square nodes are decision nodes,



1.7(a) 1.7(b) 1.7(c)

Figure 1.7 (a) Decision tree with second outcome replaced by expected value; (b) Decision tree with second decision replaced by best expected value; (c) Decision tree with first outcome replaced by expected value, producing a deterministic decision.

where we get to choose from a set of actions (where the number of actions cannot be too large). Circle nodes are outcome nodes, where something random happens, such as the change in the price of the stock. The solid lines represent decisions, while dashed lines are random outcomes. In our example, there is no cost for holding a stock, and the random outcome represents a change in the price of the stock.

If we hold the stock (which currently can be sold at a price of $50), it might go up or down by a dollar, with probabilities of 0.3 and 0.2, respectively, or hold at the same price with a probability of 0.5. After observing the change in the price, we again have a chance to hold or sell the stock. If we continue holding, the stock might go up or down by a dollar, or stay the same.

We can solve the problem of whether to hold or sell the stock initially by doing what is called "rolling back the decision tree." Figure 1.7(a) shows the tree after the first rollback. Here, we have taken the final random outcome and replaced it with the expected value, which gives us the result that we expect the price to go up by $0.10 if we hold the stock. We now have the option of holding or selling, which is a decision that we control. Since the price is likely to go up if we hold, we make this choice.

In Figure 1.7(b), we now use the expected value of the second decision to give us what we will earn at the end of each random outcome resulting from the first decision. We have the same three outcomes (up or down $1, or stay the same), but each outcome produces an expected return of $51.10, $50.10 or $49.10. Now we again have to find the expectation over these returns, which gives us the expected value of $50.20. Finally, we have to evaluate our original decision to hold or sell, and of course we are willing to hold for $50.20 rather than sell now for $50.

1.6.2 Decision tree for offline learning

The previous example provided a quick illustration of a basic decision tree. A common decision problem is whether or not we should collect information to make a decision. For

example, consider a bank that is considering whether it should grant a short term credit loan of $100,000. The bank expects to make $10,000 if the loan is paid off on time. If the loan is defaulted, the bank loses the amount of the loan.

From history, the bank knows that 95 percent of loans are repaid in full, while 5 percent default. If the bank purchases the credit report, this information will allow the bank to classify the customer into one of three groups: 52 percent fall into the top A rating, 30 percent fall into the middle B rating, while 18 percent fall into the lower C rating with the highest risk of default. The company selling the credit report provides the joint distribution $P(\text{Credit}, \text{Default})$ that a customer will receive each credit rating, and whether it defaulted or not. This data is summarized in Table 1.3.

We need to understand how the information from the credit report changes our belief about the probability of a default. For this, we use a simple application of Bayes' theorem, which states

$$\begin{aligned} P(\text{Default} \mid \text{Credit}) &= \frac{P(\text{Credit} \mid \text{Default})P(\text{Default})}{P(\text{Credit})} \\ &= \frac{P(\text{Credit}, \text{Default})}{P(\text{Credit})}. \end{aligned}$$

Bayes' theorem allows us to start with our initial estimate of the probability of a default, $P(\text{Default})$, then use the information "Credit" from the credit history and turn it into a *posterior* distribution $P(\text{Default} \mid \text{Credit})$. The results of this calculation are shown in the final two columns of Table 1.3.

		P(Credit,Default)		P(Default | Credit)	
Credit rating	P(Credit)	No	Yes	No	Yes
A	0.52	0.51	0.01	0.981	0.019
B	0.30	0.28	0.02	0.933	0.067
C	0.18	0.16	0.02	0.889	0.111
	P(Default)=	0.95	0.05		

Table 1.3 The marginal probability of each credit rating; the joint probability of a credit rating and whether someone defaults on a loan, and the conditional probability of a default given a credit rating.

Using this information, we can construct a new decision tree, shown in Figure 1.8. Unlike our first decision tree in Figure 1.7, we now see that the decision to collect information changes the downstream probabilities.

We repeat the exercise of rolling back the decision tree in Figure 1.9. Figure 1.9(a) shows the expected value of the decision to grant the loan given the information about the credit history. We see that if the grantee has an A or B credit rating, it makes sense to grant the loan, but not if the rating is C. Thus, the information from the credit report has the effect of changing the decision of whether or not to grant the loan. After we roll the tree back to the original decision of whether to purchase the credit report, we find that the credit report produces an expected value of $4,900, compared to $4,500 that we would expect to receive without the credit report. This means that we would be willing to pay up to $400 for the credit report.

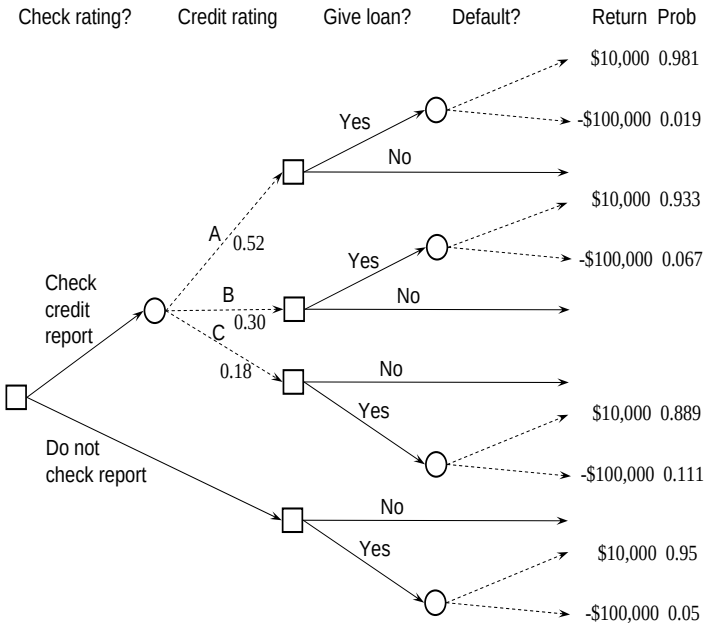
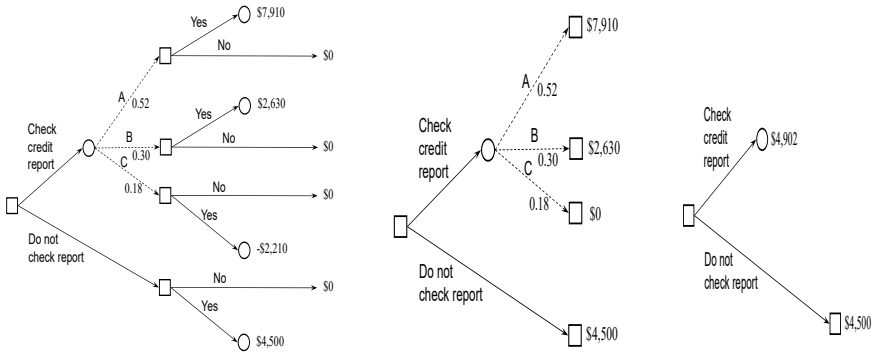


Figure 1.8 The decision tree for the question of whether we should purchase a credit risk report.



1.9(a) 1.9(b) 1.9(c)

Figure 1.9 (a) Decision tree with final default replaced with expected value; (b) Decision tree with second decision replaced by best expected value; (c) Decision tree with the uncertainty of the credit risk report replaced with its expected value.

1.6.3 Decision tree for online learning

Now consider a problem where we learn as we go. We use the setting of trying to identify the best hitter on a baseball team. The only way to collect information is to put the hitter into the lineup. Assume that we are evaluating hitters for the fourth position in the lineup,

Player	No. hits	No. at bats	Average
A	36	100	0.360
B	1	3	0.333
C	7	22	0.318

Table 1.4 History of hitting performance for three candidates.

typically reserved for power hitters. Part of what we are trying to learn is how a hitter actually performs in game situations.

Assume that we have three candidates for the position. The information we have on each hitter from previous games is given in Table 1.4. If we choose player A, we have to balance the likelihood of getting a hit, and the value of the information we gain about his true hitting ability, since we will use the event of whether or not he gets a hit to update our assessment of his probability of getting a hit. We are going to again use Bayes' theorem to update our belief about the probability of getting a hit. To do this, we have to make some probabilistic assumptions that are not relevant to our discussion here; we defer until Chapter 2, Section 2.6.4, the discussion of the model that we use to calculate the updated probabilities. Fortunately, this model produces some very intuitive updating equations. Let H^n be number of hits a player has made in n at-bats. Let $\hat{H}^{n+1} = 1$ if a hitter gets a hit in his $(n+1)$st at-bat. Our prior probability of getting a hit after n at-bats is

$$\mathbb{P}[\hat{H}^{n+1} = 1|H^n, n] = \frac{H^n}{n}.$$

Once we observe $\hat{H}^{n+1}$, it is possible to show that the posterior probability is

$$\mathbb{P}[\hat{H}^{n+2} = 1|H^n, n, \hat{H}^{n+1}] = \frac{H^n + \hat{H}^{n+1}}{n+1}.$$

In other words, all we are doing is computing the batting average (hits over at-bats). In Chapter 2 we are going to put a lot more rigor behind this, but for now, we are going to take advantage of the simple, intuitive updating equations that this theory provides.

Our challenge is to determine whether we should try player A, B or C right now. At the moment, A has the best batting average of 0.360, based on a history of 36 hits out of 100 at-bats. Why would we try player B, whose average is only 0.333? We easily see that this statistic is based on only three at bats, which would suggest that we have a lot of uncertainty in this average.

We can study this formally by setting up the decision tree shown in Figure 1.10. For practical reasons, we can only study a problem that spans two at-bats. We show the current prior probability of a hit, or no hit, in the first at bat. For the second at-bat, we show only the probability of getting a hit, to keep the figure from becoming too cluttered.

Figure 1.11 shows the calculations as we roll back the tree. Figure 1.11(c) shows the expected value of playing each hitter for exactly one more at bat using the information obtained from our first decision. It is important to emphasize that after the first decision, only one hitter has had an at bat, so the batting averages only change for that hitter. Figure 1.11(b) reflects our ability to choose what we think is the best hitter, and Figure 1.11(a) shows the expected value of each hitter before any at bats have occurred. We use as our reward function the expected number of total hits over the two at bats. So, if we choose

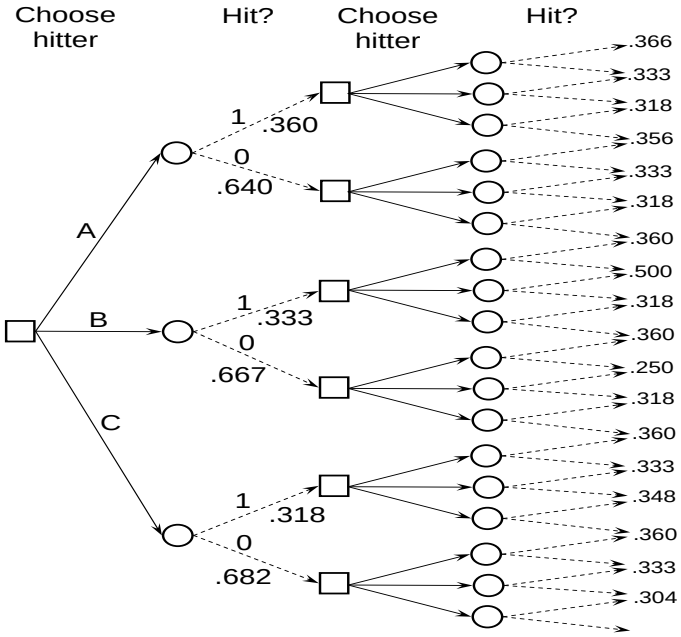


Figure 1.10 The decision tree for finding the best hitter.

batter A, the expected value is

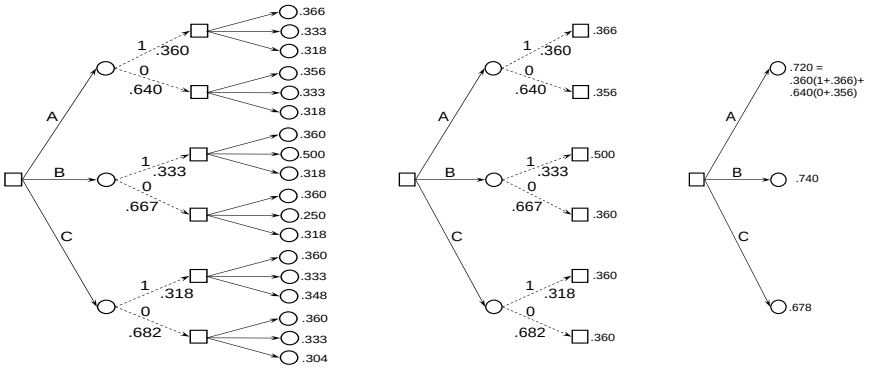
$$.720 = .360(1 + .366) + .640(0 + .356)$$

where 0.360 is our prior belief about his probability of getting a hit; .366 is the expected number of hits in his second at bat (the same as the probability of getting a hit) given that he got a hit in his first at bat. If player A did not get a hit in his first at bat, his updated probability of getting a hit, 0.356, is still higher than any other player. This means that if we have only one more at bat, we would still pick player A even if he did not get a hit in his first at bat.

Although player A initially has the highest batting average, our analysis says that we should try player B for the first at bat. Why is this? On further examination, we realize that it has a lot to do with the fact that player B has had only three at bats. If this player gets a hit, our estimate of his probability of getting a hit jumps to 0.500, although it drops to 0.250 if he does not get a hit. If player A gets a hit, his batting average moves from 0.360 to 0.366, reflecting the weight of his much longer record. This is our first hint that it can be useful to collect information about choices where there is the greatest uncertainty.

1.6.4 Discussion

These simple examples illustrate some of the issues that are fundamental to information collection. First, we see that collecting information changes our beliefs about uncertain quantities. Second, when we change our beliefs, we change our decisions which produces an economic impact. From this analysis, we can compute the expected value of the information.



1.11(a) 1.11(b) 1.11(c)

Figure 1.11 (a) Expected value of a hit in the second at bat; (b) Value of best hitter after one at bat; (c) Expected value of each hitter before first at bat.

In the credit risk example, the *experimental decision* (purchasing the credit risk report) was completely separate from the *implementation decision* (whether or not to give a loan). There are many problems where we learn from our actions. For example, we might grant a loan, and then learn from this experience. This means giving a loan allows us to observe whether or not someone defaults on the loan, which might then change our behavior in the future. We can ignore the value of this information when making a decision, but one of the goals of this book is to use this value of information in our decisions.

Most of the problems that we address in this book can be visualized as a decision tree. Decisions trees, however, grow very quickly with the number of decisions being made, the number of random outcomes that might occur, and the number of time periods. For example, the very simple decision tree that we used in Section 1.6.3 to analyze which baseball player we should use grows quickly if we look more than two at bats into the future. With three players to choose from, and only two possible outcomes, the tree grows by a factor of six for every at bat we add to our planning horizon. If we want to plan over the next 10 at bats, our decision tree would have $6^{10} = 60,466,176$ end points. And this is a tiny problem.

Although decision trees are often impractical as a computational device, they are extremely useful as a conceptual mechanism for understanding sequential decision problems.

1.7 WEBSITE AND DOWNLOADABLE SOFTWARE

The book is supported by additional material at the website

`http://optimallearning.princeton.edu/`

The website will provide additional readings, chapter notes and comments, sample applications (many drawn from the projects generated by students in the course *Optimal Learning* taught at Princeton University), and downloadable software. Most of the software is in the form of MATLAB modules that offer implementations of some of the algorithms. One module, the Optimal Learning Calculator, uses a spreadsheet front-end which talks to

a series of Java-based modules which implement some of the algorithms. The spreadsheet offers an interactive environment which makes it possible to experiment with a variety of learning policies. You can solve a learning problem manually, or simulate any of a series of policies on problems. These problems can be randomly generated, or entered manually by hand.

1.8 GOALS OF THIS BOOK

There are a number of communities that address the general problem of collecting information. These communities span economics, computer science, statistics, and operations research (which in turn includes subcommunities such as decision analysis, applied probability, and simulation). One of our goals is to bring these communities together under a common vocabulary and notation.

Given the diversity of problems where learning arises, it is not surprising that a multitude of techniques have evolved to address these problems. Our presentation reviews a number of these techniques, but considerable emphasis is put on a concept called the *knowledge gradient* which guides experiments based on the marginal value of a single experiment. The knowledge gradient is comparable to the gradient in classical optimization, but focuses on the value of information. The power of the knowledge gradient is that it is a simple concept that can be applied to a wide range of problems. In fact, this technique opens up new problems, and allows us to consider learning in settings where it has previously not been considered. Our empirical work to date suggests that it is a surprisingly robust strategy, in that it is usually competitive with competing techniques, while often outperforming other methods without the need for tunable parameters.

The book is aimed at students and professionals with a basic course in statistics and a full course in probability. The presentation emphasizes concepts and practical tools over heavy mathematical development.

PROBLEMS

1.1 Pick a problem that involves sequential information and decision processes. Ideally, the problem is one of special interest to you. Give a short description of the problem in plain English. Then describe the following components listed below. In each case, a candidate answer is provided using the setting of finding a new compound for storing energy in a battery (this is a fairly complicated example - there are many settings which are much simpler).

a) What decision are you making that determines the information you collect? [Example: Testing a particular molecule.]

b) Precisely what information is being observed as a result of your choice? [Example: The amount of energy stored per pound.]

c) What decision are you going to make with this information? [Example: We will use this information to decide which type of battery is the most economical, which in turn will impact our decision of whether to use batteries, flywheels or hydroelectric reservoirs as our major form of storage.]

d) What is the economic impact of the decision? [Example: The information will allow the Department of Energy to determine if it should invest in rooftop solar panels, where batteries are needed to smooth out variations in solar energy, or more distant wind turbines, which can be used in conjunction with water reservoirs.]

1.2 For each of the situations below, identify whether it is an on-line or off-line learning problem. Then, identify the experimental decision (what is being measured), and how you evaluate the quality of the experiments (that is, what is the value of the information you are collecting).

a) The adventurer Steve Fossett was lost somewhere in Nevada. The problem is to design a search process that might identify the location of the plane he was flying.

b) You would like to find the best price for a downloadable song by adjusting the price.

c) You are using a computer simulator of an airline to test the effect of different levels of schedule slack to achieve a given level of reliability.

d) A bank evaluates loans based on a set of attributes determined from a loan application. Some loans are turned down, others are approved. For the loans that are approved, the bank can observe if the loan is repaid or defaults. The bank can then later use this data to correlate the default rate to the attributes of the loan. Since the bank is not able to observe defaults on loans which are not approved, it occasionally may decide to grant a loan which its policy suggests should be turned down, just to observe the information.

e) A scientist is testing different molecular compounds to find a drug that is the most effective at killing cancer cells.

f) A doctor administers drugs to control the blood pressure in a patient. The doctor will adjust both the type of medication as well as the dosage, observing the effect on the patient's blood pressure.

1.3 In Section 1.6.2, we addressed the problem of whether a bank should request a credit report. Repeat the exercise of finding out whether the bank should purchase the credit report, and determine how much the bank would be willing to pay for the report. As before, assume a loan is for $100,000. If the loan is paid off, the bank makes $10,000. If the loan defaults, assume that the bank loses on average $30,000 (since a portion of the loan would have been repaid). Assume that 85 percent of the loans are repaid. The bank has collected past statistics on credit reports which are expressed as the conditional probabilities in Table 1.5 below. So, 72 percent of the loans that did not default had a credit rating of A.

		P(Credit rating | Default)		
Default	P(Credit)	A	B	C
No	0.85	0.72	0.24	0.04
Yes	0.15	0.25	0.57	0.18

Table 1.5 Data for exercise 1.3.

1.4 In Table 1.3 for the credit risk problem, the probability $P(Credit = C, Default = Yes) = 0.16$ and $P(Credit = C, Default = No) = 0.01$. Change these probabilities to 0.17 and 0.01, respectively, and solve the decision tree in Figure 1.9 again. How did this change in the data change the behavior of the system after reviewing the credit risk report? What is the new value of the information in the credit risk report? Given an intuitive explanation.

1.5 Return to the decision tree in Section 1.6.3 where we are trying to decide which hitter to use. This decision tree has been implemented in the spreadsheet available on the book website at

`http://optimallearning.princeton.edu/exercises/BaseballTree3levels.xlsx`

Note that the decision tree considers three successive at-bats.

- The decision tree takes into account the information we gain from observing the outcome of the first at bat (whether it be player A, B or C). How would your answer have changed if we formulated the decision tree without taking advantage of the information gained from the first at bat?

- Use the spreadsheet to compute the value of using each of the three batters, while changing the batting statistics from player B from 1 for 3 to 2 for 6 to 3 for 9 to 4 four 12 and finally 5 for 15. Draw a graph giving the value of choosing each of the three hitters (in the first at bat) for all five sets of batting statistics. Explain intuitively why your choice of who to use for the first at bat changes.

1.6 The decision tree for our baseball problem in section 1.6.3 can be modeled using the following notation:

S^n = The "state variable" capturing what we know *after* n at-bats (by any hitter). S^0 is what we know initially. We re-visit precisely what a state variable is below.

$\mathcal{X}$ = The set of potential batters,
= $\{A, B, C\}$

x = A particular batter from the set $\mathcal{X}$.

x^n = The choice of hitter made using the information in S^n, which means this is the person who will hit in the $n + 1st$ at bat.

$\hat{H}^{n+1}_{x^n}$ = The random variable giving the number of hits that batter x^n will get in the $n + 1st$ at-bat.

We will find it useful to define the *history* of the process:

h^n = The history of the information process, which starts with our initial information S^0, followed by the sequencing of the choice of batter, and the outcome of the at-bat.
= $(S^0, x^0, \hat{H}^1_{x^0}, S^1, x^1, \hat{H}^2_{x^1}, \ldots, S^n, x^n)$

We can solve this problem using *Bellman's equation* which we briefly introduced in class. For the baseball problem, this would be written

$$V^n(S^n) = max_{x \in \mathcal{X}} \mathbb{E}\{\hat{H}^{n+1}_x + V^{n+1}(S^{n+1})|S^n\}.$$

The expectation means averaging over the possible outcomes of the random variable $\hat{H}^{n+1}_x$ given what we know after n at-bats, which is captured in S^n.

The state S^{n+1} depends on the state S^n, the decision x and the outcome of the random variable $\hat{H}^{n+1}_x$, but we cannot write this relationship until we define the state variable, which we do below.

In this formulation, S^n is also known as the *pre-decision* state variable, because it captures what we know before we make a decision. It is often useful (as it is in this example) to define a *post-decision* state variable, which we denote $S^{x,n}$, which is the state right after we make a decision x^n using the information in the pre-decision state S^n.

We are going to use our batting example to explore this equation a bit more.

a) There are two ways to think of state variables in this example. The first is to think of the state variable as the history. Give the pre-decision state S^n and the post-decision state $S^{x,n}$ for $n = 2$ using the notation above, and show what these refer to in our decision tree. Then give a numerical example of each (you will need to pick actual examples of hitters and whether they got a hit or not).

b) For our problem with three hitters, and two outcomes (getting a hit or not), give a general formula for the number of states in our system (that is, the number of histories) after n at-bats. What is this number for $n = 10$? Is there more than one way to get to a particular state S^n (explain)?

c) Using this state variable, write S^{n+1} as a function of S^n, the decision x^n, and the outcome $\hat{H}^{n+1}_{x^n}$.

d) A more compact way of describing our system is to define:

H_x^n = The total number of hits made by batter x after n at-bats. Note that we are given values for H_x^0 before we start solving our problem.

Using this notation, give a different formulation of the state variable S^n (keep in mind that the state S^n has to capture what we know for all three batters). Show that we can calculate the probability that $\hat{H}_x^{n+1}$ equals 0 or 1 given just the information in S^n.

e) Using this representation of the state variable, write S^{n+1} as a function of S^n, the decision x^n, and the outcome $\hat{H}_{x^n}^{n+1}$.

f) Using this new state representation, give the size of the state space after n iterations. Is there more than one way to get to a state S^n? If so, give an example for $n = 3$ by giving an example of a state, and different trajectories for getting to that state.

CHAPTER 2

ADAPTIVE LEARNING

At the heart of any learning problem is a *belief model*, which is some type of statistical model that characterizes what we know about the behavior of our system. However, unlike the usual statistical model which is just a point estimate, our belief models will always include an explicit representation of the uncertainty in our model. Thus, if we have an estimate that driving a particular route is likely to take 25 minutes, our belief model will include a distribution about what the true mean might be. These beliefs will evolve as new information arrives, and our goal in this book is to acquire information that has the biggest impact on the quality of our decisions.

There are two perspectives that we can take when forming a belief, known as the *frequentist view* and the *Bayesian view*. In the frequentist view, we begin with no knowledge at all about our parameters, and our beliefs are formed entirely by the results of experiments that we run. Since experiments are inherently noisy, we can repeat a series of experiments and obtain different estimates of the parameters. If we repeat the experiments often enough, we can form a frequency distribution of any parameter that we estimate using the experiments.

In the Bayesian view, we start with initial beliefs about parameters, known as the *prior distribution*, which is formed before we make any observations. After an experiment, we combine the prior distribution with an experiment to form a posterior distribution. This then becomes the next prior distribution. This distribution is our *distribution of belief* about the true values of a set of parameters, formed from a combination of our initial belief and subsequent observations. By contrast, in the frequentist view, we form a probability distribution of estimates of parameters that reflect the variation in the observations.

Optimal Learning. By Warren B. Powell and Ilya O. Ryzhov

We begin with an overview of major classes of belief models, including lookup table, parametric and nonparametric models.

2.1 BELIEF MODELS

At the heart of any learning problem is some sort of statistical model we refer to as the *belief model*, which captures not only our estimate of how a function or system will respond to a controllable input x, but also the uncertainty in our estimate of the function.

We can represent our function or system as $f(x)$. Often, this is a function of a controllable input x (drug dosages, price of a product, concentration of a chemical), and a random variable W which covers any uncontrollable (we sometimes refer to these as exogenous) inputs. In principle, we might write

$$f(x) = \mathbb{E}F(x, W),$$

where the expectation $\mathbb{E}$ averages over all possible outcomes W.

Most of the time in this book we are going to treat x as being a discrete choice. We can write the set of choices as $\mathcal{X} = \{x_1, x_2, \ldots, x_M\}$. These might be discrete choices (color, choice of material, choice of drug), or discretized values of a continuous parameter (price, length, dosage). If x is multidimensional, we might view $\mathcal{X}$ as a sample of possible values of x. However, we will occasionally address situations where x is continuous, either scalar, or a continuous vector.

Belief models can be described as coming in one of three flavors:

Lookup tables Assume that we have a discrete set of choices $x \in \mathcal{X} = \{x_1, \ldots, x_M\}$ and let

$$\mu_x = f(x) = \mathbb{E}_W F(x, W).$$

A lookup table representation refers to an estimate $\bar{\mu}_x^n \approx \mu_x$ for each $x \in \mathcal{X}$. So, if $M = 100$, then we need to estimate 100 different parameters.

Parametric models There are settings where the set $\mathcal{X}$ can become quite large, as occurs when x is multidimensional, or continuous. In such a setting, we might want to write our belief model as

$$\begin{aligned} \bar{f}(x|\theta) &\approx \mathbb{E}F(x, W) \\ &= \sum_{f \in \mathcal{F}} \theta_f \phi_f(x), \end{aligned} \tag{2.1}$$

where $\phi_f(x),\ f \in \mathcal{F}$ is a set of features. For example, x might be a choice of a movie, while $\phi_f(x)$ captures features such as genre. Now, instead of having a belief μ_x for every movie x, we just have to estimate a vector of parameters $(\theta_f),\ f \in \mathcal{F}$ for a presumably small set of features. Equation (2.1) illustrates a linear model, which means it is linear in θ (the features $\phi_f(x)$ may be highly nonlinear in x). We may also need to use a nonlinear model such as

$$\bar{f}(x|\theta) = \begin{cases} 1 & \text{If } x < \theta^{min} \\ 0 & \text{If } \theta^{min} < x < \theta^{max} \\ -1 & \text{If } x > \theta^{max} \end{cases}$$

Another example might be a logistic function

$$\bar{f}(x|\theta) = \frac{e^{\theta_0+\theta_1 x}}{1+e^{\theta_0+\theta_1 x}}.$$

Nonparametric models Nonparametric models allow us to create estimates without having to assume a functional form such as our linear model above, but the price is high. Imagine we have a set of observations $(x^n, \hat{f}^n),\ n = 1, \ldots, N$. We can create an approximation $\bar{f}(x)$ by using a local average around x, consisting of an average of points $\hat{f}^n$ with weights that are inversely proportional to the distance $\|x - x^n\|$. Nonparametric models are quite flexible, but can be hard to use, and as a result will not play a major role in our treatment.

The remainder of this chapter will focus on methods for updating lookup table belief models. For this setting, we are going to use W_x^n as the observation of our function $f(x)$, which means that if we choose to evaluate the function at $x = x^n$, then we are going to observe

$$\begin{aligned} W_{x^n}^{n+1} &= f(x^n) + \varepsilon^{n+1} \\ &= \mu_{x^n} + \varepsilon^{n+1}. \end{aligned}$$

Below, we describe how to use frequentist and Bayesian statistics to produce estimates of μ_x for each x.

2.2 THE FREQUENTIST VIEW

The frequentist view is arguably the approach that is most familiar to people with an introductory course in statistics. Assume we are trying to estimate the mean μ of a random variable W which might be the performance of a device or policy. Let W^n be the nth sample observation. Also let $\bar{\mu}^n$ be our estimate of μ, and $\hat{\sigma}^{2,n}$ be our estimate of the variance of W. We know from elementary statistics that we can write $\bar{\mu}^n$ and $\hat{\sigma}^{2,n}$ using

$$\bar{\mu}^n = \frac{1}{n}\sum_{m=1}^{n} W^m \tag{2.2}$$

$$\hat{\sigma}^{2,n} = \frac{1}{n-1}\sum_{m=1}^{n}(\hat{f}^m - \bar{\mu}^n)^2. \tag{2.3}$$

The estimate $\bar{\mu}^n$ is a random variable (in the frequentist view) because it is computed from other random variables, namely $W^1, W^2, \ldots, W^n$. Imagine if we had 100 people each choose a sample of n observations of W. We would obtain 100 different estimates of $\bar{\mu}^n$, reflecting the variation in our observations of W. The best estimate of the variance of the estimator $\bar{\mu}^n$ is easily found to be

$$\bar{\sigma}^{2,n} = \frac{1}{n}\hat{\sigma}^{2,n}.$$

Note that as $n \to \infty$, $\bar{\sigma}^{2,n} \to 0$, but $\hat{\sigma}^{2,n} \to \sigma^2$ where σ^2 is the true variance of W. If σ^2 is known, there would be no need to compute $\hat{\sigma}^{2,n}$ and $\bar{\sigma}^{2,n}$ would be given as above with $\hat{\sigma}^{2,n} = \sigma^2$.

We can write these expressions recursively using

$$\bar{\mu}^n = \left(1 - \frac{1}{n}\right)\bar{\mu}^{n-1} + \frac{1}{n}W^n, \tag{2.4}$$

$$\hat{\sigma}^{2,n} = \begin{cases} \frac{1}{n}(W^n - \bar{\mu}^{n-1})^2 & n = 2, \\ \frac{n-2}{n-1}\hat{\sigma}^{2,n-1} + \frac{1}{n}(W^n - \bar{\mu}^{n-1})^2 & n > 2. \end{cases} \tag{2.5}$$

We will often speak of our belief state which captures what we know about the parameters we are trying to estimate. Given our observations, we would write our belief state as

$$B^{freq,n} = (\bar{\mu}^n, \hat{\sigma}^{2,n}, n).$$

Equations (2.4) and (2.5) describe how our belief state evolves over time.

The belief state is supposed to communicate a probability distribution as opposed to statistics such as mean and variance. When we are forming an average, we can apply the law of large numbers and assume that our estimate $\bar{\mu}^n$ is approximately normally distributed. This is true exactly if W is normally distributed, but it is generally a very good approximation even when W is described by other distributions.

2.3 THE BAYESIAN VIEW

The Bayesian perspective casts a different interpretation on the statistics we compute which is particularly useful in certain problem settings in optimal learning. While the frequentist perspective starts with no knowledge about a function or response (which is the case in many settings), the Bayesian perspective leverages prior knowledge that comes from past experience or an understanding of the dynamics of a problem.

For example, let x be the price of a product, and let μ_x be the sales when we offer the product at a price x. With frequentist learning, we start knowing nothing about μ_x, but as we collect data to construct estimates $\bar{\mu}_x^n$ of mu_x, we may find that $\bar{\mu}_x^n$ does not necessarily decline with x. In a Bayesian model, we would at least start with a prior $\bar{\mu}_x^0$ where this is the case.

The biggest difference between Bayesian and frequentist learning is the Bayesian prior, which is the initial distribution of belief about the truth. The prior is how we are able to communicate domain knowledge, if this is available. A prior is critical in the context of virtually any problem where experiments are really expensive (such as laboratory or field experiments), or when there is significant risk, as in health decisions.

The second difference between the two perspectives is the view of the truth. In the Bayesian view, the truth μ_x is handled as a random variable. For example, we might model μ_x as being normally distributed with mean $\bar{\mu}_x^0$ and variance $\bar{\sigma}_x^{2,0}$. As we collect information, this distribution changes, where we can guarantee that the variance will steadily shrink. In the frequentist perspective, the truth is an unknown number, and our estimate of this number is a random variable that reflects the variation in our observations. We understand, for example, that if 100 people all collected data to produce an estimate, each of these 100 estimates would be different, and could be described by some distribution (often normal).

The Bayesian perspective is well suited to information collection in settings where information is expensive (or where there is a significant risk involved if we make the wrong decision). It is in these settings where there is the incentive to bring in prior knowledge, but

these are also the settings where we are likely to have domain knowledge. By contrast, there are settings within search algorithms, or on the internet, where information is relatively much easier to obtain, and while this does not mean it is free, it is often easier to run a set of experiments to form initial estimates than to depend on a prior from an exogenous source.

We note a subtle change in notation from the frequentist perspective, where $\bar{\mu}^n$ was our statistic giving our estimate of μ. In the Bayesian view, we let $\bar{\mu}^n$ be our estimate of the mean of the random variable μ after we have made n observations. It is important to remember that μ is a random variable whose distribution reflects our prior belief about μ. The parameter $\bar{\mu}^0$ is not a random variable. This is our initial estimate of the mean of our prior distribution. By contrast, $\bar{\mu}^n$, for $n \geq 1$, is a random variable for the same reason that $\bar{\mu}^n$ is random in the frequentist view: $\bar{\mu}^n$ is computed from a series of random observations $W^1, W^2, \ldots, W^n$, and therefore the distribution of $\bar{\mu}^n$ reflects the distribution of all of our experiments. However, in the Bayesian perspective we are primarily interested in the mean and variance of μ.

Below we first use some simple expressions from probability to illustrate the effect of collecting information. We then give the Bayesian version of (2.4) and (2.5) for the case of independent beliefs, where observations of one choice do not influence our beliefs about other choices. We follow this discussion by giving the updating equations for correlated beliefs, where an observation of μ_x for alternative x tells us something about $\mu_{x'}$. We round out our presentation by touching on other important types of distributions.

2.3.1 The updating equations for independent beliefs

We begin by assuming (as we do through most of our presentation) that our random variable W is normally distributed. Let σ_W^2 be the variance of W, which captures the noise in our ability to observe the true value. To simplify the algebra, we define the *precision* of W as

$$\beta^W = \frac{1}{\sigma_W^2}.$$

Precision has an intuitive meaning: smaller variance means that the observations will be closer to the unknown mean, that is, they will be more precise. Now let $\bar{\mu}^n$ be our estimate of the true mean μ after n observations, and let β^n be the precision of this estimate. That is, having already seen the values $W^1, W^2, ..., W^n$, we believe that the mean of μ is $\bar{\mu}^n$, and the variance of μ is $\frac{1}{\beta^n}$. We say that we are "at time n" when this happens; note that all quantities that become known at time n are indexed by the superscript n, so the observation W^{n+1} is not known until time $n+1$. Higher precision means that we allow for less variation in the unknown quantity, that is, we are more sure that μ is equal to $\bar{\mu}^n$. After observing W^{n+1}, the updated mean and precision of our estimate of μ is given by

$$\bar{\mu}^{n+1} = \frac{\beta^n \bar{\mu}^n + \beta^W W^{n+1}}{\beta^n + \beta^W}, \tag{2.6}$$

$$\beta^{n+1} = \beta^n + \beta^W. \tag{2.7}$$

Equation (2.6) can be written more compactly as

$$\bar{\mu}^{n+1} = (\beta^{n+1})^{-1} \left(\beta^n \bar{\mu}^n + \beta^W W^{n+1}\right). \tag{2.8}$$

There is another way of expressing the updating which provides insight into the structure of the flow of information. First define

$$\tilde{\sigma}^{2,n} = \mathit{Var}^n[\bar{\mu}^{n+1}] \tag{2.9}$$
$$= \mathit{Var}^n[\bar{\mu}^{n+1} - \bar{\mu}^n] \tag{2.10}$$

where $\mathit{Var}^n[\cdot] = Var[\cdot \,|\, W^1, ..., W^n]$ denotes the variance of the argument given the information we have through n observations. For example,

$$\mathit{Var}^n[\bar{\mu}^n] = 0$$

since, given the information after n observations, $\bar{\mu}^n$ is a number that we can compute deterministically from the prior history of observations.

The parameter $\tilde{\sigma}^{2,n}$ can be described as the variance of $\bar{\mu}^{n+1}$ given the information we have collected through iteration n, which means the only random variable is W^{n+1}. Equivalently, $\tilde{\sigma}^{2,n}$ can be thought of as the *change* in the variance of $\bar{\mu}^n$ as a result of the observation of W^{n+1}. Equation (4.6) is an equivalent statement since, given the information collected up through iteration n, $\bar{\mu}^n$ is deterministic and is therefore a constant. We use equation (4.6) to offer the interpretation that $\tilde{\sigma}^{2,n}$ is the *change* in the variance of our estimate of the mean of μ.

It is possible to write $\tilde{\sigma}^{2,n}$ in different ways. For example, we can show that

$$\tilde{\sigma}^{2,n} = \bar{\sigma}^{2,n} - \bar{\sigma}^{2,n+1} \tag{2.11}$$
$$= \frac{(\bar{\sigma}^{2,n})}{1 + \sigma_W^2/\bar{\sigma}^{2,n}} \tag{2.12}$$
$$= (\beta^n)^{-1} - (\beta^n + \beta^W)^{-1}. \tag{2.13}$$

The proof of (2.11) is given in Section 2.8.1. Equations (2.12) and (2.13) come directly from (2.11) and (2.7), using either variances or precisions.

Just as we let $\mathit{Var}^n[\cdot]$ be the variance given what we know after n experiments, let $\mathbb{E}^n$ be the expectation given what we know after n experiments. That is, if $W^1, \ldots, W^n$ are the first n experiments, we can write

$$\mathbb{E}^n \bar{\mu}^{n+1} \equiv \mathbb{E}(\bar{\mu}^{n+1} | W^1, \ldots, W^n) = \bar{\mu}^n.$$

We note in passing that $\mathbb{E}\bar{\mu}^{n+1}$ refers to the expectation before we have made any experiments, which means $W^1, \ldots, W^n$ are all random, as is W^{n+1}. By contrast, when we compute $\mathbb{E}^n \bar{\mu}^{n+1}$, $W^1, \ldots, W^n$ are assumed fixed, and only W^{n+1} is random. By the same token, $\mathbb{E}^{n+1}\bar{\mu}^{n+1} = \bar{\mu}^{n+1}$, where $\bar{\mu}^{n+1}$ is some number which is fixed because we assume that we already know $W^1, \ldots, W^{n+1}$. It is important to realize that when we write an expectation, we have to be explicit about what is random (that is, what variable(s) we are averaging over), and what we are conditioning on.

Using this property, we can write $\bar{\mu}^{n+1}$ in a different way that brings out the role of $\tilde{\sigma}^n$. Assume that we have made n observations and let

$$Z = \frac{\bar{\mu}^{n+1} - \bar{\mu}^n}{\tilde{\sigma}^n}.$$

We note that Z is a random variable only because we have not yet observed W^{n+1}. Normally we would index $Z = Z^{n+1}$ since it is a random variable that depends on W^{n+1}, but we are going to leave this indexing implicit. It is easy to see that $\mathbb{E}^n Z = 0$ and

$Var^n Z = 1$. Also, since W^{n+1} is normally distributed, then Z is normally distributed, which means $Z \sim N(0,1)$. This means that we can write

$$\bar{\mu}^{n+1} = \bar{\mu}^n + \tilde{\sigma}^n Z. \tag{2.14}$$

Equation (2.14) makes it clear how $\bar{\mu}^n$ evolves over the observations. It also reinforces the idea that $\tilde{\sigma}^n$ is the change in the variance due to a single observation.

Equations (2.6) and (2.7) are the Bayesian counterparts of (2.4) and (2.5), although we have simplified the problem a bit by assuming that the variance of W is known. The belief state in the Bayesian view (with normally distributed beliefs) is given by

$$B^{Bayes,n} = (\bar{\mu}^n, \beta^n).$$

As we show below in Section 2.8.2, if our prior belief about μ is normally distributed with mean $\bar{\mu}^n$ and precision β^n, and if W is normally distributed, then our posterior belief after $n+1$ observations is also normally distributed with mean $\bar{\mu}^{n+1}$ and precision β^{n+1}. We often use the term *Gaussian prior*, when we want to say that our prior is normally distributed. We also allow ourselves a slight abuse of notation: we use $\mathcal{N}\left(\mu, \sigma^2\right)$ to mean a normal distribution with mean μ and variance σ^2, but we also use the notation $\mathcal{N}(\mu, \beta)$ when we are using the precision instead of the variance.

Needless to say, it is especially convenient if the prior distribution and the posterior distribution are of the same basic type. When this is the case, we say that the prior and posterior are *conjugate*, or that they are a *conjugate family*. This happens in a few special cases when the prior distribution and the distribution of W are chosen in a specific way.

The property that the posterior distribution is in the same family as the prior distribution is called *conjugacy*. The normal distribution is unusual in that the conjugate family is the same as the sampling family (the distribution of the experiment W is also normal). For this reason, this class of models is sometimes referred to as the "normal-normal" model (this phraseology becomes clearer below when we discuss other combinations).

In some cases, we may impose conjugacy as an approximation. For example, it might be the case that the prior distribution on μ is normal, but the distribution of the observation W is not normal (for example, it might be nonnegative). In this case, the posterior may not even have a convenient analytical form. But we might feel comfortable approximating the posterior as a normal distribution, in which case we would simply use (2.54)-(2.55) to update the mean and variance and then assume that the posterior distribution is normal.

2.3.2 Updating for correlated normal priors

A particularly important problem class in optimal learning involves problems where there are multiple choices, where our beliefs about the choices are correlated. Some examples of correlated beliefs are as follows:

- We are interested in finding the price of a product that maximizes total revenue. We believe that the function $R(p)$ that relates revenue to price is continuous. Assume that we set a price p^n and observe revenue R^{n+1} that is higher than we had expected. If we raise our estimate of the function $R(p)$ at the price p^n, our beliefs about the revenue at nearby prices should be higher.

- We choose five people for the starting lineup of our basketball team and observe total scoring for one period. We are trying to decide if this group of five people is better than another lineup that includes three from the same group with two different people.

If the scoring of these five people is higher than we had expected, we would probably raise our belief about the other group, since there are three people in common.

- A physician is trying to treat diabetes using a treatment of three drugs, where she observes the drop in blood sugar from a course of a particular treatment. If one treatment produces a better-than-expected response, this would also increase our belief of the response from other treatments that have one or two drugs in common.
- We are trying to find the highest concentration of a virus in the population. If the concentration of one group of people is higher than expected, our belief about other groups that are close (either geographically, or due to other relationships) would also be higher.

Correlated beliefs are a particularly powerful device in optimal learning, allowing us to generalize the results of a single observation to other alternatives that we have not directly measured.

Let $\bar{\mu}_x^n$ be our belief about alternative x after n experiments. Now let

$$Cov^n(\mu_x, \mu_y) \quad = \quad \text{the covariance in our belief about } \mu_x \text{ and } \mu_y.$$

We let Σ^n be the covariance matrix, with element $\Sigma^n_{xy} = Cov^n(\mu_x, \mu_y)$. Just as we defined the precision β_x^n to be the reciprocal of the variance, we are going to define the precision matrix B^n to be

$$B^n = (\Sigma^n)^{-1}.$$

Let e_x be a column vector of zeroes with a 1 for element x, and as before we let W^{n+1} be the (scalar) observation when we decide to measure alternative x. We could label W^{n+1} as W_x^{n+1} to make the dependence on the alternative more explicit. For this discussion, we are going to use the notation that we choose to measure x^n and the resulting observation is W^{n+1}. If we choose to measure x^n, we can also interpret the observation as a column vector given by $W^{n+1}e_{x^n}$. Keeping in mind that $\bar{\mu}^n$ is a column vector of our beliefs about the expectation of μ, the Bayesian equation for updating this vector in the presence of correlated beliefs is given by

$$\bar{\mu}^{n+1} \quad = \quad (B^{n+1})^{-1}\left(B^n\bar{\mu}^n + \beta^W W^{n+1}e_{x^n}\right), \tag{2.15}$$

where B^{n+1} is given by

$$B^{n+1} \quad = \quad (B^n + \beta^W e_{x^n}(e_{x^n})^T). \tag{2.16}$$

Note that $e_x(e_x)^T$ is a matrix of zeroes with a one in row x, column x, whereas β^W is a scalar giving the precision of our experimental outcome W.

It is possible to perform these updates without having to deal with the inverse of the covariance matrix. This is done using a result known as the Sherman-Morrison formula. If A is an invertible matrix (such as Σ^n) and u is a column vector (such as e_x), the Sherman-Morrison formula is

$$[A + uu^T]^{-1} = A^{-1} - \frac{A^{-1}uu^TA^{-1}}{1 + u^TA^{-1}u}. \tag{2.17}$$

Let $\lambda^W = \sigma_W^2 = 1/\beta^W$ be the variance of our experimental outcome W^{n+1}. We are going to simplify our notation by assuming that our experimental variance is the same across all

alternatives x, but if this is not the case, we can replace λ^W with λ_x^W throughout. Using the Sherman-Morrison formula, and letting $x = x^n$, we can rewrite the updating equations as

$$\bar{\mu}^{n+1}(x) = \bar{\mu}^n + \frac{W^{n+1} - \bar{\mu}_x^n}{\lambda^W + \Sigma_{xx}^n}\Sigma^n e_x, \tag{2.18}$$

$$\Sigma^{n+1}(x) = \Sigma^n - \frac{\Sigma^n e_x (e_x)^T \Sigma^n}{\lambda^W + \Sigma_{xx}^n}. \tag{2.19}$$

where we express the dependence of $\bar{\mu}^{n+1}(x)$ and $\Sigma^{n+1}(x)$ on the alternative x which we have chosen to measure.

To illustrate, assume that we have three alternatives with mean vector

$$\bar{\mu}^n = \begin{bmatrix} 20 \\ 16 \\ 22 \end{bmatrix}.$$

Assume that $\lambda^W = 9$ and that our covariance matrix Σ^n is given by

$$\Sigma^n = \begin{bmatrix} 12 & 6 & 3 \\ 6 & 7 & 4 \\ 3 & 4 & 15 \end{bmatrix}.$$

Assume that we choose to measure $x = 3$ and observe $W^{n+1} = W_3^{n+1} = 19$. Applying equation (2.18), we update the means of our beliefs using

$$\begin{aligned} \bar{\mu}^{n+1}(3) &= \begin{bmatrix} 20 \\ 16 \\ 22 \end{bmatrix} + \frac{19 - 22}{9 + 15} \begin{bmatrix} 12 & 6 & 3 \\ 6 & 7 & 4 \\ 3 & 4 & 15 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} 20 \\ 16 \\ 22 \end{bmatrix} + \frac{-3}{24} \begin{bmatrix} 3 \\ 4 \\ 15 \end{bmatrix} \\ &= \begin{bmatrix} 19.625 \\ 15.500 \\ 20.125 \end{bmatrix}. \end{aligned}$$

The update of the covariance matrix is computed using

$$
\begin{aligned}
\Sigma^{n+1}(3) &= \begin{bmatrix} 12 & 6 & 3 \\ 6 & 7 & 4 \\ 3 & 4 & 15 \end{bmatrix} - \frac{\begin{bmatrix} 12 & 6 & 3 \\ 6 & 7 & 4 \\ 3 & 4 & 15 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} [0 \ 0 \ 1] \begin{bmatrix} 12 & 6 & 3 \\ 6 & 7 & 4 \\ 3 & 4 & 15 \end{bmatrix}}{9+15} \\
&= \begin{bmatrix} 12 & 6 & 3 \\ 6 & 7 & 4 \\ 3 & 4 & 15 \end{bmatrix} - \frac{1}{24} \begin{bmatrix} 3 \\ 4 \\ 15 \end{bmatrix} [3 \ 4 \ 15] \\
&= \begin{bmatrix} 12 & 6 & 3 \\ 6 & 7 & 4 \\ 3 & 4 & 15 \end{bmatrix} - \frac{1}{24} \begin{bmatrix} 9 & 12 & 45 \\ 12 & 16 & 60 \\ 45 & 60 & 225 \end{bmatrix} \\
&= \begin{bmatrix} 12 & 6 & 3 \\ 6 & 7 & 4 \\ 3 & 4 & 15 \end{bmatrix} - \begin{bmatrix} 0.375 & 0.500 & 1.875 \\ 0.500 & 0.667 & 2.500 \\ 1.875 & 2.500 & 9.375 \end{bmatrix} \\
&= \begin{bmatrix} 11.625 & 5.500 & 1.125 \\ 5.500 & 6.333 & 1.500 \\ 1.125 & 1.500 & 5.625 \end{bmatrix}.
\end{aligned}
$$

These calculations are fairly easy, which means we can execute them even if we have thousands of alternatives. But we will run up against the limits of computer memory if the number of alternatives is in the 10^5 range or more, which arises when we consider problems where an alternative x is itself a multidimensional vector.

2.3.3 Bayesian updating with an uninformative prior

What if we truly have no prior information about a parameter before we start collecting information? We can use what is known in the Bayesian statistics literature as an uninformative prior, which is equivalent to a normal density with infinite variance (or zero precision). We note that it is not necessary for the prior to be a true density (which integrates to 1.0). For example, our prior on a random variable x can be $f(x) = .01$ for all $-\infty < x < \infty$, which of course integrates to infinity. This is known as an *improper prior*.

When we look at the Bayesian updating equations in (2.6) and (2.7), we see that if we use $\beta^0 = 0$, then it simply means that we put no weight on the initial estimates. It is easy to see that if $\beta^0 = 0$, then $\bar{\mu}^1 = W^1$ (the first observation) and $\beta^1 = \beta^W$ (the precision of our experiments).

The problem with uninformative priors is that we have no guidance at all regarding our first experiment x^0. Fortunately, in most applications of information collection we start with some prior knowledge, but this is not always the case.

Another strategy we can use if we have no prior information is known as *empirical Bayes*. Put simply, empirical Bayes requires that we collect a small initial sample, and then use the results of this sample to form a "prior" (obviously, this "prior" is formed *after* we calculate our small sample, but before we do any guided learning). Empirical Bayes sounds like a different name for frequentist (essentially our prior belief is created using a frequentist procedure), but the interpretation of what is random is different than with a frequentist model. The main distinguishing feature of the Bayesian approach is that it puts a number on the likelihood that the unknown value takes on a certain value or falls within a particular interval.

2.4 BAYESIAN UPDATING FOR SAMPLED NONLINEAR MODELS

There will be many instances where we need to work with models that are nonlinear in the parameter vector θ. Some examples are

---

EXAMPLE 2.1

Pricing a product on the internet - Imagine that we set a price x^n for day n and then observe total revenue Y_t. We might model the demand for the product using

$$D(x|\theta) = \theta_1 e^{-\theta_2(x-\theta_3)}.$$

The revenue would then be given by $f(x|\theta) = xD(x|\theta)$. We might set a price x^n for day x and then observe revenue Y^n. After N days, we have a dataset $(Y^n, x^n), n = 1, \ldots, N$.

EXAMPLE 2.2

Assume a physical has to choose from a set of treatments $\mathcal{X} = \{x_1, \ldots, x_M\}$ for reducing blood sugar. We are interested in estimating the probability that a treatment x will be judged a success.

We might begin by constructing a utility function that captures the attractiveness of the product as a function of price p, which we might write as

$$U(x|\theta) = \theta_0 + \sum_{m=1}^{M} \theta_m \mathbb{1}_{x=x_m}.$$

where

$$\mathbb{1}_{x=x_m} = \begin{cases} 1 & \text{If } x = x_m \\ 0 & \text{Otherwise.} \end{cases}$$

Now assume that we model the probability of the treatment x being effective is given by

$$P(x) = \frac{e^{U(x|\theta)}}{1 + e^{U(x|\theta)}}.$$

Normally the estimation of nonlinear models can become quite complex, but we are going to use a technique known as a *sampled belief model*. We assume that we are given a nonlinear function $f(x|\theta)$ where x is the input (in our examples this was either a price or medical treatment), and then we observe a response Y. We then assume that θ is one of the set $\Theta = \{\theta_1, \ldots, \theta_K\}$, initially with probability $P[\theta = \theta_k] = p_k^0$. We refer to $p^0 = (p_k^0), k = 1, \ldots, K$ as the prior.

The next step is to design the updating equations for the probability vector p^n after running an experiment with $x = x^n$ and observing a response y^{n+1}. We start with Bayes theorem

$$\mathbb{P}(A|B) = \frac{\mathbb{P}(B|A)\mathbb{P}(A)}{\mathbb{P}(B)}.$$

Now interpret the event A as the event that $\theta = \theta_k$, and B as the new information y^{n+1}. We are going to let H^n be the history of our experiments where

$$H^n = (S^0, x^0, \hat{y}^1, x^1, \hat{y}^2, \ldots, x^{n-1}, \hat{y}^n).$$

We can write our belief probabilities as

$$p_k^n = \mathbb{P}[\theta = \theta_k | H^n].$$

We note that we can write Bayes' theorem where all of the probabilities are conditioned on a third event C, as in

$$\mathbb{P}(A|B,C) = \frac{\mathbb{P}(B|A,C)\mathbb{P}(A|C)}{\mathbb{P}(B|C)}.$$

In our setting, the new event C is our history H^n. Adapting this to our setting gives us

$$\mathbb{P}[\theta = \theta_k | y^{n+1} = y, H^n] = \frac{\mathbb{P}[\hat{y}^{n+1} = y | \theta_k, H^n]\mathbb{P}[\theta = \theta_k | H^n]}{\mathbb{P}[\hat{y}^{n+1} = y | H^n]}. \tag{2.20}$$

Let $f^y(\hat{y} = y|\theta)$ be the distribution of the random observation $\hat{y}$ given θ, or

$$f^y(\hat{y} = y|\theta) = \mathbb{P}[\hat{y} = y|\theta].$$

The conditioning on the history H^n affects the probability that θ takes on a particular value. We can rewrite equation (2.20) as

$$p_k^{n+1} = \frac{f^y(\hat{y}^{n+1} = y|\theta_k)p_k^n}{f^y(\hat{y}^{n+1} = y)} \tag{2.21}$$

where

$$f^y(\hat{y}^{n+1} = y) = \sum_{k=1}^{K} f^y(\hat{y}^{n+1} = y|\theta_k)p_k^n. \tag{2.22}$$

We are exploiting the fact that the distribution of $\hat{y}^{n+1}$ depends only on θ, so we can drop the dependence on H^n when we are also conditioning on θ, since the history only affects the probabilities p_k^n.

We again note that we assume that $f^y(y^{n+1}|\theta_k)$ is a known distribution that can be easily computed from the structure of the problem. For example, it might be a normal density, a Poisson distribution, or a logistic regression.

Equation 2.21 is quite easy to compute (given $f^y(y^{n+1}|\theta_k)$) when we use a sampled belief model. Without the sampled belief model, the expectation in equation (2.22) can become quite complex (imagine what happens when θ is a vector). By contrast, this equation is easy to compute for the sampled belief model, even if θ is a high-dimensional vector.

The process is illustrated in figure 2.1 for a problem where we are trying to show the market response to the price using a logistics curve. The figure shows (a) the initial, uniform prior of a sampled belief model with four possible values of θ, (b) the possible outcomes from an experiment x, (c) the actual result of an experiment, and (d) the posterior distribution which are indicated by the thickness of the lines.

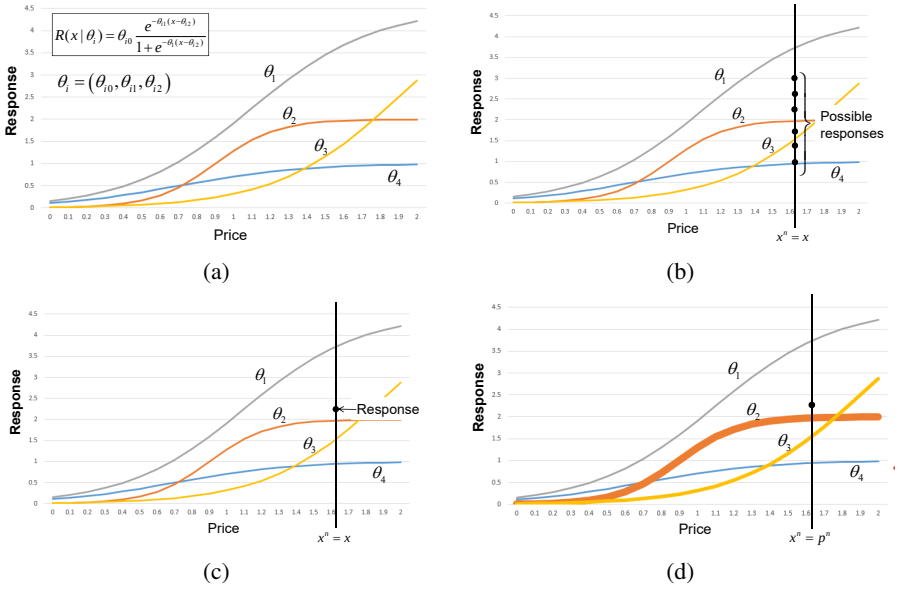


Figure 2.1 Illustration of Bayesian updating for a sampled belief model. Starting from top left: a) initial uniform prior, b) set of potential outcomes from an experiment x, c) the response $\hat{y}^{n+1}$, d) the posterior (width proportional to probability).

2.5 THE EXPECTED VALUE OF INFORMATION

The previous section described how we update the mean and variance (or precision) for a particular observation of W. It is useful to see what happens when we look at the *expected* change in the mean and variance when we average over all possible realizations.

Let μ be the unknown true value of some quantity. Then, μ is a random variable with (assumed) mean $\bar{\mu}^0$ and variance $\bar{\sigma}^{2,0}$. Let W be a random observation of whatever we are measuring, from which we might update our estimate of the mean and variance. Note that

$$\mathbb{E}W = \mathbb{E}_\mu \mathbb{E}_{W|\mu} (W \mid \mu) = \mathbb{E}\mu = \bar{\mu}^0.$$

We observe that

$$\begin{aligned} \mathbb{E}\mu &= \bar{\mu}^0, \\ Var(\mu) &= \bar{\sigma}^{2,0}. \end{aligned}$$

Assume that we observe $W = w$. For example, we might assume that the travel time on a link has mean $\bar{\mu}^0 = 22$ minutes, but we observe an actual travel time of $W = w = 27$ minutes. Then, $\mathbb{E}(\mu|W = w)$ would be the updated mean, and $Var(\mu|W = w)$ would be the updated variance.

Now let's consider what happens *on average* to our estimates of the mean and variance when we consider all possible outcomes of our observation of W. Let $\mathbb{E}_\mu \mu$ be the

expected value of μ (over our density function for μ), which is equal to $\bar{\mu}^0$. Now let $\mathbb{E}_\mu(\mu|W)$ be the expected value of μ given a particular observation of W. The expectation $\mathbb{E}_\mu(\mu|W)$ is a random variable (since it depends on W), but we are interested in its expectation over all possible values of W, which we can write as $\mathbb{E}_W\mathbb{E}_\mu(\mu|W)$. We can compute this by taking expectations of equation (2.8) given what we know at iteration n (which means that only W^{n+1} is random). We start by observing that

$$\mathbb{E}_\mu(\mu|W) = \bar{\mu}^1 = (\beta^1)^{-1}\left(\beta^0\bar{\mu}^n + \beta^W W\right),$$

where $\beta^1 = \beta^0 + \beta^W$. We then take expectations of both sides over the random observation W to obtain

$$\begin{aligned}
\mathbb{E}_W\bar{\mu}^1 &= \mathbb{E}_W\left((\beta^1)^{-1}\left(\beta^0\bar{\mu}^0 + \beta^W W\right)\right) \\
&= (\beta^1)^{-1}\left(\beta^0\bar{\mu}^0 + \beta^W\mathbb{E}W\right) \\
&= (\beta^1)^{-1}\left(\beta^0\bar{\mu}^0 + \beta^W\bar{\mu}^0\right) \\
&= (\beta^1)^{-1}\left(\beta^0 + \beta^W\right)\bar{\mu}^0 \\
&= \bar{\mu}^0.
\end{aligned}$$

This result seems to be saying that collecting an observation of W does not change our belief of the true mean μ. This is not the case. As we saw in the previous section, a particular observation of W will, in fact, change our belief about the mean of μ. But if we look at all possible realizations of W, before the observation occurs, *on average* our estimate of the mean does not change.

This simple equation provides an insight into priors and learning. Imagine if

$$\mathbb{E}\mu = \mathbb{E}_W\mathbb{E}_\mu(\mu|W) = \bar{\mu}^0 + a.$$

That is, observing W will, on average, shift our belief about the true mean from $\bar{\mu}^0$ to $\bar{\mu}^0 + a$, where a is a constant which would have to be known before we run our experiment. If this were true (for a other than zero), then this would mean that our initial estimate $\bar{\mu}^0$ is not a true prior. That is, we could shift our prior by a so that it becomes an unbiased estimate of the mean.

Now consider what happens to the variance after an experiment. We use some basic relationships from probability to obtain

$$\begin{aligned}
\mathit{Var}(\bar{\mu}) &= \mathbb{E}_\mu(\bar{\mu}^2) - (\mathbb{E}_\mu(\mu))^2 && (2.23)\\
&= \mathbb{E}_\mu(\bar{\mu}^2) - (\mathbb{E}_\mu(\mu))^2 + (\mathbb{E}_\mu(\mu))^2 - (\mathbb{E}_\mu(\mu))^2 && (2.24)\\
&= \mathbb{E}_W[(\mathbb{E}_\mu(\bar{\mu}^2|W))] - \mathbb{E}_W[(\mathbb{E}_\mu(\mu|W))^2] + \mathbb{E}_W[(\mathbb{E}_\mu(\mu|W))^2] - (\mathbb{E}_W[\mathbb{E}_\mu(\mu|W)])^2 && (2.25)\\
&= \mathbb{E}_W[(\mathbb{E}_\mu(\bar{\mu}^2|W)) - (\mathbb{E}_\mu(\mu|W))^2] + \mathbb{E}_W[(\mathbb{E}_\mu(\mu|W))^2] - (\mathbb{E}_W[\mathbb{E}_\mu(\mu|W)])^2 && (2.26)\\
&= \mathbb{E}_W[\mathit{Var}(\mu|W)] + \mathit{Var}[\mathbb{E}_\mu(\mu|W)]. && (2.27)
\end{aligned}$$

Equation (2.23) is the definition of the variance. In (2.24), we add and subtract $(\mathbb{E}_\mu(\mu|W))^2$. In (2.25), we then condition throughout on W and then take the expectation over W. In (2.26), we pull the $\mathbb{E}_W$ out front of the first two terms, setting up the final (and classic) result given by equation (2.28). Our interest is primarily in

equation (2.26). Above, we pointed out that $\mathbb{E}_W \mathbb{E}_\mu(\mu|W) = \mathbb{E}_\mu \mu$. This is not the case with the variance, where equation (2.28) tells us that

$$\mathbb{E}_W[\mathit{Var}(\mu|W)] = \mathit{Var}(\mu) - \mathit{Var}[\mathbb{E}_\mu(\mu|W)]. \tag{2.28}$$

This means that the variance after an experiment will, on average, always be smaller than the original variance. Of course, it might be the case that $\mathit{Var}[\mathbb{E}_\mu(\mu|W)] = 0$. This would happen if W were an irrelevant piece of information. For example, assume that μ is our estimate of the travel time on a path in a network, and W is an observation of the change in the S&P stock index yesterday. The S&P stock index does not tell us anything about the travel time on the path, which means that $\mathbb{E}_\mu(\mu|W)$ is a constant (in our example, it would be $\bar{\mu}^0$). Clearly, $\mathit{Var}(\bar{\mu}^0) = 0$, since $\bar{\mu}^0$ is not a random variable (it is just a number).

We collect observations one at a time. So, the above discussion continues to apply after we observe $W^1, W^2, ..., W^n$. Since the posterior mean $\bar{\mu}^n$ is a known, fixed quantity at time n, we can simply view it as a new prior. Our problem essentially restarts after each observation, just with different parameters for our distribution of belief. The advantage of recursive updating is that it allows us to turn a problem with a long-time horizon into a sequence of small problems – a concept that will also inform our solution techniques in later chapters.

2.6 UPDATING FOR NON-GAUSSIAN PRIORS

So far, we have considered a setting where the random observations W^n are assumed to come from a normal distribution, whose mean is the true value that we wish to estimate. We have also used a normal distribution to describe our prior belief about the unknown value. These are two very different normal distributions, and it is important to distinguish between them. We use the term "sampling distribution" to refer to the distribution of the observations W^n. This distribution depends on certain unknown parameters, like the value μ in the preceding section. Although we do not know this distribution, we have a way of obtaining samples from it.

By contrast, the term "prior distribution" describes the distribution of our own beliefs about the unknown parameters of the sampling distribution. Unlike the sampling distribution, which is determined by nature and unknown to us, the prior distribution is something that we construct to encode our own uncertainty about the truth. The prior distribution changes as we accumulate observations, reflecting the changes in our beliefs and the reduction in our uncertainty that result from collecting new information.

In the preceding section, both the sampling distribution and the prior distribution are assumed to be normal. This model is particularly intuitive and versatile because the unknown parameter in the sampling distribution is precisely the mean of the sampled observations. The mean $\bar{\mu}^0$ of the prior distribution can easily be interpreted as our "best guess" as to the unknown value, with σ^0 representing our uncertainty. Thus, whenever we have a rough idea of what the unknown mean might be, a normal distribution provides a very intuitive and understandable way to encode that idea in a mathematical model.

Unfortunately, the normal-normal model (normal prior, normal samples) is not always suitable. For one thing, a normal sampling distribution assumes that our random observations can take on any real value. In reality, this might not be the case: for instance, we might be observing the waiting times of customers in a service center, where we are trying to estimate the service rate. Waiting times are always positive, so an exponential distribution would seem to be a more appropriate choice than a normal distribution. Even more troubling is a situation where our observations are obviously discrete. For instance, we might be observing the success or failure of a medical test, where the outcome is 0 or 1. A normal distribution is certainly a poor choice for a sampling model in this setting.

The normal distribution is not always the best choice for a prior, either. For example, if we are observing exponentially distributed service times, the service rate is necessarily a strictly positive number. If we were to put a normal prior on the service rate, then even with a positive prior mean, we would essentially be allowing the possibility that our exponential sampling distribution has a negative rate, which is impossible. Our uncertainty about the service rate should be encoded using a different kind of distribution that accounts for the fact that the rate must be positive. In the case of 0/1 observations (success or failure of a test), the sample mean is the probability of success. We know that this number must be between 0 and 1, so a normal distribution is again not the best choice to represent our beliefs.

In this section, we discuss several other possible learning models where the sampling and prior distributions are not normal. However, all the distributions that we consider will retain the conjugacy property, which means that the posterior distribution is of the same type as the prior distribution.

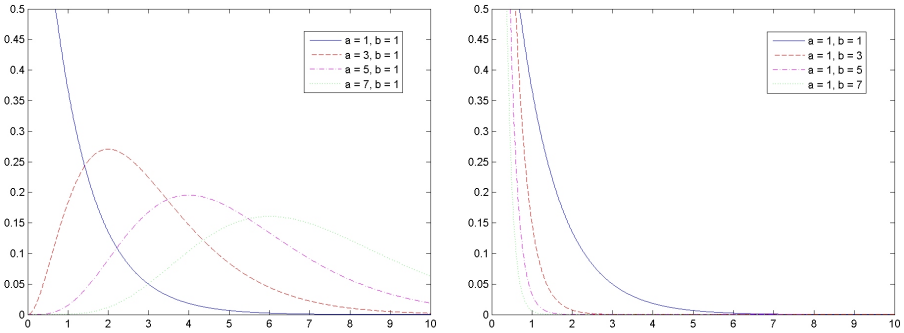
2.6.1 The gamma-exponential model

The gamma-exponential model is one possible choice for a situation where the observations are continuous and positive. Suppose that we are trying to estimate the service time distribution at a car repair shop by combining our prior belief with observations of actual service times. We feel comfortable assuming that the sampling distribution governing the service times is exponential with parameter λ. The service rate is the unknown value that we wish to estimate.

Since λ is itself unknown, we view it as a random variable. Clearly it is not appropriate to assume that it follows a normal distribution, since this would mean that we believe that λ might be negative. Assume instead that λ comes from a gamma distribution with parameters a and b. This distribution is given by

$$f(x|a,b) = b(bx)^{a-1}\frac{e^{-bx}}{\Gamma(a)}$$

where a is typically an integer (as it will be for our applications) and $\Gamma(a) = (a-1)!$. Figure 2.2 illustrates several examples of the gamma density. If $a = 1$, the gamma is an exponential distribution. For $a > 1$, it takes on a skewed shape which approaches a normal distribution for larger values of a.



2.2(a) Varying a 2.2(b) Varying b

Figure 2.2 Illustration of a family of gamma distributions a) varying a and b) varying b.

The mean of this distribution is given by

$$\mathbb{E}(\lambda) = \frac{a^0}{b^0}. \tag{2.29}$$

The quantities a^0 and b^0 should be chosen by us in such a way so that (2.29) represents our initial beliefs about the service rate.

We can let a^n/b^n be our estimate of λ after n observations, as before. After observing W^{n+1}, we update our beliefs using the equations

$$a^{n+1} = a^n + 1, \tag{2.30}$$

$$b^{n+1} = b^n + W^{n+1}. \tag{2.31}$$

Our belief about λ is that it follows a gamma distribution with parameters a^{n+1} and b^{n+1}. Despite the complexity of the gamma density, the updating equations governing the way in which we learn about λ are actually quite simple.

Equations (2.30) and (2.31) give a simple explanation of the gamma prior that makes the parameters a^n and b^n seem a bit less mysterious. Essentially, b^n is the sum of the first n service times (plus some prior constant b^0), whereas a^n is roughly equal to n (again, plus a prior constant). Thus, after n observations, our estimate of the service rate is given by

$$\mathbb{E}\left(\lambda \,|\, W^1, ..., W^n\right) = \frac{a^n}{b^n}.$$

This estimate is roughly the number of customers that were served per single unit of time. This is precisely the meaning of a service rate.

While the gamma-exponential model (gamma prior, exponential sampling distribution) is useful for modeling problems with continuous, positive observations, it is incapable of handling correlated beliefs. There is no easy multivariate analog for the gamma distribution, the way there is with the normal distribution, and thus no analog of the correlated normal updating equations (2.18)-(2.19). In a setting where there are multiple unknown values with heavy correlations, it is important to consider the trade-off between using a multivariate normal model to capture the correlations, and using a different type of model to more accurately represent the individual distributions of the alternatives.

2.6.2 The gamma-Poisson model

The gamma-Poisson model is similar to the gamma-exponential model, but the observations are now assumed to be discrete. For example, we may now be interested in the arrival rate of customers to the car repair shop, rather than the service time. Suppose that the total number of customers N that visit the shop in a single day follows a Poisson distribution with rate λ customers per day. Our observations are now the actual numbers of customers that arrive on different days. If the arrival rate is λ, the distribution of N follows the Poisson distribution given by

$$\mathbb{P}[N = x] = \frac{\lambda^x e^{-\lambda}}{x!},$$

where $x = 0, 1, \ldots$. The problem is that we do not know λ, and we wish to estimate it from observations N^n where N^n is the observed number of arrivals on the n^{th} day.

Once again, we assume that λ comes from a gamma distribution with parameters a^0 and b^0. The prior distribution changes after each observation according to the equations

$$a^{n+1} = a^n + N^{n+1}, \tag{2.32}$$
$$b^{n+1} = b^n + 1. \tag{2.33}$$

After n observations, our estimate of the Poisson rate,

$$\mathbb{E}\left(\lambda \,|\, W^1, ..., W^n\right) = \frac{a^n}{b^n},$$

is roughly equal to the average number of customers that arrived per day. This is in line with the meaning of the Poisson rate.

The gamma-Poisson case highlights the distinction between the sampling distribution and the prior. While the individual Poisson observations are discrete, the Poisson rate itself can be any positive real number, and thus can be modeled using the gamma distribution.

2.6.3 The Pareto-uniform model

Suppose that W is uniform on the interval $[0, B]$, where B is unknown. Our problem is thus to estimate the maximum of a uniform distribution. This problem is the continuous version of a production estimation problem, in which we can observe a sequence of serial numbers, and the goal is to guess the highest serial number produced. We can also use this model to estimate the maximum possible demand for a product or other extreme values.

We assume that B comes from a Pareto distribution with parameters $b > 0$ and $\alpha > 1$. The density of this distribution is given by

$$f\left(x|\alpha, b\right) = \begin{cases} \frac{\alpha b^\alpha}{x^{\alpha+1}} & \text{if } x > b \\ 0 & \text{otherwise.} \end{cases}$$

Thus, our prior estimate of B using priors $\alpha = \alpha^0$ and $b = b^0$ is given by

$$\mathbb{E}(B) = \frac{\alpha^0 b^0}{\alpha^0 - 1}.$$

The parameter b^0 estimates the $\frac{\alpha^0-1}{\alpha^0}$-quantile of the uniform distribution, and α^0 gives us the multiplier used to obtain the estimate of the maximum.

Although this model looks somewhat peculiar, it also has the conjugacy property. Our beliefs continue to have a Pareto distribution as we make observations, and the parameters of the distribution evolve according to the equations

$$b^{n+1} = \max\left(b^n, W^{n+1}\right), \tag{2.34}$$

$$\alpha^{n+1} = \alpha^n + 1. \tag{2.35}$$

Thus, b^n is roughly the maximum of the first n uniform observations. Our beliefs tell us that the true maximum of the distribution must be larger than b^n. However, if we have made many observations, it is likely that b^n is fairly close to the maximum. The degree of this "closeness" is represented by α^n.

2.6.4 Models for learning probabilities*

In many problems, our objective is to learn the probability that a certain event will occur, rather than the economic value of the event. For example, in a medical setting, our observations might simply be whether or not a certain medical treatment is successful. Such an observation can be modeled as a Bernoulli random variable, which is equal to 1 (success) with probability ρ, and 0 (failure) with probability $1-\rho$. The success probability ρ is the unknown true value in this case.

We assume that ρ comes from a beta distribution with parameters α and β. Recall that the beta density is given by

$$f(x|\alpha,\beta) = \begin{cases} \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1} & \text{if } 0 < x < 1 \\ 0 & \text{otherwise.} \end{cases}$$

As before, $\Gamma(y) = y!$ when y is integer. In this setting, α and β are always integer. Figure 2.3 illustrates the beta distribution for different values of α and β.

Our prior estimate of ρ using $\alpha = \alpha^0$ and $\beta = \beta^0$ is given by

$$\mathbb{E}(\rho) = \frac{\alpha^0}{\alpha^0 + \beta^0}. \tag{2.36}$$

Thus, α^0 and β^0 are weights that, when normalized, give us the probabilities of success and failure, respectively. If α^0 is large relative to β^0, this means that we believe success to be more likely than failure.

A common trait of all the learning models we have discussed thus far is that, while the prior or sampling distributions can have fairly complicated densities, the resulting updating equations are simple and often have an intuitive meaning. This is also the

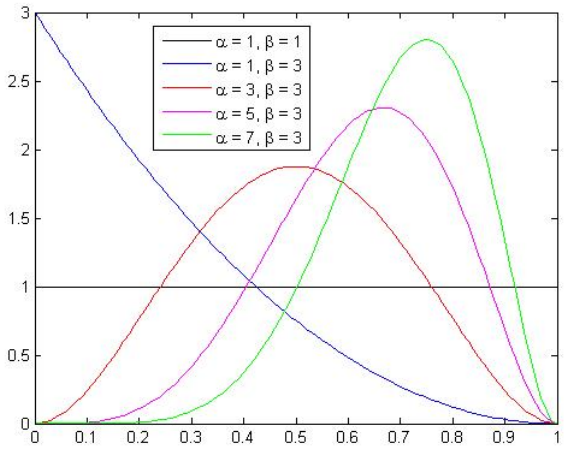


Figure 2.3 Illustration of a family of beta distributions.

case for the beta-Bernoulli model. The conjugacy property holds, and the parameters evolve according to the equations

$$\alpha^{n+1} = \alpha^n + W^{n+1}, \tag{2.37}$$
$$\beta^{n+1} = \beta^n + \left(1 - W^{n+1}\right), \tag{2.38}$$

where the observations W^n are 1 or 0, indicating a success or a failure. We see that the parameters α^n and β^n roughly keep track of the number of successes and failures in n observations. For instance, the parameter α^n is the number of successes in n observations, plus a prior constant α^0. This prior constant serves as a scaling factor of sorts. To see the importance of this scaling, consider the following three scenarios:

$$\begin{array}{ll} \text{Scenario 1:} & \alpha^0 = \beta^0 = 0.1 \\ \text{Scenario 2:} & \alpha^0 = \beta^0 = 1 \\ \text{Scenario 3:} & \alpha^0 = \beta^0 = 10 \end{array}$$

In each scenario, our estimate of ρ is $1/2$, because the prior constants α^0 and β^0 are equal. However, suppose now that $W^1 = 1$ in all three cases, and consider how our beliefs change:

$$\begin{array}{lll} \text{Scenario 1:} & \alpha^1 = 1.1, \beta^1 = 0.1, & \mathbb{E}\left(\rho \,|\, W^1\right) = 0.916 \\ \text{Scenario 2:} & \alpha^1 = 2, \beta^1 = 1, & \mathbb{E}\left(\rho \,|\, W^1\right) = 0.666 \\ \text{Scenario 3:} & \alpha^0 = 11, \beta^0 = 10, & \mathbb{E}\left(\rho \,|\, W^1\right) = 0.524 \end{array}$$

In the first scenario, observing a single success leads us to greatly increase our estimate of the success probability. In the other two scenarios, we also increase our estimate of ρ, but by a much smaller amount. Thus, the prior values of α^0 and β^0 can be viewed as a measure of our confidence in our estimate of ρ. High values of α^0 and β^0 show that we are very confident in our prior estimate, and hence this estimate is not likely to change by very much. Low values of α^0 and β^0 show that we have

little prior knowledge about ρ, and our prior estimate can easily be changed by only a few observations.

The beta-Bernoulli model can be easily generalized to a multivariate setting. Suppose that, instead of a simple 0/1 value, each observation can be classified as belonging to one of K different categories. For example, instead of merely measuring the success or failure of a medical treatment, we also consider cases where the treatment is generally effective, but causes certain side effects. This result should be viewed differently from either total failure or unqualified success. Consequently, we now have more than two possible outcomes.

We model our observations as individual trials from a multinomial distribution with K categories. The probability that an observation belongs to category $k = 1, ..., K$ is $P(W^n = k) = \rho_k$, with each $\rho_k \in [0, 1]$ and $\sum_{k=1}^{K} \rho_k = 1$. The unknown true values are now the probabilities ρ_k. Let us use $\rho = (\rho_1, ..., \rho_K)$ to denote the vector containing all these probabilities.

Our prior is the multivariate generalization of the beta distribution, called the *Dirichlet distribution*. This distribution has a vector of parameters $\alpha = (\alpha_1, ..., \alpha_K)$, with one parameter for each category, satisfying $\alpha_k \geq 0$ for all k. The Dirichlet density is given by

$$f(x) = \begin{cases} \frac{\Gamma(\alpha_1 + ... + \alpha_K)}{\prod_{k=1}^{K} \Gamma(\alpha_k)} \prod_{k=1}^{K} x_k^{\alpha_k - 1} & \text{if } x_k \geq 0 \text{ for all } k \text{ and } \sum_{k=1}^{K} x_k = 1 \\ 0 & \text{otherwise.} \end{cases} \tag{2.39}$$

Essentially, the Dirichlet density is a probability distribution for the probability that an observation belongs in a particular category. The density can only be non-zero on the set of points x (the probabilities) in a k-dimensional space such that every component of x is positive, and the sum of the components is 1 (since the sum of the probabilities of being in each category has to sum to 1). For example, in two dimensions, this set is the part of the line $x_1 + x_2 = 1$ that lies in the non-negative quadrant.

By computing the marginal densities of (2.39), it can be shown that our estimate of the probability of observing an outcome in category k given a prior $\alpha = \alpha^0$ is

$$\mathbb{E}(\rho_k) = \frac{\alpha_k^0}{\alpha_1^0 + ... + \alpha_K^0},$$

a straightforward generalization of (2.36). Just as with the beta-Bernoulli model, the prior values α_k^0 represent the weight that we want to assign to the kth category. Large values of α_k^0 relative to $\alpha_{k'}^0$ indicate that we are more likely to observe category k than category k'. Our earlier discussion of scaling applies here as well.

To update our beliefs, we apply the equation

$$\alpha_k^{n+1} = \begin{cases} \alpha_k^n + 1 & \text{if } W^{n+1} \text{ belongs to category } k \\ \alpha^n & \text{otherwise.} \end{cases} \tag{2.40}$$

We can write this more concisely if we model W as taking values of the form e_k, where e_k is a K-vector of zeroes, with only the kth component equal to 1. Then, the probability mass function of W is given by $P(W = e_k) = \rho_k$, and (2.40) can be rewritten as

$$\alpha^{n+1} = \alpha^n + W^{n+1}. \tag{2.41}$$

It is important to remember that (2.41) is a vector equation, where only one component of W^{n+1} is equal to 1, and the other components are equal to zero. Simply put, if observation $n+1$ belongs to category k, we increment α_k^n by 1 and leave the other components of α^n unchanged.

2.6.5 Learning an unknown variance*

Our last learning model takes us back to the basic setting of one-dimensional Gaussian priors from Section 2.3.1. As before, we assume that the observation $W \sim \mathcal{N}\left(\mu, \beta^W\right)$, where $\beta^W = 1/\sigma_W^2$ is the precision. However, we will now suppose that both the true mean μ and the precision β^W are unknown. We will have to learn both of these quantities at the same time.

It is easy to imagine applications where β^W is unknown. In fact, the precision of an observation is often more difficult to estimate than the mean. For example, in finance, the return of a stock can be directly observed from market data, but the volatility has to be indirectly inferred from the returns. We often assume that β^W is known because it makes our model cleaner, but even then, in practice the value that we plug in for this quantity will be some sort of statistical estimate.

Because the mean and precision are both estimated using the same data, our beliefs about these two quantities are correlated. We create a joint prior distribution on $\left(\mu, \beta^W\right)$ in the following way. First, the marginal distribution of β^W is $Gamma\left(a, b\right)$, where $a, b > 0$ are prior parameters of our choosing. Next, given that $\beta^W = r$, the conditional distribution of μ is $\mathcal{N}\left(\theta, \tau r\right)$, where $-\infty < \theta < \infty$ and $\tau > 0$ are also prior parameters. Note that τr denotes the conditional precision of μ, not the conditional variance. We can write the joint density of $\left(\mu, \beta^W\right)$ as

$$f\left(x, r \mid \theta, \tau, a, b\right) = \frac{1}{\sqrt{2\pi\tau^{-1}r^{-1}}} \frac{b\left(br\right)^{a-1} e^{-br}}{\Gamma\left(a\right)} e^{-\frac{(x-\theta)^2}{2\tau^{-1}r^{-1}}}.$$

This is widely known as a "normal-gamma" distribution. It is closely related to Student's t-distribution (often simply called the t-distribution), used in statistics to estimate the mean of a sample under unknown variance. In fact, if $\left(\mu, \beta^W\right)$ is normal-gamma with parameters θ, τ, a and b, the marginal distribution of μ can be connected back to the t distribution. The random variable $\sqrt{\frac{\tau a}{b}}\left(\mu - \theta\right)$ follows the standard t distribution with $2a$ degrees of freedom, analogous to expressing a Gaussian random variable in terms of the standard Gaussian distribution.

The estimates of the unknown quantities that we obtain from the normal-gamma distribution are given by

$$\mathbb{E}\mu = \theta, \qquad \mathbb{E}\beta^W = \frac{a}{b}.$$

The parameter τ only affects the amount of uncertainty in our beliefs, with

$$Var\left(\mu\right) = \frac{b}{\tau\left(a-1\right)}, \qquad Var\left(\beta^W\right) = \frac{a}{b^2}.$$

Recall that τ affects the precision, so lower τ leads to more uncertainty and higher prior variance.

Like the other distributions considered in this chapter, the normal-gamma prior is conjugate when combined with normal observations. Suppose that $(\bar{\mu}^n, \tau^n, a^n, b^n)$ represent our beliefs after n observations, and we make an observation $W^{n+1} \sim \mathcal{N}\left(\mu, \beta^W\right)$. Then, the posterior distribution of $\left(\mu, \beta^W\right)$ is normal-gamma with parameters

$$\bar{\mu}^{n+1} = \frac{\tau^n \bar{\mu}^n + W^{n+1}}{\tau^n + 1}, \tag{2.42}$$

$$\tau^{n+1} = \tau^n + 1, \tag{2.43}$$

$$a^{n+1} = a^n + \frac{1}{2}, \tag{2.44}$$

$$b^{n+1} = b^n + \frac{\tau^n \left(W^{n+1} - \bar{\mu}^n\right)^2}{2\left(\tau^n + 1\right)}. \tag{2.45}$$

The equations are more complicated than their analogs in Section 2.3.1. However, (2.42) is actually a straightforward generalization of (2.6), replacing the precisions β^n and β^W from the known-variance model by scale factors, τ^n for the prior precision and 1 for the observation. In this way, we can see that the parameter τ^n is roughly equal to n, plus a prior constant.

Later on, we will use the normal-gamma model in a setting where, instead of collecting one observation at a time, we can obtain a batch of k observations simultaneously. In this case, we can easily update our beliefs by calculating (2.42)-(2.45) k times for the individual observations $W^{n+1}, ..., W^{n+k}$. It is instructive, however, to look at the equivalent "batch version" of the updating equations. Let $\bar{W}^{n,k} = \frac{1}{k}\sum_{i=1}^{k} W^{n+i}$ be the average of our k observations. Then, the posterior distribution of $\left(\mu, \beta^W\right)$ is normal-gamma with parameters

$$\bar{\mu}^{n+k} = \frac{\tau^n \bar{\mu}^n + k\bar{W}^{n,k}}{\tau^n + k}, \tag{2.46}$$

$$\tau^{n+k} = \tau^n + k, \tag{2.47}$$

$$a^{n+k} = a^n + \frac{k}{2}, \tag{2.48}$$

$$b^{n+k} = b^n + \frac{1}{2}\sum_{i=1}^{k}\left(W^{n+i} - \bar{W}^{n,k}\right)^2 + \frac{\tau^n k\left(\bar{W}^{n,k} - \bar{\mu}^n\right)^2}{2\left(\tau^n + k\right)}. \tag{2.49}$$

The differences between (2.42)-(2.45) and (2.46)-(2.49) are mostly straightforward. For example, in (2.46), the scale factor of k observations is simply k times the scale factor of a single observation. Notice, however, that (2.49) now involves a sum of squared errors $\left(W^{n+i} - \bar{W}^{n,k}\right)^2$ for $i = 1, ..., k$.

This sum of squares will be automatically computed if we apply (2.45) k times in a row, much like the recursive expression in (2.5) computes the sum of squares in (2.3). In this way, we see that the frequentist and Bayesian models are estimating the variance in roughly the same way. The Bayesian model simply adds a correction to the frequentist estimate in the form of the last term in (2.45), which uses the prior information $\bar{\mu}^n$ and τ^n. Larger values of τ^n correspond to a more precise prior and will place more weight on this term.

The Bayesian model allows us to learn the unknown variance from a single observation, whereas the frequentist model requires at least two. Essentially, the prior

stands in for the missing "first" observation. The difference is mostly cosmetic: in practice, the prior is frequently constructed using old or preliminary observations. The true difference between the frequentist and Bayesian philosophies is not in the updating equations, but in the philosophical interpretation of μ as an "unknown, but fixed number" (frequentist) or a "random variable" (Bayesian).

2.6.6 The normal as an approximation

The list of different priors and sampling distributions can make this entire problem of deriving posteriors seem quite complicated. Perhaps it is not surprising that we keep coming back to the normal distribution, drawing on the power of the central limit theorem.

Consider the setting of counting successes and failures. Let $\bar{p}^n$ be our probability of a success, and let W^{n+1} be our binary outcome capturing whether the experiment was a success ($W^{n+1} = 1$) or a failure ($W^{n+1} = 0$). The estimated probability $\bar{p}^n$ is a random variable with a beta distribution, but for n large enough (and this does not have to be very large), it is also accurately approximated by a normal distribution. If we then observe a binary W^{n+1}, the posterior distribution will not be normal, but it will be approximately normal.

This is a reason why the normal distribution is so widely used. With the notable exception of our sampled beliefs for nonlinear models, the normal distribution will be our most common reference distribution in this book.

2.7 MONTE CARLO SIMULATION

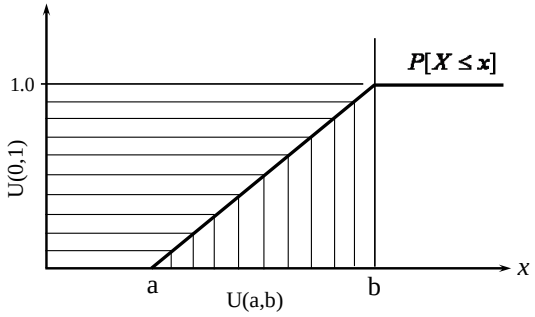
There are many settings where we need to take a sample realization of a random variable, a process known widely as Monte Carlo simulation. There are a number of good books on this subject. This section provides only a brief introduction to some elementary methods for generating samples of a random variable.

All computer languages include a utility for generating a random number that is uniformly distributed between 0 and 1. For example, in Microsoft Excel, this function is called "RAND()", and we can generate a random number from this function by entering "=RAND()" in a cell.

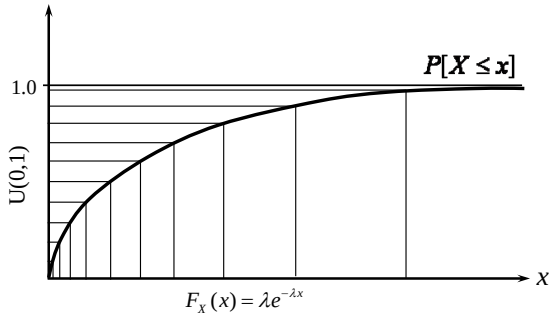
We can use this simple function to generate random variables with virtually any distribution. Let U be a random variable that is uniformly distributed between 0 and 1. If we want a random variable that is uniformly distributed between a and b, we first compute U and then calculate

$$X = a + (b - a)U.$$

It is fairly easy to generate random variables with general distributions if we can compute the inverse cumulative distribution function. Let $F(x) = P[X \leq x]$ be the cdf of a random variable X. If we have a way of generating a sample realization of X with the cumulative distribution $F(x)$, then if we let $Y = F(X)$ (where X is a



2.4a: Generating uniform random variables.



2.4a: Generating exponentially-distributed random variables.

Figure 2.4 Generating uniformly and exponentially distributed random variables using the inverse cumulative distribution method.

realization of our random variable), then it is a simple probability exercise to show that Y is uniformly distributed between 0 and 1. Now assume that we can find the inverse of the cumulative distribution function, which we represent as $F^{-1}(u)$ for a value $0 \leq u \leq 1$. If U is a random variable that is uniformly distributed between 0 and 1, then if we compute X using

$$X = F_X^{-1}(U),$$

we can show that X has the cumulative distribution $F(x)$.

For example, consider the case of an exponential density function $\lambda e^{-\lambda x}$ with cumulative distribution function $1 - e^{-\lambda x}$. Setting $U = 1 - e^{-\lambda x}$ and solving for x gives

$$X = -\frac{1}{\lambda} \ln(1 - U).$$

Since $1 - U$ is also uniformly distributed between 0 and 1, we can use

$$X = -\frac{1}{\lambda} \ln(U).$$

Figure 2.4 demonstrates how we can use a random variable that is uniformly distributed between 0 and 1 to create a random variable that is uniformly distributed

between a and b (in figure 2.4a), and a random variable that has an exponential distribution (in figure 2.4b).

We can use this method to compute random variables with a normal distribution. Excel and MATLAB, for example, have a function called $\texttt{NORMINV}(p, \mu, \sigma)$ where p is a probability, μ is the mean of a normally distributed random variable with standard deviation σ. Writing

$$x = \texttt{NORMINV}(p, \mu, \sigma)$$

in MATLAB (in Excel, you would enter "$= \texttt{NORMINV}(p, \mu, \sigma)$" in a cell) generates the value of a normally distributed random variable X such that $P(X \leq x) = p$. To generate a random variable that is normally distributed with mean μ and standard deviation σ, simply generate a uniform random variable U and them compute

$$x = \texttt{NORMINV}(U, \mu, \sigma).$$

Now imagine that we want to generate a Monte Carlo sample for a vector of correlated random variables. Let μ be a M-dimensional vector of means, and let Σ be a $M \times M$ covariance matrix. We would like to find a sample realization $u \sim N(\mu, \Sigma)$. This can be done very simply. Let C be the "square root" of Σ which is computed using Cholesky decomposition. In MATLAB, this done using

$$C = \texttt{chol}(\Sigma).$$

The result is an upper triangular matrix C, which is sometimes called the square root of Σ because

$$\Sigma = CC^T.$$

Let Z be an M-dimensional vector of independent, standard normal variables generated as we just described above. Given Z, we can compute a sample realization of μ using

$$u = \mu + CZ.$$

The vector u satisfies $\mathbb{E}u = 0$. To find the variance, we use

$$Cov(u) = \mathit{Var}(u + CZ) = CCov(Z)C^T.$$

Since the elements of the vector Z are independent with variance 1, $Cov(Z) = I$ (the identity matrix), which means $Cov(u) = CC^T = \Sigma$.

To illustrate, assume our vector of means is given by

$$\mu = \begin{bmatrix} 10 \\ 3 \\ 7 \end{bmatrix}.$$

our covariance matrix is given by

$$\Sigma = \begin{bmatrix} 9 & 3.31 & 0.1648 \\ 3.31 & 9 & 3.3109 \\ 0.1648 & 3.3109 & 9 \end{bmatrix}.$$

The Cholesky decomposition computed by MATLAB using $C = \texttt{chol}(\Sigma)$ is

$$C = \begin{bmatrix} 3 & 1.1033 & 0.0549 \\ 0 & 3 & 1.1651 \\ 0 & 0 & 3 \end{bmatrix}.$$

Imagine that we generate a vector Z of independent standard normal deviates

$$Z = \begin{bmatrix} 1.1 \\ -0.57 \\ 0.98 \end{bmatrix}.$$

Using this set of sample realizations of Z, a sample realization u would be

$$u = \begin{bmatrix} 10.7249 \\ 2.4318 \\ 7.9400 \end{bmatrix}.$$

Using computers to generate random numbers has proven to be an exceptionally powerful tool in the analysis of stochastic systems. Not surprisingly, then, the field has matured into a rich and deep area of study. This presentation is little more than a hint at the many tools available to help with the process of generating random numbers.

2.8 WHY DOES IT WORK?*

2.8.1 Derivation of $\tilde{\sigma}$

An important quantity in optimal learning is the variance $\tilde{\sigma}_x^n$ of $\bar{\mu}_x^{n+1}$ given what we know after n experiments.

Proposition 2.8.1 *The variance of $\bar{\mu}^{n+1}$, defined as*

$$\begin{aligned} \tilde{\sigma}^n &= \mathit{Var}^n[\bar{\mu}^{n+1}] \\ &= \mathit{Var}^n[\bar{\mu}^{n+1} - \bar{\mu}^n], \end{aligned}$$

is given by $(\tilde{\sigma}_x^n)^2 = (\sigma_x^n)^2 - (\sigma_x^{n+1})^2$.

Proof: Keep in mind that after n experiments, $\bar{\mu}_x^n$ is deterministic. We are dealing with two random variables: the truth μ_x, and the estimate $\bar{\mu}_x^{n+1}$ after we have made n experiments (the $n+1$st experiment, W^{n+1}, is unknown). We begin with the relation

$$(\bar{\mu}_x^{n+1} - \mu_x) = (\bar{\mu}_x^{n+1} - \bar{\mu}_x^n) + (\bar{\mu}_x^n - \mu_x). \tag{2.50}$$

Recall that $(\sigma_x^{n+1})^2 = \mathbb{E}^{n+1}\left[(\bar{\mu}_x^{n+1} - \mu_x)^2\right]$. Squaring both sides of (2.50) and taking the conditional expectation $\mathbb{E}^{n+1}(\cdot) = \mathbb{E}(\cdot|W^1, \ldots, W^{n+1})$ gives

$$\begin{aligned} (\sigma_x^{n+1})^2 &= \mathbb{E}^{n+1}\left[(\bar{\mu}_x^n - \mu_x)^2\right] + 2\mathbb{E}^{n+1}\left[(\bar{\mu}_x^n - \mu_x)(\bar{\mu}_x^{n+1} - \bar{\mu}_x^n)\right] \\ &\quad + \mathbb{E}^{n+1}\left[(\bar{\mu}_x^{n+1} - \bar{\mu}_x^n)^2\right] \\ &= \mathbb{E}^{n+1}\left[(\bar{\mu}_x^n - \mu_x)^2\right] + 2(\bar{\mu}_x^n - \bar{\mu}_x^{n+1})(\bar{\mu}_x^{n+1} - \bar{\mu}_x^n) + (\bar{\mu}_x^{n+1} - \bar{\mu}_x^n)^2 \\ &= \mathbb{E}^{n+1}\left[(\bar{\mu}_x^n - \mu_x)^2\right] - (\bar{\mu}_x^{n+1} - \bar{\mu}_x^n)^2. \end{aligned}$$

Keep in mind that $\mathbb{E}^{n+1}\bar{\mu}_x^{n+1} = \bar{\mu}_x^{n+1}$ and $\mathbb{E}^{n+1}\mu_x = \bar{\mu}_x^{n+1}$. We then observe that while $\bar{\mu}_x^{n+1}$ is random given the first n observations, σ_x^{n+1} is deterministic, because σ_x^{n+1} does not depend on W^{n+1} - it only depends on our decision of what to measure x^n. Using this property, we can take the expectation given $W^1, \ldots, W^n$ to obtain

$$\begin{aligned}\mathbb{E}^n(\sigma_x^{n+1})^2 = (\sigma_x^{n+1})^2 &= \mathbb{E}^n\left[\mathbb{E}^{n+1}\left[(\bar{\mu}_x^n - \mu_x)^2\right]\right] - \mathbb{E}^n\left[(\bar{\mu}_x^{n+1} - \bar{\mu}_x^n)^2\right] \\ &= \mathbb{E}^n\left[(\bar{\mu}_x^n - \mu_x)^2\right] - \mathbb{E}^n\left[(\bar{\mu}_x^{n+1} - \bar{\mu}_x^n)^2\right] \\ &= (\sigma_x^n)^2 - (\tilde{\sigma}_x^n)^2.\end{aligned}$$

2.8.2 Derivation of Bayesian updating equations for independent beliefs

Bayesian analysis begins with a simple formula that everyone learns in their first probability course. Given the events A and B, the basic properties of conditional probability imply

$$P(A, B) = P(A|B)P(B) = P(B|A)P(A)$$

which implies

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)}.$$

This expression is famously known as Bayes' theorem. In a learning setting, the event A refers to an experiment (or some type of new information), while B refers to the event that a parameter (say, the mean of a distribution) takes on a particular value. $P(B)$ refers to our initial (or prior) distribution of belief about the unknown parameter before we run an experiment, and $P(B|A)$ is the distribution of belief about the parameter after the experiment has been run. For this reason, $P(B|A)$ is known as the *posterior distribution*.

We can apply the same idea for continuous variables. We replace B with the event that $\mu = u$ (to be more precise, we replace B with the event that $u \leq \mu \leq u + du$), and A with the event that we observed $W = w$. Let $g(u)$ be our prior distribution of belief about the mean μ, and let $g(u|w)$ be the posterior distribution of belief about μ given that we observed $W = w$. We then let $f(w|u)$ be the distribution of the random variable W if $\mu = u$. We can now write our posterior $g(u|w)$, which is the density of μ given that we observe $W = w$, as

$$g(u|w) = \frac{f(w|u)g(u)}{f(w)},$$

where $f(w)$ is the unconditional density of the random variable W which we compute using

$$f(w) = \int_u f(w|u)g(u).$$

Equation (2.51) gives us the density of μ given that we have observed $W = w$.

We illustrate these calculations by assuming that our prior $g(u)$ follows the normal distribution with mean $\bar{\mu}^0$ and variance $\bar{\sigma}^{2,0}$, given by

$$g(u) = \frac{1}{\sqrt{2\pi}\sigma^0} \exp\left(-\frac{1}{2}\frac{(u-\bar{\mu}^0)^2}{\bar{\sigma}^{2,0}}\right).$$

We further assume that the observation W is also normally distributed with mean μ and variance σ_ϵ^2, which is sometimes referred to as the experimental or observation error. The conditional distribution $f(w|u)$ is

$$f(w|u) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2}\frac{(w-u)^2}{\sigma_\epsilon^2}\right).$$

We can compute $f(w)$ from $f(w|u)$ and $g(u)$, but it is only really necessary to find the density $g(u|w)$ up to a normalization constant ($f(w)$ is part of this normalization constant). For this reason, we can write

$$g(u|w) \propto f(w|u)g(u). \tag{2.51}$$

Using this reasoning, we can drop coefficients such as $\frac{1}{\sqrt{2\pi}\sigma^0}$, and write

$$\begin{aligned} g(u|w) &\propto \left[\exp\left(-\frac{1}{2}\frac{(w-u)^2}{\sigma^2}\right)\right] \cdot \left[\exp\left(-\frac{1}{2}\frac{(u-\bar{\mu}^0)^2}{\bar{\sigma}^{2,0}}\right)\right], \\ &\propto \exp\left[-\frac{1}{2}\left(\frac{(w-u)^2}{\sigma^2} + \frac{(u-\bar{\mu}^0)^2}{\bar{\sigma}^{2,0}}\right)\right]. \end{aligned} \tag{2.52}$$

After some algebra, we find that

$$g(u|w) \propto \exp -\frac{1}{2}\beta^1(u-\bar{\mu}^1)^2 \tag{2.53}$$

where

$$\bar{\mu}^1 = \frac{(\beta^W w + \beta^0\bar{\mu}^0)}{\alpha+\beta^0}, \tag{2.54}$$

$$\beta^1 = \beta^W + \beta^0. \tag{2.55}$$

The next step is to find the normalization constant (call it K) which we do by solving

$$K\int_u g(u|w)du = 1.$$

We could find the normalization constant by solving the integral and picking K so that $g(u|w)$ integrates to 1, but there is an easier way. What we are going to do is look around for a known probability density function with the same structure as (2.53), and then simply use its normalization constant. It is fairly easy to see that (2.53) corresponds to a normal distribution, which means that the normalization constant $K = \sqrt{\frac{\beta^1}{2\pi}}$. This means that our posterior density is given by

$$g(u|w) = \sqrt{\frac{\beta^1}{2\pi}} \exp -\frac{1}{2}\beta^1(u-\bar{\mu}^1)^2. \tag{2.56}$$

From equation (2.56), we see that the posterior density $g(u|w)$ is also normally distributed with mean $\bar{\mu}^1$ given by (2.54), and precision β^1 given by (2.55) (it is only now that we see our choice of notation $\bar{\mu}^1$ and β^1 in equations (2.54) and (2.55) was not an accident). This means that as long as we are willing to stay with our assumption of normality, then we need only to carry the mean and variance (or precision). The implication is that we can write our belief state as $B^n = (\bar{\mu}^n, \beta^n)$ (or $B^n = (\bar{\mu}^n, \bar{\sigma}^{2,n})$), and that (2.54)-(2.55) is our belief transition function.

Our derivation above was conducted in the context of the normal distribution, but we followed certain steps that can be applied to other distributions. These include:

1) We have to be given the prior $g(u)$ and the conditional density $f(w|u)$ of the experimental outcome.

2) We use equation (2.51) to find the posterior up to the constant of proportionality, as we did in (2.52) for the normal distribution.

3) We then manipulate the result in the hope of finding a posterior distribution, recognizing that we can discard terms that do not depend on u (these are absorbed into the normalization constant). If we are lucky, we will find that we have a conjugate family, and that we end up with the same class of distribution we started with for the prior. Otherwise, we are looking for a familiar distribution so that we do not have to compute the normalization constant ourselves.

4) We identify the transition equations that relate the parameters of the posterior to the parameters of the prior and the distribution of the experimental outcome, as we did with equations (2.54)-(2.55).

2.9 BIBLIOGRAPHIC NOTES

Sections 2.2-2.3 - There are numerous books on both frequentist and Bayesian statistics. An excellent reference for frequentist statistics is Hastie et al. (2009), which is available as of this writing as a free download in PDF form. An excellent reference for Bayesian statistics is Gelman et al. (2004). Another classical reference, with a focus on using Bayesian statistics to solve decision problems, is DeGroot (1970).

Section 2.6 - See Gelman et al. (2004) for a thorough treatment of Bayesian models.

Section 2.7 - There are a number of outstanding references on Monte Carlo simulation, including Banks et al. (1996), Roberts & Casella (2004), Glasserman (2004) and Rubinstein & Kroese (2008), to name a few.

PROBLEMS

2.1 In a spreadsheet, implement both the batch and recursive formulas for the frequentist estimates mean and variance of a set of random numbers (just use RAND() to produce random numbers between 0 and 1). Use a sequence of 20 random numbers. Note that Excel has functions to produce the batch estimates (AVERAGE and VAR) of

the mean and variance. Compare your results to Bayesian estimates of the mean and variance assuming that your prior is a mean of .5 and a variance of .2 (the prior estimate of the mean is correct, while the variance is incorrect).

2.2 Download the spreadsheet from the book website:

`http://optimallearning.princeton.edu/exercises/FiveAlternative.xls`

On day 0, the spreadsheet shows your initial estimate of the travel time on each path. In the column painted yellow, enter the path that you wish to follow the *next* day. You will see an observed time, and in the column to the right, we record the time you experience the next day when you follow your chosen path. To the right there is a work area where you can code your own calculations. Assume all random variables are normally distributed.

a) In the work area, use the Bayesian updating formulas to compute updated estimates of the mean and variance. Assume that the standard deviation of the observed travel time for any path is 5 minutes.

b) Using your estimates from part a, simulate a policy where you always choose the path that you thought was fastest. Record your total travel time, and the path that you thought was best.

c) You should find that you are getting stuck on one path, and that you do not "discover" the best path (you can quickly find that this is path 1). Suggest a policy that could be applied to any dataset (there can not be any hint that you are using your knowledge of the best path). Report your total travel time and the final path you choose.

2.3 You are trying to determine the distribution of how much people weigh in a population. Your prior distribution of belief about the mean μ is that it is normally distributed with mean 180 and standard deviation 40. You then observe the weights of n students drawn from this population. The average weight in the sample is $\bar{y}$ = 150 pounds. Assume that the weights are normally distributed with unknown mean μ and a known standard deviation of 20 pounds.

a) Give your posterior distribution for μ given the sample you have observed. Note that your answer will be a function of n.

b) A new observation is made and has a weight of $\tilde{y}$ pounds. Give the posterior distribution for $\tilde{y}$ (again, your answer will be a function of n).

c) Give a 95 percent posterior confidence intervals for μ and $\tilde{y}$ if $n = 10$.

d) Repeat (c) with $n = 100$.

2.4 Show that, for a fixed $k > 1$, equation (2.49) is equivalent to applying (2.45) k times in a row. Use the equivalence of (2.3) and (2.5). Now implement both the recursive and batch formulas in a spreadsheet and verify that they produce the same numbers.

2.5 In this problem, you will derive the Bayesian updating equations (2.30) and (2.31) for the gamma-exponential model. Suppose that W is a continuous random variable that follows an exponential distribution with parameter λ. The parameter λ is also random, reflecting our distribution of belief. We say that λ has a gamma prior by writing $\lambda \sim Gamma(a, b)$, meaning that our distribution of belief is gamma. Each time we observe W, we use this observation to update our belief about the true distribution of W, reflecting our uncertainty about λ.

a) Write down the prior density $f(u)$. What does u refer to?

b) Write down the conditional density $g(w|u)$ of the observation.

c) Write down the unconditional density $g(w)$ of the observation. (Hint: Start with $g(w|u)$ and take an expectation over the prior. Remember that the gamma function has the property that $\Gamma(a) = (a-1)\Gamma(a-1)$.)

d) Apply Bayes' rule to get $f(u|w)$, the posterior density after the observation. What distribution does this density correspond to? How does this verify equations (2.30) and (2.31)?

2.6 In this problem, you will see how the updating equations you derived in exercise 2.5 works in practice. Suppose that the customer service time at a certain store is exponentially distributed, and we are using the gamma-exponential model to learn the service rate as we observe the service times of individual customers.

a) Let U be a random variable that is uniformly distributed between 0 and 1. Let $R = -\frac{1}{\lambda}\log U$. Show that R follows an exponential distribution with parameter λ. This gives us a way to create samples from any exponential distribution by transforming samples from a uniform distribution (see section 2.4).

b) In a spreadsheet, use the above method to simulate 10 observations from an exponential distribution with parameter $\lambda = 3$. Now suppose that we do not know that λ has this value, and model our beliefs using a gamma prior with parameters $\alpha^0 = 1$ and $\beta^0 = 0.2$. What is our best initial estimate of λ? In your spreadsheet, record the values of α^n and β^n, as well as the resulting estimate of λ, for each n. Copy the results into a table and hand in a paper copy.

CHAPTER 3

OFFLINE LEARNING

Offline learning represents the set of problems where we have a budget to run N experiments, after which we have to use the information we gained to make the best decision possible based on what we learned about the function. One of the earliest versions of this problem arose in statistics under the name *ranking and selection*. The two defining characteristics of this problem class are:

- The possible decisions x fall in a finite set $\mathcal{X} = \{x_1, \ldots, x_M\}$, where x_m may be a categorical choice (type of drug or material, type of color), a discretized value of a continuous parameter, or a possibly sampled value of a multidmensional vector.

- We have a finite budget N, and we only care about the quality of the final decision rather than how well we do while we are learning. We refer to the performance at the end as the *terminal reward*.

Offline learning problems arise in many settings. We may have to choose a type of cholesterol drug, the parameters of a chemical process (e.g. temperature, relative mix of certain inputs), or the design of a manufacturing process (choice of equipment, size of buffers, routing of jobs). Testing an alternative might involve running a time-consuming computer simulation, or it might require a physical experiment. We assume that in either case, the experiment involves some noise, which means we may need to repeat the experiment several times. We assume that we have a budget that

Optimal Learning. By Warren B. Powell and Ilya O. Ryzhov

determines how many times we can perform these experiments, after which we have to take the best design and live with it.

We need to pause and comment on different interpretations of the terms "offline" and "online" (a class of problems we turn to in chapter 5). In this book, we interpret offline problems as having two distinct phases:

The information collection phase This is when we are running experiments to learn about the properties of a function or process, without regard to how well we are doing.

The implementation phase This is when we use what we have learned to design a system or process, and then we have to live with the performance of the resulting design.

By contrast, online problems represent problems that are running in the field, where we learn and implement at the same time. For example, testing medication on a patient would be an online problem; if we misdiagnose the condition, the patient will suffer. Similarly, trying to find the best price for a product requires setting a price and then receiving the associated revenue stream which may be lower than it could be, but we learn from this. This chapter focuses on offline learning. We turn to online learning, which has a separate but very rich history, in chapter 5.

This is not the only interpretation of offline vs. online. The machine learning community (to name one) uses offline learning to refer to using a batch dataset to estimate a function. By contrast, "online" refers to procedures that are sequential in nature. Implementing a process in the field is inherently sequential, hence the association of "online" with "sequential." But, learning in the lab or a simulation (offline) can also be sequential, and hence the machine learning community will interpret this setting as online as well, while recognizing that we do not care about how well we do.

In this chapter, we are going to show how to model offline learning problems. We then illustrate how to design and evaluate different learning policies for choosing the next experiment to run. The policies we introduce in this chapter are not the most sophisticated, although they can work quite well if properly tuned. Further, we are going to introduce new policies in chapter 5 for online learning that can also be applied to offline problems.

3.1 THE MODEL

The learning process proceeds as follows. Let S^n be our belief state, which captures the statistics we have collected from our first n observations using the methods we introduced in chapter 2. We assume we have some policy $X^\pi(S^n)$ that returns a decision x^n given our belief state that is given by S^n. We start at time $n = 0$, and make decision $x^0 = X^\pi(S^0)$ using our initial state S^0 (that is, our prior). Using the lookup table model we introduced in chapter 2, assume we choose $x^n = X^\pi(S^n)$ and then observe

$$W_{x^n}^{n+1} = \mu_{x^n} + \varepsilon^{n+1}. \tag{3.1}$$

If we are using a Bayesian model, our initial state S^0 might be that our prior is $\mu_x \sim N(\bar{\mu}_x^0, \beta_x^0)$. If we are using a frequentist model, our initial state contains no information, and we might be forced to test each x at least once (later in the book, we introduce parametric belief models that require fewer experiments to create a prior).

We then observe $W_{x_0}^1$, and use this information to update our belief state to obtain S^1 using the methods we described in chapter 2 (note that we can use either a Bayesian or frequentist belief state). We can write the sequence of states, decisions and information as

$$(S^0, x^0, W_{x^0}^1, S^1, x^1, \ldots, x^{N-1}, W_{x^{N-1}}^N, S^N).$$

We note that our indexing reveals the information content of each variable. Thus, S^{n-1} and x^{n-1} are computed without seeing W^n.

Keeping in mind that we do not update beliefs for alternatives that are not observed, if we are using a Bayesian model, we would update our belief state using

$$\bar{\mu}_x^{n+1} = \begin{cases} \frac{\beta_x^n \bar{\mu}_x^n + \beta_x^W W_x^{n+1}}{\beta_x^n + \beta_x^W} & \text{if } x^n = x \\ \bar{\mu}_x^n & \text{otherwise,} \end{cases} \tag{3.2}$$

$$\beta_x^{n+1} = \begin{cases} \beta_x^n + \beta_x^W & \text{if } x^n = x \\ \beta_x^n & \text{otherwise.} \end{cases} \tag{3.3}$$

Our belief state with the Bayesian model would be written

$$S^n = (\bar{\mu}_x^n, \beta_x^n)_{x \in \mathcal{X}}.$$

The goal of our problem is to design a policy that we designate $X^\pi(S^n)$ that returns a decision x^n which specifies the decision of what experiment to run next, which will yield the information W^{n+1}. If we are using a Bayesian belief model, our state is $S^n = (\bar{\mu}^n, \beta^n)$, which evolves according to a *transition function* that we denote by

$$S^{n+1} = S^M(S^n, x^n, W^{n+1}), \tag{3.4}$$

where the function $S^M(S, x, W)$ is known by names such as the transition function or the state transition model (hence the superscript "M"). For our Bayesian model, the transition function consists of equations (3.2) and (3.3).

If we use a frequentist model, the state variable is given by

$$S^n = (\bar{\mu}_x^n, \hat{\sigma}_x^{2,n}, N_x^n)_{x \in \mathcal{X}}$$

as developed in section 2.2, where N_x^n is the number of times we have tested alternative x, and where the transition equations are given by equations (2.4) and (2.5).

To evaluate a policy, we have to average its performance over many different realizations. For a frequentist model, the random variables are the observations $W^1, \ldots, W^N$ which produce the final estimates $\bar{\mu}_x^{\pi,N}$. We use this to choose our best decision (or design), given by

$$x^{\pi,N} = \arg\max_x \bar{\mu}_x^{\pi,N}. \tag{3.5}$$

We then choose the policy that works the best on average. Using our frequentist model, we need to recognize two sources of uncertainty: the sequence of observations $W^1, \ldots, W^N$ which occur while we are running our experiments, and the performance of the final design when it is implemented. We can write this as

$$F^\pi = \mathbb{E}_{W^1,\ldots,W^N}\mathbb{E}_W W_{x^{\pi,N}} \quad (3.6)$$

$$= \mathbb{E}_{W^1,\ldots,W^N}\mu_{x^{\pi,N}}. \quad (3.7)$$

If we are using a Bayesian model, we have to recognize one more source of uncertainty, which is the prior on the true value of μ. In our Bayesian model, each W^n depends on μ (as we saw in equation (3.1)). Equations (3.6) - (3.7) become

$$F^\pi = \mathbb{E}_\mu\mathbb{E}_{W^1,\ldots,W^N|\mu}\mathbb{E}_W W_{x^{\pi,N}} \quad (3.8)$$

$$= \mathbb{E}_\mu\mathbb{E}_{W^1,\ldots,W^N|\mu}\mu_{x^{\pi,N}}. \quad (3.9)$$

These expectations can look pretty frightening. Later in this chapter we discuss practical ways of estimating these in a straightforward way.

3.2 LEARNING POLICIES

Central to the concept of optimal learning is a learning policy. This is a rule that tells us which action x we should take next in order to observe something new. In addition, we may also be receiving rewards or incurring costs, which have to be balanced against the value of the information being gained.

In this section, we contrast deterministic versus sequential policies, and then provide a mathematical framework for finding optimal sequential policies. Unfortunately, this framework does not provide us with practical policies that we can compute. The section closes with a presentation of a number of the more popular heuristic policies that have been used on this problem class.

3.2.1 Deterministic vs. sequential policies

Before we begin our presentation, it is important to make a distinction between what we call *deterministic policies* and *sequential policies*:

Deterministic policies These are policies that determine all the experiments to run in advance, without learning the results of any of the experiments.

Sequential policies These policies use the results of the $n-1st$ experiment before deciding what to do for the nth experiment.

An example of a deterministic policy is where a business may decide to perform four market research studies in different parts of the country before finalizing the pricing and advertising strategy in a full roll-out to the entire country. The decision to do four studies (and their locations) is made before we have any information from any of the studies. We return to these policies in section 12.1.

By contrast, sequential policies assume that we make a decision x^{n-1}, then observe the result W^n, before making the next decision x^n. So, we try one path to work through the city before choosing the path we try the next day. Similarly, we see how a patient responds to one drug before trying another.

There are problem classes where deterministic policies are optimal. For example, we might be interested in running experiments that minimizes some function of the variance of the quantities that we are trying to estimate. If you take a close look at our formula for updating the variance (or equivalently the precision) in equation (3.3), we see that our estimate of the variance is a deterministic function of what we choose to measure. This means that any rule that depends purely on the variance can be solved deterministically.

Our interest is primarily in sequential policies, where the decision of what to measure next may depend on past experiments. For example, when we are trying to find the shortest path, we may decide to continue sampling a path if it remains competitive, or give up on a path if the observed travel times are simply too long. Our decisions of what to measure in this case depend on the outcomes of prior experiments.

3.2.2 Optimal learning policies

It is possible to provide a mathematical characterization of an optimal learning policy. We start by considering a simple shortest path problem, illustrated in figure 3.1(a), where we are in state (node) 2, considering a decision to go to state (node) 5. The field of dynamic programming tells us that the value of being in a state S, given by $V(S)$, satisfies Bellman's equation which is given by

$$V(S) = \max_x \big(C(S,x) + V(S^M(S,x))\big). \tag{3.10}$$

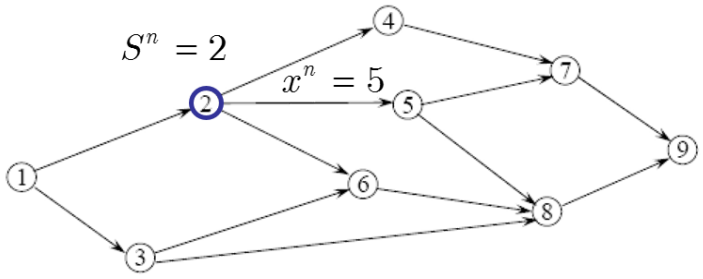
Here, $S^M(S,x)$ is known as the transition function, which determines the state we transition to if we make decision x. For our shortest path problem, x (the decision to go to node 5) determines the downstream state deterministically. There are many problems where the downstream state involves a combination of the decision x and random information W. For example, we could model time, where every node is a point in space and time. We might model randomness in travel times, which introduces randomness in when we arrive. In this case, we would write our transition as $S' = S^M(S,x,W)$, and write Bellman's equation as

$$V(S) = \max_x \big(C(S,x) + \mathbb{E}_W V(S^M(S,x,W))\big). \tag{3.11}$$

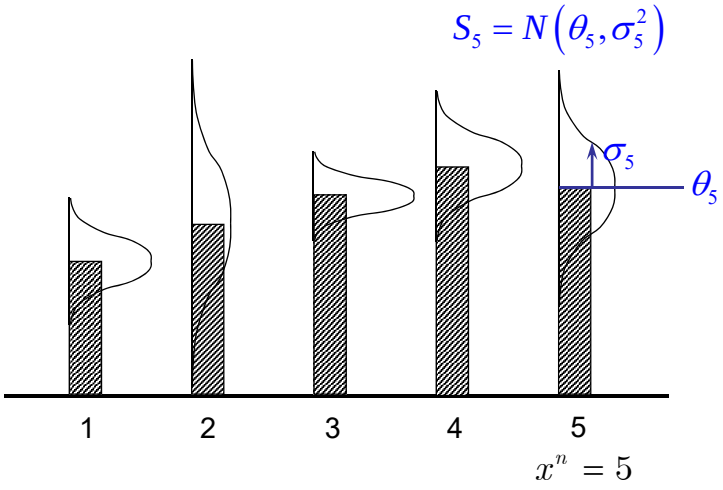
We can solve equation (3.10) or (3.11) by starting at the ending state(s) where $V(S) = 0$, and then stepping backward. This, in fact, is how shortest path problems are actually solved in practice. Once we have the value functions, our policy is given by

$$X^*(S) = \arg\max_x \big(C(S,x) + \mathbb{E}_W V(S^M(S,x,W))\big). \tag{3.12}$$

Now consider the situation in figure 3.1(b), where now our state is the belief state about the performance of each of five choices. These choices might represent five



(a)

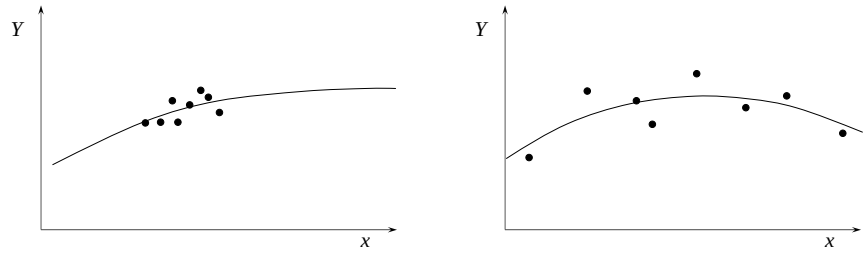


(b)

Figure 3.1 (a) Illustration of a shortest path problem, where we are in state (node) 2, considering going to node 5; (b) Illustration of a learning problem, where the distributions represent our current belief state, and we are considering making a decision to evaluate alternative 5.

medications for lowering blood sugar, and the distributions represent our belief about how each medication will affect the patient. In this context, the transition function $S^M(S, x, W)$ is precisely our belief updating functions (3.2) and (3.3).

The problem with our optimal policy in (3.12) is one of computation. We can solve these problems when the state S is a set of discrete nodes (even if there are a lot of them). But we face a completely different problem if the state is a multidimensional set of continuous variables such as the means and precisions of all the beliefs. However, there are special cases where (3.12) can be solved, and without pointing it out, we already did this in section 1.6 when we set up and solved the decision tree for our problem of finding the next baseball player to send up to bat. The key feature of this problem was that the information was in the form of discrete (binary) random variables, which means that the belief state was also discrete.



3.2(a) 3.2(b)

Figure 3.2 Estimating a function where we sample closest to the point that might be best (a), versus sampling a wide range of points so that we get a better estimate of the function (b).

3.2.3 Heuristic policies

Our goal, ultimately, is to find the best possible policies for learning. The reality, however, is that most of the time we are happy to find good policies. Below are some popular methods that have been suggested for problems that are typically associated with discrete selection problems, which is to say that the set of experimental decisions is discrete and "not too large."

Pure exploration A pure exploration strategy might sample a decision $x^n = x$ with probability $1/M$ (the probabilities do not have to be the same - they just have to be strictly positive). We would only use a pure exploration policy if we were focusing purely on estimating the value of each choice, as opposed to making a good economic decision. If we really are trying to find the best value of μ_x, a pure exploration strategy means that we would spend a lot of time measuring suboptimal choices.

Pure exploration can be effective for offline learning problems, especially when the number of choices is extremely large (and especially if the experimental design x is multidimensional). This is often what has been used when we are given a dataset of observations from which we have to fit a model so that we can find the best choice or design. With offline learning, it does not hurt us to observe a poor choice, and extreme choices can give us the best estimates of a function. For example, consider the problem of fitting a linear regression model of the form

$$Y = \theta_0 + \theta_1 x + \theta_2 x^2 + \epsilon.$$

Imagine that x is a scalar between 0 and 10, and we believe the highest values of Y are likely to be found close to the middle of the range. Figure 3.2(a) shows what happens when we draw most of our samples from a narrow range. We may get a lot of data to estimate the function around those points, but we do not get the kind of information we need to get an accurate estimate of the function, which would allow us to do the best job finding the maximum. In Figure 3.2(b), we explore a wider range of points, which allows us to do a better job of estimating the entire curve. For instance, we discover that Y decreases once x is large enough, whereas 3.2(a) leads us to believe that Y always increases with x.

Pure exploitation Exploitation means making the best decision given our current set of estimates (we are "exploiting" our knowledge). So, after iteration n, we would next measure

$$x^n = \arg\max_{x \in \mathcal{X}} \bar{\mu}_x^n.$$

This strategy would seem to focus our energy on the options that appear to be the best. However, it is very easy to get stuck measuring choices that seem to be the best, especially when we simply had some bad luck measuring the better choices.

Pure exploitation is a common strategy in online problems, where we have to live with the results of each experiment. With pure exploitation, we can always defend our choice because we are doing what we believe is the best, but we may be ignoring errors in our own beliefs.

Excitation policies A popular policy in the control of engineering systems with continuous controls is to assume we have some policy that specifies the action we think is best and then adds a random noise term. For example, we might specify the rate at which gasoline flows into a combustion chamber as a function of the speed S_t of a vehicle. Assume that we use the following policy (known as a control law in engineering):

$$X^\pi(S_t|\theta) = \theta_0 + \theta_1 S_t + \theta_2 S_t^2.$$

Presumably we think this is the policy that produces the best gas mileage while maintaining the speed of the vehicle, so we would call this a pure exploitation policy. We can introduce a random noise term to induce exploration using

$$X^\pi(S_t|\theta) = \theta_0 + \theta_1 S_t + \theta_2 S_t^2 + \varepsilon.$$

The noise ε is known in engineering as *excitation*, and it helps to learn new behaviors.

Epsilon-greedy exploration A simple strategy that avoids the limitations of pure exploration and pure exploitation is to use a mixed strategy, where we explore with probability ϵ (known as the exploration rate) and we exploit with probability $1 - \epsilon$. The value of ϵ has to be tuned for each application.

Mixing exploration and exploitation is appealing because it allows you to spend more time evaluating the choices that appear to be best (to make sure this is the case) while still doing a certain amount of exploration. As our experimentation budget goes to infinity, we can still provide guarantees that we will find the best alternative because of the exploration component. But this policy still suffers from the significant limitation that when we do choose to explore, we sample from the entire population of alternatives, which may be extremely large, including choices that are clearly suboptimal.

The problem with a mixed exploration/exploitation strategy with fixed ϵ is that the correct balancing of exploration and exploitation changes with the number of iterations. In the beginning, it is better to explore. As we build confidence in our estimates, we would prefer to exploit more. We can do this by using an exploration

probability ϵ^n at iteration n that declines with n. We have to make sure it does not decline too quickly. We do this by setting

$$\epsilon^n = c/n$$

for $0 < c < 1$. If we explore, we would choose experimental design x with probability $1/|\mathcal{X}|$. This means that in the limit, the number of times we will measure x is given by

$$\sum_{n=1}^{\infty} \frac{c}{n|\mathcal{X}|} = \infty.$$

This assures us that we will estimate each experiment x perfectly, but as the experiments progress, we will spend more time measuring what we think are the best choices.

Boltzmann exploration A different strategy for balancing exploration and exploitation is known as Boltzmann exploration. With this strategy, we run experiment x with probability p_x^n given by

$$p_x^n(\theta^B) = \frac{e^{\theta^B \bar{\mu}_x^n}}{\sum_{x' \in \mathcal{X}} e^{\theta^B \bar{\mu}_{x'}^n}}. \tag{3.13}$$

This policy is also known as the *soft max* policy. If $\theta^B = 0$, we are going to run each experiment x with equal probability (pure exploration). As $\theta^B \to \infty$, we will run the experiment with the highest value of $\bar{\mu}^n$ with probability 1 (pure exploitation). In between, we explore the better options with higher probability. Furthermore, we can make θ^B increase with n (which is typical), so that we explore more in the beginning, converging to a pure exploitation strategy. Tuning such policies, however, can be tricky; we return to this below in section 3.7.

Care should be used when computing the probabilities using a Boltzmann distribution, especially if you are increasing θ^B as you progress to focus attention on the best alternatives. The problem is that the exponent $\theta^B \bar{\mu}_x^n$ can become so large as to make it impossible to evaluate $e^{\theta^B \bar{\mu}_x^n}$. A better way is to first compute

$$\bar{\mu}^n = \max_{x \in \mathcal{X}} \bar{\mu}_x^n,$$

and then compute the probabilities using

$$p_x^n(\theta^B) = \frac{e^{\theta^B (\bar{\mu}_x^n - \bar{\mu}^n)}}{\sum_{x' \in \mathcal{X}} e^{\theta^B (\bar{\mu}_{x'}^n - \bar{\mu}^n)}}.$$

This calculation can be further streamlined by excluding any choices x where $\bar{\mu}_x^n$ is sufficiently far from $\bar{\mu}^n$ (for example, where $\theta^B |\bar{\mu}^n - \bar{\mu}_x^n| > 10$).

We write the Boltzmann policy by first generating a random variable U that is uniform on $[0, 1]$. To simplify notation, assume that $x_m = m$. Next, create a cumulative distribution $P_x^n(\theta^B) = \sum_{x'=1}^{x} p_{x'}^n(\theta^B)$. Finally, we can write our policy using

$$X^{Boltz}(S^n|\theta^B) = \arg\max_x \{x | P_x^n(\theta^B) \leq U\}. \tag{3.14}$$

Boltzmann exploration is more effective than epsilon-greedy which either chooses the very best, or chooses among the rest purely at random. Boltzmann focuses its efforts more on the choices that are at least somewhat attractive. However, it is missing an important feature that is enjoyed by the policies which follow.

Upper confidence bounding UCB policies (as they are known) were developed primarily for the setting of online learning, so we are going to cover these in more depth in chapter 5. However, they represent an important class of policy, and they can be used in offline settings if they are properly tuned (the same can be said of all of the heuristic policies described in this section).

A basic version of a UCB policy (known as "UCB1" in the research literature) is given by

$$X^{UCB1,n}(S^n|\theta^{UCB}) = \arg\max_x \left(\bar{\mu}_x^n + \theta^{UCB} \sqrt{\frac{2\log n}{N_x^n}}. \right) \tag{3.15}$$

The parameter θ^{UCB} is one that we have introduced, and we will have to tune it for an offline setting (this is discussed below).

Thompson sampling Thompson sampling is a simple policy based on the idea of created a *sampled* estimate of what *might* happen in the next experiment for each possible design. We illustrate using a Bayesian belief model, although a frequentist model could be used as well.

Assume that after n experiments, we characterize our belief about $\mu_x \sim N(\bar{\mu}_x^n, \bar{\sigma}_x^{2,n})$. Now, randomly sample $\hat{\mu}_x^{n+1}$ from the belief $N(\bar{\mu}_x^n, \bar{\sigma}_x^{2,n})$, where $\mathbb{E}\hat{\mu}_x^{n+1} = \bar{\mu}_x^n$. However, $\hat{\mu}_x^{n+1}$ may be higher or lower than $\bar{\mu}_x^n$. The Thompson sampling policy is then given simply by

$$X^{TS}(S^n) = \arg\max_x \hat{\mu}_x^{n+1}. \tag{3.16}$$

The randomization in choosing $\hat{\mu}_x^{n+1}$ is the mechanism by which Thompson sampling explores different alternatives, with a bias toward the choices that appear to be better.

Interval estimation Imagine that instead of running the experiment that we think is best, we choose an alternative that we think might learn is the best if we were to take enough observations. With this idea, we might construct a confidence interval and then value an option based on the upper limit of, say, a 95% confidence interval. Letting α be our confidence level and denoting by z_α the standard normal deviate leaving α in the upper tail, our upper limit would be

$$\nu_x^{IE,n} = \bar{\mu}_x^n + z_\alpha \sigma_x^n \tag{3.17}$$

where $\sigma_x^n = \sqrt{\frac{1}{\beta_x^n}}$ is the standard deviation of the distribution of our belief at time n. When we use an interval exploration policy, we choose the experiment x^n with the highest value of $\nu_x^{IE,n}$.

Although the interpretation as the upper limit of a confidence interval is appealing, the confidence level α carries no particular meaning. In fact, the policy we would

implement in practice looks like

$$X^{IE}(S^n|\theta^{IE}) = \arg\max_x \left(\bar{\mu}_x^n + \theta^{IE}\bar{\sigma}_x^n\right), \tag{3.18}$$

where now, θ is simply a tunable parameter. It has been reported in the literature that values around 2 or 3 work best for many applications, but it is possible to construct problems where the best value may be anywhere from 0.1 to 100 (the high values arise when the priors are really poor). Furthermore, it has been found that the algorithm can be very sensitive to the choice of θ. However, if properly tuned, this policy can work extremely well in many settings.

Interval estimation introduces two important dimensions relative to policies based on exploration and exploitation. First, like Boltzmann estimation, the likelihood of measuring a choice depends on how well we think the choice may work. This means that we will avoid exploring options that seem to be genuinely bad. Second, we are more willing to explore options which we are more uncertain about. The term $z_\alpha\sigma_x^n$ has been called the "uncertainty bonus." As we explore an option with a high value of $\nu_x^{IE,n}$, σ_x^n will decrease, which will often push us to try other options.

IE is not guaranteed to find the best option, even with an infinitely large experimental budget. It is possible for us to get some poor initial estimates of what might be the best option. If $\bar{\mu}_x^n$ is low enough for some choice x, we may never revisit this choice again. Our experimental work suggests that this can happen, but rarely.

3.3 SIMULATING OUTCOMES

Now that we have covered a menu of policies, we have to address the problem of evaluating them to determine which one is best. We are going to do this by simulating the policy, which means we have to simulate the sequence of experimental outcomes $W^1, \ldots, W^N$ that result from using a policy. There are three ways to do this. The first two are independent of the policy, which allows them to be done in advance. The last is a policy-dependent representation which may seem more natural and can be easier to implement, but introduces a slight problem when comparing policies.

3.3.1 Policy-independent representations

For Bayesian belief models, there are two ways to represent the outcome of an experiment which are not dependent on the policy. The first models the truth μ and the experimental outcomes $W^1, \ldots, W^N$ separately, while the second blends these. We start with the first method.

We use the notation ω to denote a sample realization of all the observations made from a sequence of experiments. Think of a matrix $W(\omega)$ of numbers, such that the number in the nth row and xth column represents a sample realization $W_x^n(\omega)$ of the observation W_x^n. Figure 3.1 illustrates this idea with three sample realizations of W_x^n (three different values of ω) for $n = 1, ..., 10$ and $x = 1, 2, 3$. It is useful to think of generating realizations for every possible experiment, but then we are going to choose a learning policy $X^\pi(S)$ that determines which of these realizations that

	$\omega = \omega_1$			$\omega = \omega_2$			$\omega = \omega_3$		
n	W_1^n	W_2^n	W_3^n	W_1^n	W_2^n	W_3^n	W_1^n	W_2^n	W_3^n
1	37.4	64.1	59.2	38.1	66.9	55.7	33.4	66.5	40.1
2	24.3	65.2	56.0	28.0	57.6	59.3	25.4	60.1	59.5
3	30.5	64.5	56.5	38.9	59.3	50.2	22.7	59.5	48.7
4	20.5	63.2	55.9	34.1	57.7	57.1	24.1	59.8	58.8
5	29.1	64.5	44.8	36.3	60.1	56.4	37.3	65.1	57.1
6	36.7	62.1	59.6	37.0	53.6	42.9	20.7	51.8	42.9
7	27.3	53.5	44.8	27.7	60.9	48.9	32.0	53.4	53.3
8	26.8	57.7	48.1	28.8	68.6	54.4	34.3	53.0	48.6
9	40.0	58.6	42.8	39.6	52.8	53.5	31.1	64.0	59.9
10	21.4	51.6	56.4	33.1	54.5	42.7	33.4	56.7	42.7

Table 3.1 Three sample realizations of three alternatives over 10 observations.

	$\omega = \omega_1$			$\omega = \omega_2$			$\omega = \omega_3$		
	$\mu_1(\omega_1)$	$\mu_2(\omega_1)$	$\mu_3(\omega_1)$	$\mu_1(\omega_2)$	$\mu_2(\omega_2)$	$\mu_3(\omega_2)$	$\mu_1(\omega_3)$	$\mu_2(\omega_3)$	$\mu_3(\omega_3)$
	32.1	57.8	55.8	29.5	62.3	60.1	34.4	59.2	59.7
n	W_1^n	W_2^n	W_3^n	W_1^n	W_2^n	W_3^n	W_1^n	W_2^n	W_3^n
1	33.8	57.7	53.3	32.6	62.7	60.5	29.6	55.7	62.6
2	37.0	55.0	59.7	26.0	60.8	55.3	36.8	62.1	59.1
3	36.0	54.9	51.9	34.3	65.0	58.3	38.0	60.3	60.4
4	30.8	53.3	58.6	30.0	57.7	57.8	36.1	59.9	63.2
5	30.9	58.6	58.3	30.7	59.4	61.0	38.6	58.3	63.2
6	29.7	56.3	55.6	26.3	62.4	60.9	33.9	54.6	64.5
7	32.1	57.0	57.9	31.6	57.8	59.9	30.8	63.3	55.3
8	31.1	53.9	54.4	29.1	59.6	59.0	31.6	58.9	60.8
9	35.9	55.5	50.8	31.5	65.9	60.8	36.0	62.3	62.9
10	35.4	60.5	54.7	33.7	62.4	55.9	38.9	54.9	64.2

Table 3.2 Three sample realizations of both a truth μ and a set of sample realizations W drawn from this truth.

we are actually going to see. That is, we only get to see W_x^n if we choose to observe $x = X^\pi(S^n)$.

Since the true values μ_x are also random variables (in our Bayesian model), we can let ψ be a sample realization of the truth. That is, $\mu(\psi)$ is a particular set of truth values $\mu_{x_1}(\psi), ..., \mu_{x_M}(\psi)$ for the different alternatives. Our Bayesian model makes the fundamental assumption that $\mu_x \sim \mathcal{N}(\bar{\mu}_x^0, \beta_x^0)$, which means that we assume our prior distribution is assumed to be accurate *on average*. Therefore, the sample path ψ is generated from the prior distribution. This is sometimes known as the *truth from prior* assumption. Of course, we have to recognize that our prior may not be accurate, but we generally assume it is unbiased.

	$\omega = \omega_1$			$\omega = \omega_2$			$\omega = \omega_3$		
n	W_1^n	W_2^n	W_3^n	W_1^n	W_2^n	W_3^n	W_1^n	W_2^n	W_3^n
1	37.4	-	-	-	66.9	-	-	66.5	-
2	-	65.2	-	28.0	-	-	-	60.1	-
3	30.5	-	-	-	-	50.2	22.7	-	-
4	-	-	55.9	34.1	-	-	-	-	58.8
5	-	64.5	-	-	60.1	-	37.3	-	-
6	-	62.1	-	37.0	-	-	-	51.8	-
7	-	-	44.8	-	-	48.9	-	-	53.3
8	26.8	-	-	-	-	54.4	34.3	-	-
9	-	-	42.8	-	52.8	-	31.1	-	-
10	-	51.6	-	33.1	-	-	-	56.7	-

Table 3.3 Three sample realizations where we only model actual observations generated by following a policy.

The second strategy for representing the outcome of a set of experiments is to combine the uncertainty about the truth and the uncertainty about experiments into a single space of outcomes. Table 3.2 illustrates how we might represent three sample realizations. Here, ω represents both a realization of the truth μ as well as a realization of the observations W_x^n drawn from this truth.

3.3.2 Policy-dependent representation

The third approach for representing the outcome of a set of experiments (and perhaps the most natural) is to model the sequence of decisions x^n and corresponding experimental outcomes W^{n+1}. If we are using a frequentist representation, a sample realization would be

$$(x^0, W^1, x^1, W^2, \ldots, x^{N-1}, W^N).$$

In this representation, we let ω be a single sample path of decisions and outcomes. Since the decisions depend on what policy we are following, we would write our set of all possible outcomes as Ω^π. One way to represent sample paths is depicted in table 3.3, where we only show the observations of alternatives we actually observe.

3.3.3 Discussion

The policy-independent representations require that we pre-generate all the sample realizations and store these in a file (typically). While this requires more work, it allows us to compare different policies on the same sample realizations, which simplifies the comparison of policies. By contrast, it is easier to generate outcomes W_x^n on the fly while following a policy, but this typically means comparing policies using different sample outcomes.

With both methods, if we are using a Bayesian model, we can generate truths $\mu(\psi)$ independently of the experimental outcomes $W^1(\omega), \ldots, W^N(\omega)$, or we can generate

them simultaneously, creating a single sample realization $(\mu(\omega), W^1(\omega), \ldots, W^N(\omega))$. For the same number of simulations, the latter approach will produce a better mixture of truths and experimental outcomes.

If we are using a frequentist belief model, we skip the sampling of a truth. Instead, we have to sample the experimental outcomes W from an assumed distribution.

3.4 EVALUATING POLICIES

We now want to estimate the performance of policy $X^\pi(S)$. Our challenge is to evaluate a policy which we might do using our Bayesian objective (equation (3.9)) given by

$$F^\pi = \mathbb{E}_\mu \mathbb{E}_{W^1,\ldots,W^N|\mu} \mu_{x^{\pi,N}}.$$

The problem is that we cannot actually compute these expectations. For this reason, we have to use Monte Carlo simulation to approximate the value of a policy.

Recall that $x^{\pi,N}$ is the best decision based on our final estimates $\bar{\mu}_x^{\pi,N}$ computed when we follow policy π with an experimental budget of N observations (see equation (3.5)). We refine this notation by indexing it by the sample truth and the sample of observations, which gives us

$x_x^{\pi,N}(\psi,\omega)$ = The optimal decision based on the estimates $\bar{\mu}_x^{\pi,N}$ for each alternative x when the truth is $\mu(\psi)$ and the observations are $W^1(\omega), \ldots, W^N(\omega)$.

There are two ways we can evaluate the performance of a policy. The first is based purely on the actual estimates of the value of each alternative. Let

$\bar{\mu}_x^{\pi,N}(\psi,\omega)$ = The estimate of μ_x if $\mu_x(\psi)$ is the true value of μ, we observe $W(\omega)$ while following policy $X^\pi(S)$ with a budget N.

Remember that each observation $W_x^n(\omega)$ is generated using

$$W_{x^n}^{n+1}(\omega) = \mu_{x^n}(\psi) + \varepsilon^{n+1}(\omega)$$

for a given truth $\mu_x(\omega)$. With this, we finally can compute a sample realization of the value of a policy using

$$\begin{aligned} F^\pi(\mu(\psi), W(\omega)) &= \max_{x\in\mathcal{X}} \bar{\mu}_x^{\pi,N}(\psi,\omega) \\ &= \bar{\mu}_{x^{\pi,N}(\psi,\omega)}^{\pi,N}(\psi,\omega). \end{aligned} \tag{3.19}$$

Equation (3.19) is simply using our final estimate of each μ_x to evaluate how well the policy has performed. Naturally, we need to average over different truths $\mu(\psi)$ and sample observations $W^1(\omega), \ldots, W^N(\omega)$. If we sample L truths $\mu(\psi_\ell)$ and K experiments $W^1(\omega_k), \ldots, W^N(\omega_k)$, then we can compute an estimate of the performance of the policy using

$$\overline{F}^\pi = \frac{1}{L}\sum_{\ell=1}^{L}\left(\frac{1}{K}\sum_{k=1}^{K} F^\pi(\mu(\psi_\ell), W(\omega_k))\right). \tag{3.20}$$

We can also use the representation illustrated in table 3.2 where we sample the truth μ and the outcomes $W^1, \ldots, W^N$ simultaneously, rather than in the nested fashion that we use in our estimate in equation (3.20). In this case, we let $\bar{\mu}_x^{\pi,N}(\omega)$ be the estimate of μ_x when we sample the combination of truth and experimental outcomes $(\mu(\omega), W^1(\omega), \ldots, W^N(\omega))$. We also let $x^{\pi,N}(\omega)$ be the optimal solution given by

$$x^{\pi,N}(\omega) = \arg\max_x \bar{\mu}_x^{\pi,N}(\omega).$$

Using this representation, our estimate of the value of a policy would be written

$$\overline{F}^{\pi} = \frac{1}{K}\sum_{k=1}^{K} F^{\pi}(\mu(\omega_k), W(\omega_k)). \tag{3.21}$$

An important feature of equation (3.20) is that it requires nothing more than sampled observations that could come from any field source. However, it is often the case that we are using simulations to evaluate and compare policies that would then be used in the field. When we are simulating a policy, we do this using a simulated truth $\mu(\psi)$. When this is the case, then we can evaluate the policy using the actual truth, equation (3.19) becomes

$$\begin{aligned} F^{\pi}(\mu(\psi), W(\omega)) &= \max_{x\in\mathcal{X}} \mu_x(\psi, \omega) \\ &= \mu_{x^{\pi,N}(\psi,\omega)}(\psi, \omega). \end{aligned} \tag{3.22}$$

We then use this method for computing $F^{\pi}(\mu(\psi), W(\omega))$ in equation (3.20). Equation (3.22) avoids the noise inherent when using the estimates $\bar{\mu}^{\pi,N}_{x^{\pi,N}(\psi,\omega)}(\psi, \omega)$.

A particularly powerful idea is to compute what is known variously as the *opportunity cost* or the *regret* which calculates how much worse we are doing than the optimal solution. Using our ability to use the known truth (because we are simulating it), we can find the difference between how well we do using our estimates to find the best, and what is truly the best. This is given by

$$\bar{R}^{\pi} = \frac{1}{L}\sum_{\ell=1}^{L}\left(\frac{1}{K}\sum_{k=1}^{K}\left(\mu_{x*}(\psi) - \mu_{x^{\pi}(\mu(\psi_\ell),W(\omega_k))}(\psi)\right)\right). \tag{3.23}$$

where $\mu_{x*}(\psi) = \max_x \mu_x(\psi)$ is the best we can do for a particular truth. The regret has a lower bound of zero, which provides a nice reference point.

There are other ways to evaluate a policy which we discuss in greater depth in chapter 6. However, most of the time we will use either the expected performance or, quite frequently, the opportunity cost (or regret).

We suggest that the best environment for identifying good learning policies is inside the computer, where you can assume a truth and then try to discover the truth. Once you have decided on a good learning policy, you can go to the field where the truth is unknown, and you only have access to the estimates $\bar{\mu}_x^{\pi,N}$.

3.5 HANDLING COMPLEX DYNAMICS

It is useful to note that the process of evaluating policies is quite robust to the underlying dynamics or choice of belief model. For example, in chapter 2 we

introduced updating equations if we have correlated beliefs. Our process of simulating policies is quite robust to the introduction of correlated beliefs, or more complex information processes.

To see this, imagine that we have a current belief state $S^n = (\bar{\mu}^n, \Sigma^n)$ where $\bar{\mu}^n$ is a vector with element $\bar{\mu}_x^n$ (as we have assumed above), and where Σ^n is our estimate of the covariance matrix in our beliefs, with element

$$\Sigma_{xx'} = Cov^n(\mu_x, \mu_{x'}).$$

This is the covariance in our belief about the truths μ_x and $\mu_{x'}$ after we have observed $W^1, \ldots, W^n$. After choosing x^n and observing W^{n+1}, we can then use the updating equations for correlated beliefs (see section 2.3.2) to produce updated estimates $\bar{\mu}^{n+1}$ and Σ^{n+1}.

Similarly, our information process W may have its own complex dynamics. For example, we might use a time series equation of the form

$$W^{n+1} = \theta_0 W^n + \theta_1 W^{n-1} + \theta_2 W^{n-2}.$$

Incorporating more complex time series models that depend on trailing observations does not affect our process of testing and comparing policies.

3.6 EQUIVALENCE OF USING TRUE MEANS AND SAMPLE ESTIMATES*

It turns out that our evaluation of the policy using sampled estimates, given in equation (3.19), is the same (on average) as that computed using the real truth, given in equation (3.22). This result means that

$$\mathbb{E}_\mu \mathbb{E}_{W^1,\ldots,W^N|\mu} \mu_{x^\pi} = \mathbb{E}_\mu \mathbb{E}_{W^1,\ldots,W^N|\mu} \max_x \bar{\mu}_x^{\pi,N}. \tag{3.24}$$

Below, we are going to use $\mathbb{E}$ instead of the nested $\mathbb{E}_\mu \mathbb{E}_{W^1,\ldots,W^N|\mu}$. When using the single expectation operator, it is easiest to think of this as using the combined sampling of both the truth and the outcomes as we illustrated in table 3.2.

To demonstrate equation (3.24), recall that $\mathbb{E}^N \mu_x = \bar{\mu}_x^{\pi,N}$ for any fixed x. We use the tower property of conditional expectation, which simply means that we can nest expectations. For example, imagine that we have random variables A and B, Assume that the expectation $\mathbb{E}$ refers to taking the expectation over the combination of A and B, which we might write as $\mathbb{E}_{A,B}$. The tower property can be written

$$\mathbb{E}X = \mathbb{E}_{A,B}X = \mathbb{E}_A \mathbb{E}_{B|A} X.$$

For our problem, we can think of the truth μ as the random variable A, and then let the sequence $W^1, \ldots, W^N$ be the random variable B. The tower property allows us to write

$$\mathbb{E}\mu_{x^\pi} = \mathbb{E}_\mu \mathbb{E}_{W^1,\ldots,W^N|\mu} \mu_{x^\pi} = \mathbb{E}\bar{\mu}_{x^\pi}^{\pi,N},$$

because $x^\pi = \arg\max_x \bar{\mu}_x^{\pi,N}$ is known at time N (that is, it is fixed from the point of view of our time-N beliefs). However, $\bar{\mu}_{x^\pi}^{\pi,N} = \max_x \bar{\mu}_x^{\pi,N}$ by definition of x^π.

There is an interesting corollary to this result. Denote by χ an "implementation policy" for selecting an alternative at time N. We can think of χ as a function mapping the belief state S^N to an alternative $\chi(S^n) \in \{x_1, ..., x_M\}$. Then,

$$\max_{\chi} \mathbb{E}\mu_{\chi(S^N)} = \max_{x} \bar{\mu}_x^{\pi,N}.$$

In other words, the optimal decision at time N is always to select $\bar{\mu}_x^{\pi,N}$. If we have no more opportunities to learn, the best possible decision we can make is to go with our final set of beliefs. This result addresses the issue of why most of our policies seek to maximize $\max_x \bar{\mu}_x^{\pi,N}$ in some way, even though what we really want to learn is the unknown true value $\max_x \mu_x$.

3.7 TUNING POLICIES

Most heuristic policies have a tunable parameter. Perhaps the simplest illustration is the interval estimation policy, which we restate for easy reference as

$$X^{IE}(S^n|\theta^{IE}) = \arg\max_x \left(\bar{\mu}_x^n + \theta^{IE}\sigma_x^n\right).$$

We now have the problem of tuning θ^{IE} so that the policy performs at its best. When working with a specific policy such as interval estimation, the problem of searching over policies π becomes one of searching over values of θ^{IE}.

We generally assume that we are tuning a policy using a simulator, which means that we can evaluate our policy using the actual (simulated) truth as we did in equation (3.22). Instead of writing the performance of the policy as $\overline{F}^{\pi}$, we can write it as $\overline{F}(\theta^{IE})$. Now we can write our tuning problem as

$$\max_{\theta^{IE}} \overline{F}(\theta^{IE}).$$

It should not be lost on us that the problem of designing a good policy for learning is itself a learning problem. An important difference between our tuning problem and the learning problem that we really want to solve is that we assume that we are using a fairly fast simulator to evaluate a policy.

If we do very accurate simulations, we can treat $\overline{F}(\theta^{IE})$ as if it were deterministic and unimodular (which means it has a single maximum), which makes it fairly easy to search. In chapter 10, we review a number of methods for efficiently searching for the maximum of a scalar function, even when the simulations are noisy.

Often, the problem of tuning a policy involves performing a one-dimensional search to find the best value of a single parameter such as θ^{IE}. In some settings (interval estimation is one of them) we have a rough idea of the magnitude of the tunable parameter. In the case of interval estimation, θ^{IE} is unitless, so we know it is on the order of 1, but might be 0.1 or perhaps as large as 10. However, the tunable parameter for the Boltzmann policy has units, and as a result might be anywhere from 10^{-5} to 10^5. For this reason, it is a good idea to first search across orders of magnitude, and then refine the search over smaller ranges.

To appreciate the importance of tuning, we tuned two policies on a library of nine problems. The first policy was interval estimation, while the second is a class of upper confidence bounding policy called UCB-E, which is given by

$$X^{UCBE,n}(\theta^{UCBE}) = \arg\max_x \left(\bar{\mu}_x^n + \sqrt{\frac{\theta^{UCBE}}{N_x^n}} \right). \tag{3.25}$$

where N_x^n is the number of times we have observed alternative x. We note that while the tunable parameter θ^{IE} is unitless, the parameter for the UCBE policy, θ^{UCBE}, is not.

Table 3.4 shows the tuned values of these parameters for each of nine test problems. We see that the variation in θ^{IE}, which ranges from 10 to 10^{-2}, is much narrower than the variation of θ^{UCBE}, which ranges from 10^3 to 10^{-4}. We also note that tuning matters; the performance of a policy can vary quite a bit as the tunable parameter is adjusted.

Problem class	IE	UCBE
P1	0.0994	2571
P2	0.0150	0.319
P3	0.0187	1.591
P4	0.0109	6.835
P5	0.2694	.00036
P6	1.1970	1.329
P7	0.8991	21.21
P8	0.9989	0.00016
P9	0.2086	0.001476

Table 3.4 Illustration of tuned parameters for interval estimation (IE) and the UCB policy UCB-E over nine different problems.

We are going to assume that we always have access to a simulator for the purpose of tuning our policy. However, this presumes that our simulator captures the behavior of the real environment in which we are doing learning. Needless to say, this will introduce errors.

There are settings where simulators are simply not available. This means that we have to do tuning in the field, in the same expensive environment where we also have to do learning. As of this writing, policy tuning in the field remains an open research question.

3.8 EXTENSIONS OF THE RANKING AND SELECTION PROBLEM

Our basic ranking and selection problem, characterized by a discrete set of alternatives $\mathcal{X} = \{x_1, \ldots, x_M\}$ and a lookup table belief model, can be extended in a number of ways to produce an almost unlimited number of problem variations. Below we list some of the more popular variations.

Decisions There are a number of different characterizations of decisions:

- Binary $\mathcal{X} = \{0, 1\}$, which arises in A/B testing of websites (old versus new), or stopping problems (continue/stop).
- Finite set, where $\mathcal{X} = \{x_1, \ldots, x_M\}$, which is the setting we have focused on above.
- Subsets, where we have K items (such as members of a basketball team), where we have to choose 5 players. If $K = 10$, some elements of our set might look like:

$$(0, 1, 1, 0, 0, 1, 0, 1, 1, 0)$$
$$(0, 1, 0, 0, 0, 1, 1, 0, 1, 0)$$
$$(1, 0, 1, 0, 0, 1, 0, 1, 0, 1)$$
$$(0, 0, 1, 1, 0, 0, 1, 1, 1, 0)$$
$$(1, 0, 1, 0, 1, 1, 0, 0, 1, 0)$$

  The distinguishing feature of subset selection problems is that the number of alternatives may be extremely large.
- Continuous scalar, as in $\mathcal{X} = [a, b]$. This might describe the price of a product, concentration of an additive or dosage of a drug.
- Vector-valued continuous, which we might write as $x = (x_1, \ldots, x_K)$ where each x_k is a continuous scalar.
- Vector valued discrete, where each x_k has to be 0 or 1. This is similar to the subset selection problem, except that it may be limited by more complex constraints.
- Multiattribute. We might be choosing among people, who are each characterized by a number of attributes (gender, age, weight, ethnicity, education). Alternative, we might need to choose to target a particular population for an ad campaign, where a population group might be characterized by country or region, the device they are using (laptop or smartphone), their gender and age. The number of attributes can be quite large, but we might be able to exploit a hierarchical structure.

Types of noise Above we assumed that our random noise W was normally distributed, but we have a library of distributions to choose from, including

Unknown distribution - We may start with the realization that we simply do not know the distribution. For example, we may not even know if the proper dosage is .01mg, .1, mg, 1 mg, 10 mg.

Binomial - This might describe settings where the outcome is a success or failure, or whether a drug is judged to be a success (allowing us to go to market) or not.

Exponential family - This includes a number of distributions such as normal, exponential, log-normal, gamma, chi-squared, beta, Bernoulli, Poisson, and geometric.

Uniform distributions - We may know that a random variable is bounded. For example, the energy generated by a 2mw wind turbine may be reasonably approximated as a uniform distribution between 0 and 2mw.

Heavy-tailed distributions - This might include the Cauchy distribution (which has infinite variance) or spikes, as occurs with electricity prices, which can jump from $20 per megawatt-hour to $300 per mwh in a matter of minutes, and then drop back.

Mixture distributions - We randomly choose an observation from one of two distributions, one with a relatively smaller variance, and one with an extremely high variance.

Bursts - Bursts can arise when the wind rises during a storm, a product briefly becomes popular through word of mouth, or a disease pops up and spreads before it is contained.

Rare events - These are events that may occur and which we have to prepare for, even though it is quite rare. Examples include major floods, a tree falling on your house, or being robbed.

Offline vs. online learning In this chapter we have considered only offline learning, where we are only interested in the performance of the decision after all experiments have been conducted. In chapter 5 we make the transition to online learning, where we wish to maximize the cumulative rewards that arise while we are learning. These two problem classes have evolved since the 1950's in very distinct communities, but are actually quite closely related.

Experimentation costs and budgets There are settings in the internet where we can try different alternatives quite quickly, often making it possible to consider very high dimensional choice problems (such as which movies to display on a users account). There are many other settings where experiments are quite expensive, sharply limiting the number of experiments. For example, a scientist may be able to test 20 or 30 compounds over the course of a summer before his budget runs out. A business might want to test market features of a product, but it may require several months to observe sales. A patient may require a month before we know how she responds to a new diabetes medication.

Switching cost It may be easy to switch from one design x to another x', as an internet retailer might do when testing different prices for a product. There are many settings, however, where there is a switching cost. For example, we may be taking samples from people to test for the presence of a disease in a location. Moving to another location involves a switching cost.

Transient performance We have assumed that the underlying truth μ is unknown but constant. However, the market response to a price may change as a result of decisions made by competitors; the performance of an athlete may improve as he gets more playing time; and the impact of a drug may drop as a disease acquires immunity.

Transient alternatives A standard assumption is that the set of alternatives $\mathcal{X}$ is static, but even this is not always the case. We can only try a restaurant if we can get a reservation; we can only learn about the performance of a storm sensor if there is a storm; and we can only test the efficacy of a drug for a particular medical condition if we can find a patient that meets our criteria.

Belief models We have illustrated our learning with a lookup table belief model, but these become clumsy when the number of alternatives is very large (or continuous). Later we are going to consider other classes of linear and nonlinear models.

Needless to say, the number of variations of learning problems is quite large.

3.9 BIBLIOGRAPHIC NOTES

Section 3.1 - We present here a standard Bayesian framework for ranking and selection: see e.g. Gupta & Miescke (1996) or Chick (2006). Our presentation is based on the measure-theoretic idea of random variables as functions on a space of outcomes or sample paths; although measure theory is far outside the scope of this book, interested readers are directed to Cinlar (2011), a definitive rigorous exposition of measure-theoretic probability.

Section 3.2 - The design of policies for taking observations (or experiments) of noisy functions has its roots in the 1950s and 1960s. It evolved originally under the umbrella of stochastic optimization over a continuous domain from the seminal paper of Robbins and Monro (Robbins & Monro 1951), but this literature did not focus on the issue of maximizing the information gained from each observation (there was more attention on asymptotic convergence than rate of convergence). The ranking and selection community evolved with a focus on the problem of finding the best out of a set of discrete alternatives; see D. R. Barr (1966) for an early review, and Fu (2002) for a more current review. The challenge of collecting information in an optimal way has its roots in DeGroot (1970), which appears to give the first presentation of optimal learning as a dynamic program (but without an algorithm). Interval estimation was introduced by Kaelbling (1993). The adaptation of interval estimation using Chernoff bounds was done by Streeter & Smith (2006). Epsilon-greedy is described in Sutton & Barto (1998), with an analysis of convergence properties given in Singh et al. (2000). There is an emerging area of research which uses the concept of upper confidence bounding (UCB), originally developed for online problems, in an offline setting. We introduce the idea of upper confidence bounding in chapter 5 for online ("bandit") problems. Drawing on this framework, Audibert & Bubeck (2010) present a frequentist approach to ranking and selection with finite alternatives, with provable guarantees.

PROBLEMS

3.1 We wish to find a good learning policy to solve the problem in Table 3.5 in an offline setting.

a) Briefly describe the epsilon-greedy policy, the Boltzmann policy and the interval-estimation policy. Evaluate each policy in terms of its ability to capture important characteristics of a good learning policy.

b) Define the expected opportunity cost (EOC). Describe in words how you would approximate the EOC (since computing it exactly is impossible) using Monte Carlo simulation.

c) Let ω^n be the index for the nth sample realization of the random observations. Give an expression for the EOC for some policy π and give a precise formula for the confidence interval for the true performance of a policy π.

Iteration	A	B	C
Prior (μ_x, β_x)	(7,.05)	(3,.02)	(4,.01)
1	9	-	-
2	-	-	6
3	-	1	-

Table 3.5 Three observations, for three alternatives, given a normally distributed belief, and assuming normally distributed observations.

3.2 This exercise requires that you test an exploration policy in MATLAB. You will need to download two files from the course website:

`http://optimallearning.princeton.edu/exercises/exploration.m`
`http://optimallearning.princeton.edu/exercises/explorationRun.m`

The MATLAB file `exploration.m` executes a pure exploration policy for a general ranking and selection problem. The file `explorationRun.m` creates the data for a problem with 10 alternatives, where it simulates 1000 truths and 50 samples per truth. The program `exploration.m` computes the average opportunity cost "o_cost" and the standard deviation "se_result."

a) Write out the meaning of "o_cost" mathematically using the notation we have been using in the course.

b) The standard deviation "se_result" is the standard deviation of what variable?

c) Construct a 95 percent confidence interval for the value of the exploration policy, and modify the code to produce this confidence interval.

3.3 In this exercise you have to implement the Boltzmann learning policy, which you should model after the `exploration.m` routine in the previous exercise.

a) Create a new file `boltzmann.m` which implements the Boltzmann learning policy. Keep in mind that you will have to introduce a tunable parameter ρ. The Boltzmann policy gives you the probability that you should sample an alternative. Imagine you have three alternatives, and the Boltzmann distribution gives you sampling probabilities of .2, .5 and .3. To sample from this distribution, generate a random number between 0 and 1. If this number is less than or equal to .2, you choose the first alternative; if it is between .2 and .7, choose the second alternative; otherwise choose the third. You will need to generalize this for an arbitrary number of alternatives.

b) Using $N = 5000$, vary θ over .001, .01, .1, 1.0, until it appears that you have found the range which bounds the best value of ρ. For each value of θ that you try, record it in a table and report the value of the policy and the standard error. Narrow your range until you begin getting results that are indistinguishable from each other.

c) Compare the performance of the best Boltzmann policy that you can find to a pure exploration policy that was developed in exercise 3.2.

3.4 Implement the interval estimation policy using the `exploration.m` routine provided in exercise 3.2.

a) Create a file called `ie.m` which implements the interval estimation policy. Again, you will need a tunable parameter z_α which you might call $zalpha$.

b) The optimal value of z_α will be in the range from 0 to 5. Search over this range, first in increments of 1.0, and then in smaller increments, until you again are unable to distinguish between values (continue using $N = 5000$).

c) Compare the performance of the best Boltzmann policy that you can find to a pure exploration policy that was developed in exercise 3.2.

3.5 This exercise builds on the exploration policy, Boltzmann policy and interval estimation policy that was implemented in exercises 3.2, 3.3 and 3.4.

a) Run each policy using $N = 5000, M = 50$ and report the confidence intervals, using your best estimate of the tunable parameters. Can you conclude that one policy is better than the other two? If so, which one? If not, use the fact that the standard deviation declines inversely with $\sqrt{N}$ (it also decreases inversely with $\sqrt{M}$, but we are going to hold M constant for our study). Use this to determine how small you need the standard error to be, and then how large N needs to be to produce a confidence interval that is small enough to conclude that one policy is best. You have may to repeat this exercise a few times for the numbers to settle down.

b) Now set $N = 1, M = 1$ and run each policy 10 times, reporting the actual outcome (noticed that you will not get a standard error in this case). This simulates the application of a learning policy in a specific situation, where you put it into practice (but we are going to pretend that you can replicate this 10 times). Record how often each policy discovers the best alternative (o_cost = 0). Comment on the likelihood that your best policy would outperform the other two policies on a single sample path.

3.6 You have to choose the best of three medications to treat a disease. The performance of each medication depends on the genetic characteristics of the individual. From the perspective of this medication, people can be divided into five genetic subgroups. If a doctor knew a patient's subgroup, he would know the medication he should use, but this information is not available. Lacking this information, the doctor has to resort to trial and error. Complicating the process is that measuring the performance of a medication involves a certain level of noise.

Table 3.6 gives the average performance of each type of medication on the five patient types, based on extensive prior records. The five patient types occur with equal probability in the population. The doctor will typically test a medication on a patient for one month, after which a blood test provides a measure of the performance of the

medication. But these experiments are not precise; the error between the experiment and the actual impact of the medication is normally distributed with a standard deviation of 2.2 (we assume this is constant for all medications and patient types).

	Patient type				
Medication	A	B	C	D	E
M1	6.2	7.3	5.4	7.2	5.4
M2	8.4	6.9	6.8	6.6	4.2
M3	5.2	5.8	6.3	5.5	3.7

Table 3.6 The performance of each type of medication for each type of patient.

a) What is the prior vector $\bar{\mu}^0$ that you would use for this problem, given the data in the table?

b) What is your prior probability distribution for the performance of medication M1? Note: It is *not* normally distributed.

c) Test each of the following policies below. You may use the MATLAB routines developed in the previous exercises, or perform the exercise manually with a budget of $N = 10$ observations. If you are using the MATLAB routines, use a budget of $N = 50$ observations.

   1) Pure exploitation.
   2) Boltzmann exploration, using scaling factor $\rho = 1$.
   3) Epsilon-greedy exploration, where the exploration probability is given by $1/n$.
   4) Interval estimation. Test the performance of $z_\alpha = 1.0, 2.0, 3.0$ and 4.0 and select the one that performs the best.

   For each policy, report the average performance based on 100 trials, and compute a 95 percent confidence interval.

CHAPTER 4

VALUE OF INFORMATION POLICIES

An important class of problems are those where experiments are expensive, and as a result the budget for the number of experiments is relatively small. For now, we are going to stay with the same types of problems we addressed in chapter 3, which is to say that we have a not-too-large set of discrete alternatives, described by a lookup table belief model. However, now we are going to assume that are experiments are expensive, which translates to relatively small budgets, which may be smaller (sometimes much smaller) than the number of alternatives.

At the same time, we are going to relax what are sometimes strict conditions on how much time we have to determine what to do next. In internet applications, we may have 50 milliseconds to determine what to do next. In the context of expensive decisions, we are not going to object to decision rules that might take seconds or minutes (or more) to compute.

Examples of expensive experiments include

- We may drill a well to evaluate the potential of the underground geology for producing oil or gas.
- We may need to observe a patient taking a new drug for several weeks to see how her body responds to the medication.
- A business may need to observe the sales of a product over a period of several weeks to evaluate the market response to a price.

Optimal Learning. By Warren B. Powell and Ilya O. Ryzhov

- A basketball coach needs to observe how well a team of five players performs over a course of several games.
- A scientist may require a day (or longer) to run a single experiment to assess the impact of a particular experimental design on the strength or conductivity of a material.

In the realm of high volume information (as is often the case in internet applications), we can often run a series of exploratory tests. When we face the problem of expensive experiments, then we find ourselves thinking carefully about the first experiment, before we have collected any information. However, it is often the case that we already know something about the properties of the problem.

For this reason, we are going to adopt a Bayesian modeling approach so that we have a mechanism for incorporating prior knowledge, which tends to almost always be available for these complex problems. Priors can come from a variety of sources:

Prior experiments We may have already run experiments, perhaps in other settings.

Research literature Others may have worked on this or related problems.

Domain expertise Experts may offer initial insights, often drawn from experiences with similar settings (such as the response of other patients to a drug, or the market response to the price of a textbook).

Computer simulations We may learn from numerical models.

A valuable class of policies for expensive experiments is based on maximizing the value of information. In chapter 3, we optimized policies to maximize $\overline{F}^{\pi}$, but this simply means doing the best we can with a particular heuristic. In this chapter, we are going to design policies that are specifically designed to maximize the performance from *each* experiment so that we obtain the fastest learning rate possible.

4.1 THE VALUE OF INFORMATION

There are different ways of estimating the value of information, but one strategy, called the *knowledge gradient*, works as follows. Assume that we have a finite number of discrete alternatives with independent, normally distributed beliefs (the same problem we considered in chapter 3). After n experiments, we let $\bar{\mu}^n$ be our vector of estimates of means and β^n our vector of precisions (inverse variances). We represent our belief state as $S^n = (\bar{\mu}^n_x, \beta^n_x)_{x\in\mathcal{X}}$. If we stop measuring now, we would pick the best option, which we represent by

$$x^n = \arg\max_{x\in\mathcal{X}} \bar{\mu}^n_x.$$

The value of being in state S^n is then given by

$$V^n(S^n) = \bar{\mu}^n_{x^n}. \tag{4.1}$$

Now let $S^{n+1}(x)$ be the next state if we choose to measure $x^n = x$ right now, allowing us to observe $W_{x^n}^{n+1}$. This allows us to update $\bar{\mu}_x^n$ and β_x^n, giving us an estimate $\bar{\mu}_x^{n+1}$ for the mean and β_x^{n+1} for the precision (using equations (3.2) and (3.3)). Given that we choose to measure $x = x^n$, we transition to a new belief state $S^{n+1}(x)$, and the value of being in this state is now given by

$$V^{n+1}(S^{n+1}(x)) = \max_{x' \in \mathcal{X}} \bar{\mu}_{x'}^{n+1}(x).$$

Let $\bar{\mu}_{x'}^{n+1}(x)$ be the updated estimate of $\bar{\mu}_{x'}^n$ if we run experiment x' and observe $W_{x'}^{n+1}$.

$$\bar{\mu}_{x'}^{n+1}(x) = \begin{cases} \frac{\beta_{x'}^n \bar{\mu}_{x'}^n + \beta^W W_{x'}^{n+1}}{\beta_{x'}^n + \beta^W} & \text{If } x' = x, \\ \bar{\mu}_{x'}^n & \text{Otherwise} \end{cases}. \tag{4.2}$$

At time n when we have chosen $x = x^n$ as our next experiment, but before we have observed W_x^{n+1}, the estimate $\bar{\mu}_{x'}^{n+1}(x)$ is a random variable.

We would like to choose x at iteration n which maximizes the expected value of $V^{n+1}(S^{n+1}(x))$. At time n, we write this expectation as

$$\mathbb{E}\{V^{n+1}(S^{n+1}(x))|S^n\} = \mathbb{E}_\mu \mathbb{E}_{W|\mu}\{V^{n+1}(S^{n+1}(x))|S^n\},$$

where the right hand side brings out that there are two random variables: the truth μ (which is uncertain in a Bayesian belief model) and the outcome $W_x^{n+1} = \mu_x + \epsilon^{n+1}$, which depends on the unknown truth μ. We want to choose an experiment x that maximizes $\mathbb{E}\{V^{n+1}(S^{n+1}(x))|S^n\}$, but instead of writing our objective in terms of maximizing the value from an experiment, we are going to write it equivalent as maximizing the incremental value from the experiment, which is given by

$$\begin{aligned} \nu_x^{KG,n} &= \mathbb{E}_\mu \mathbb{E}_{W|\mu}\left\{V^{n+1}(S^{n+1}(x)) - V^n(S^n)|S^n\right\} \\ &= \mathbb{E}_\mu \mathbb{E}_{W|\mu}\left\{V^{n+1}(S^{n+1}(x))|S^n\right\} - V^n(S^n). \end{aligned} \tag{4.3}$$

Keep in mind that the state S^n is our belief about μ after n experiments, which is that $\mu \sim N(\bar{\mu}^n, \beta^n)$. Given S^n (that is, given what we know at time n), the value $V^n(S^n)$ is calculated just using our current estimates $\bar{\mu}^n$, as is done in equation (4.1). Thus, at time n, $V^n(S^n)$ is a number, which is why $\mathbb{E}\{V^n(S^n)|S^n\} = V^n(S^n)$. However, $V^{n+1}(S^{n+1}(x))$ is a random variable since it depends on the outcome of the $n+1st$ experiment W^{n+1}.

The right hand side of (4.3) can be viewed as the derivative (or gradient) of $V^n(S^n)$ with respect to the experiment x. Thus, we are choosing our experiment to maximize the gradient with respect to the knowledge gained from the experiment. For this reason we refer to $\nu_x^{KG,n}$ as the *knowledge gradient*. We write the knowledge gradient policy using

$$X^{KG,n} = \arg\max_{x \in \mathcal{X}} \nu_x^{KG,n}. \tag{4.4}$$

The knowledge gradient, $\nu^{KG,n}$, is the amount by which the solution improves if we choose to measure alternative x. This is illustrated in Figure 4.1, where the estimated mean of choice 4 is best, and we need to find the value from measuring choice 5. The

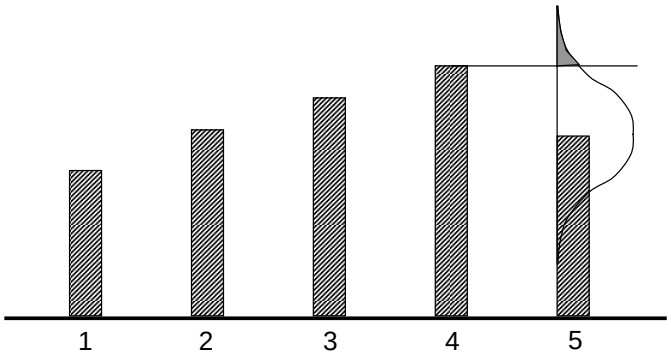


Figure 4.1 Illustration of the knowledge gradient if we were to measure choice 5.

estimated mean of choice 5 will move up or down according to a normal distribution (we assume with mean 0). The solid area under the curve that exceeds the estimate for choice 4 is the probability that measuring 5 will produce a value that is better than the current best, which means that V^{n+1} will increase. The knowledge gradient is the expected amount by which it will increase (we receive a value of 0 if it does not go up).

4.2 THE KNOWLEDGE GRADIENT FOR INDEPENDENT BELIEFS

For the case of independent normally distributed beliefs, the knowledge gradient is particularly easy to compute. When independence holds, we only change our beliefs about one alternative in every time period. Suppose that we are at time n, with estimates $\bar{\mu}_x^n$ of the true values μ_x, after which we choose to measure a particular alternative x^n. Then, we will have $\bar{\mu}_x^{n+1} = \bar{\mu}_x^n$ for $x \neq x^n$. Furthermore, for $x = x^n$, while the exact value of $\bar{\mu}_x^{n+1}$ is still unknown at time n, there is a very simple expression for the *conditional* distribution of $\bar{\mu}_x^{n+1}$ given the time-n beliefs. We are able to write

$$\bar{\mu}_x^{n+1} \sim \mathcal{N}\left(\bar{\mu}_x^n, \tilde{\sigma}_x^{2,n}\right) \tag{4.5}$$

where $\tilde{\sigma}_x^{2,n}$ is conditional change in the variance (which we first saw in chapter 2) given by

$$\begin{aligned} \tilde{\sigma}_x^{2,n} &= \mathit{Var}^n[\bar{\mu}_x^{n+1}] \\ &= \mathit{Var}^n[\bar{\mu}_x^{n+1} - \bar{\mu}_x^n]. \end{aligned} \tag{4.6}$$

We can think of $\tilde{\sigma}_x^{2,n}$ as the variance of the change in the estimate, or the conditional variance of $\bar{\mu}_x^{n+1}$ (given what we know at time n).

The distribution in (4.5) is known as the *predictive distribution* of $\bar{\mu}_x^{n+1}$, because it represents our best prediction of the results of our next observation before the observation actually occurs.

Below we provide the calculations required to compute the knowledge gradient, and follow this presentation with a discussion of some properties of this policy. The full derivation of the knowledge gradient policy is deferred to Section 4.12.1. For now, it is enough to keep in mind that (4.3) involves computing an expected value over the predictive distribution.

4.2.1 Computation

For the case of independent normally distributed beliefs, the knowledge gradient is particularly easy to compute. Recall that the precision is simply the inverse of the variance, which is given by $\bar{\sigma}_x^{2,n}$. Making the transition from precisions to variances, let σ_W^2 be the variance of our experiment W. The updating formula for the variance of our belief $\bar{\sigma}_x^{2,n}$, assuming we measure $x^n = x$, is given by

$$\begin{aligned}\bar{\sigma}_x^{2,n} &= \left((\bar{\sigma}_x^{2,n-1})^{-1} + (\sigma_W^2)^{-1}\right)^{-1} \\ &= \frac{\bar{\sigma}_x^{2,n-1}}{1 + \bar{\sigma}_x^{2,n-1}/\sigma_W^2}. \end{aligned} \tag{4.7}$$

Now let $\mathit{Var}^n(\cdot)$ be the variance of a random variable given what we know about the first n experiments. For example, $\mathit{Var}^n \bar{\mu}_x^n = 0$ since, given the first n experiments, $\bar{\mu}_x^n$ is deterministic. Next we compute the change in the variance in our belief about $\bar{\mu}^{n+1}$ given $\bar{\mu}^n$, given by

$$\tilde{\sigma}_x^{2,n} = \mathit{Var}^n[\bar{\mu}_x^{n+1} - \bar{\mu}_x^n].$$

We need to remember that given what we know at iteration n, which means given $\bar{\mu}^n$, the only reason that $\bar{\mu}^{n+1}$ is random (that is, with a variance that is not equal to zero) is because we have not yet observed W^{n+1}. As in Chapter 2, after some derivation, we can show that

$$\tilde{\sigma}_x^{2,n} = \bar{\sigma}_x^{2,n} - \bar{\sigma}_x^{2,n+1} \tag{4.8}$$

$$= \frac{\bar{\sigma}_x^{2,n}}{1 + \sigma_W^2/\bar{\sigma}_x^{2,n}} \tag{4.9}$$

$$= (\beta_x^n)^{-1} - (\beta_x^n + \beta^W)^{-1}. \tag{4.10}$$

It is useful to compare the updating equation for the variance (4.7) with the change in the variance in (4.9). The formulas have a surprising symmetry to them. Equation (4.10) gives the expression in terms of the precisions.

We then compute ζ_x^n which is given by

$$\zeta_x^n = -\left|\frac{\bar{\mu}_x^n - \max_{x' \neq x} \bar{\mu}_{x'}^n}{\tilde{\sigma}_x^n}\right|. \tag{4.11}$$

ζ_x^n is the *normalized influence* of decision x. It is the number of standard deviations from the current estimate of the value of decision x, given by $\bar{\mu}_x^n$, and the best alternative other than decision x. We always need to keep in mind that the value of information lies in its ability to change our decision (we first saw this in our decision tree illustration in section 1.6). So, we are always comparing the value of a choice to

the best of all the other alternatives. The quantity ζ_x^n captures the distance between a choice and the next best alternative, measured in units of standard deviations of the change resulting from an experiment.

We next compute

$$f(\zeta) = \zeta\Phi(\zeta) + \phi(\zeta), \tag{4.12}$$

where $\Phi(\zeta)$ and $\phi(\zeta)$ are, respectively, the cumulative standard normal distribution and the standard normal density. That is,

$$\phi(\zeta) = \frac{1}{\sqrt{2\pi}} e^{-\frac{\zeta^2}{2}},$$

and

$$\Phi(\zeta) = \int_{-\infty}^{\zeta} \phi(x)dx.$$

The density $\phi(\zeta)$ is, of course, quite easy to compute. The cumulative distribution $\Phi(\zeta)$ cannot be calculated analytically, but very accurate approximations are easily available. For example, MATLAB provides the function `normcdf`(x, μ, σ), while Excel provides `NORMSDIST`(ζ). Searching the Internet for "calculate cumulative normal distribution" will also turn up analytical approximations of the cumulative normal distribution.

Finally, the knowledge gradient is given by

$$\nu_x^{KG,n} = \tilde{\sigma}_x^n f(\zeta_x^n). \tag{4.13}$$

Table 4.1 illustrates the calculations for a problem with five choices. The priors $\bar{\mu}^n$ are shown in the second column, followed by the prior precision. The precision of the experiment is $\beta^W = 1$.

Choice	$\bar{\mu}^n$	β^n	β^{n+1}	$\tilde{\sigma}$	$max'_x x'$	ζ	$f(\zeta)$	ν_x^{KG}
1	20.0	0.0625	1.0625	3.8806	28	-2.0616	0.0072	0.0279
2	22.0	0.1111	1.1111	2.8460	28	-2.1082	0.0063	0.0180
3	24.0	0.0400	1.0400	4.9029	28	-0.8158	0.1169	0.5731
4	26.0	0.1111	1.1111	2.8460	28	-0.7027	0.1422	0.4048
5	28.0	0.0625	1.0625	3.8806	26	-0.5154	0.1931	0.7493

Table 4.1 Calculations illustrating the knowledge gradient index

Interestingly, the knowledge gradient formula in (4.13) is symmetric. This means that, if we are looking for the alternative with the lowest value (rather than the highest), we still have

$$\mathbb{E}\left[\min_{x'} \bar{\mu}_{x'}^n - \min_{x'} \bar{\mu}_{x'}^{n+1} \mid S^n, x^n = x\right] = \tilde{\sigma}_x^n f\left(\zeta_x^n\right),$$

with the only difference being that $\max_{x' \neq x} \bar{\mu}^n_{x'}$ is replaced by $\min_{x' \neq x} \bar{\mu}^n_{x'}$ in the definition of ζ^n_x. This symmetry is a consequence of our choice of a Gaussian learning model with Gaussian observations. Intuitively, the normal density is symmetric about its mean, and thus, whether we are looking for the largest or the smallest normal value, the computation consists of integrating over the tail probability of some normal distribution. This does not mean that alternative x has the exact same KG factor in both cases (we are changing the definition of ζ^n_x), but it does mean that the KG formula retains the same basic form and intuitive interpretation. Later on, we show that this does not hold when the learning model is not Gaussian.

4.2.2 Some properties of the knowledge gradient

Recall that we provided a mathematical framework for an optimal learning policy in Section 3.2.2. It is important to keep in mind that the knowledge gradient policy is not optimal, in that it is not guaranteed to be the best possible policy for collecting information for any budget N. But for ranking and selection problems, the knowledge gradient policy has some nice properties. These include

- Property 1: The knowledge gradient is always positive, $\nu^{KG,n}_x \geq 0$ for all x. Thus, if the knowledge gradient of an alternative is zero, that means we won't measure it.
- Property 2: The knowledge gradient policy is optimal (by construction) if we are going to make exactly one experiment.
- Property 3: If there are only two choices, the knowledge gradient policy is optimal for any experiment budget N.
- Property 4: If N is our experimental budget, the knowledge gradient policy is guaranteed to find the best alternative as N is allowed to be big enough. That is, if x^N is the solution we obtain after N experiments, and

  $$x^* = \arg\max \mu_x$$

  is the true best alternative, then $x^N \to x^*$ as $N \to \infty$. This property is known as asymptotic optimality.
- Property 5: There are many heuristic policies that are asymptotically optimal (for example, pure exploration, mixed exploration-exploitation, epsilon-greedy exploration and Boltzmann exploration). But none of these heuristic policies is myopically optimal. The knowledge gradient policy is the only pure policy (an alternative term would be to say it is the only stationary policy) that is both myopically and asymptotically optimal.
- Property 6: The knowledge gradient has no tunable algorithmic parameters. Heuristics such as the Boltzmann policy (θ^B in equation (3.13)) and interval estimation (θ^{IE} in equation (3.17)) have tunable algorithmic parameters. The knowledge gradient has no such parameters, but as with all Bayesian methods, assumes we have a prior. Later, we demonstrate settings where we do have to introduce tunable parameters.

The knowledge gradient is not an optimal policy for collecting information, but these properties suggest that it is generally going to work well. There are situations

where it can work poorly, as we demonstrate in section 4.3 below. In addition, it is more complicated to compute, and for more general belief models, it can be much more complicated to compute. This is a general property of all value-of-information policies.

4.3 THE VALUE OF INFORMATION AND THE S-CURVE EFFECT

The knowledge gradient computes the marginal value of information. It is easy to expect that the marginal value of information declines as we do more experiments, but this is not always the case. Below, we show how to identify when this property is violated, in which case the value of a single experiment can be quite low. When this happens, the knowledge gradient as described above does not add much value. We then describe some methods for overcoming this limitation by modeling what happens when you consider repeating the same experiment multiple times.

4.3.1 The S-curve

What if we perform n_x observations of alternative x, rather than just a single experiment? In this section, we derive the value of n_x experiments to study the marginal value of information. Note that this can be viewed as finding the value of a single experiment with precision $n_x\beta^W$, so below we view this as a single, more accurate experiment.

As before, let $\bar{\mu}_x^0$ and β_x^0 be the mean and precision of our prior distribution of belief about μ_x. Now let $\bar{\mu}_x^1$ and β_x^1 be the updated mean and precision after measuring alternative x a total of n_x times in a row. As before, we let $\beta^W = 1/\sigma_W^2$ be the precision of a single experiment. This means that our updated precision after n_x observations of x is

$$\beta_x^1 = \beta_x^0 + n_x\beta^W.$$

In Section 2.3.1, we showed that

$$\tilde{\sigma}^{2,n} = \mathit{Var}^n[\bar{\mu}^{n+1} - \bar{\mu}^n],$$

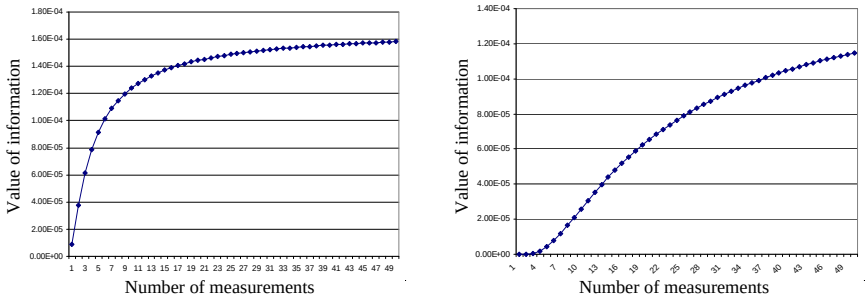
where Var^n is the conditional variance given what we know after n iterations. We are interested in the total variance reduction over n experiments. We denote this by $\tilde{\sigma}^{2,0}$, and calculate

$$\begin{aligned}\tilde{\sigma}^{2,0}(n_x) &= \bar{\sigma}^{2,0} - \bar{\sigma}^{2,1} \\ &= (\beta^0)^{-1} - (\beta^0 + n_x\beta^W)^{-1}.\end{aligned}$$

We next take advantage of the same steps we used to create equation (2.14) and write

$$\bar{\mu}_x^1 = \bar{\mu}_x^0 + \tilde{\sigma}_x^0(n_x)Z$$

where Z is a standard normal random variable, and where $\tilde{\sigma}_x^0(n_x) = \sqrt{\tilde{\sigma}_x^{2,0}(n_x)}$ is the standard deviation of the conditional change in the variance of $\bar{\mu}^1$ given that we make n_x observations.



4.2(a) 4.2(b)

Figure 4.2 Value of making n measures. In (a), the value of information is concave, while in (b) the value of information follows an S-curve.

We are now ready to calculate the value of our n_x experiments. Assume we are measuring a single alternative x, so $n_x > 0$ and $n_{x'} = 0$ for $x' \neq x$. Then we can write

$$\nu_x^{KG}(n_x) = \mathbb{E}\left[\max_{x'}(\bar{\mu}_{x'}^0 + \tilde{\sigma}_{x'}^0(n_{x'})Z_{x'})\right] - \max_{x'}\bar{\mu}_{x'}^0.$$

We can compute the value of n_x observations of alternative x using the knowledge gradient formula in equation (4.13),

$$\nu_x^{KG}(n_x) = \tilde{\sigma}_x^0(n_x) f\left(\frac{\bar{\mu}_x^0 - \max_{x' \neq x}\bar{\mu}_{x'}^0}{\tilde{\sigma}_x^0(n_x)}\right),$$

where $f(\zeta)$ is given in equation (4.12).

Now we have what we need to study some properties of the value of information. Consider a problem where $\sigma_W = 1.0$, $\sigma^0 = 1.5$ and $\Delta = \mu_x - \max_{x' \neq x}\mu_{x'} = 5$. Figure 4.2(a) shows the value of n experiments as n ranges from 1 to 50. This plot shows that the value of information is concave, as we might expect. Each additional experiment brings value, but less than the previous experiment, a behavior that seems quite intuitive.

Figure 4.2(b), however, gives the same plot for a problem where $\sigma_W = 2.5$. Note that when the experimental noise increases, the value of information forms an S-curve, with a very small value of information from the first few experiments, but then rising.

The S-curve behavior arises when a single experiment simply contains too little information to change a decision (and remember: the value of information is from its ability to change a decision). This behavior actually arises fairly frequently, and always arises when the outcome of an experiment is one of two outcomes such as success or failure (these are known as binomial outcomes). It is like introducing a new player to a team and then observing whether the team wins or not. You learn almost nothing about the new player purely from the outcome of whether or not the team won. In fact, it is entirely possible that the knowledge gradient may be some tiny number such as 10^{-10}.

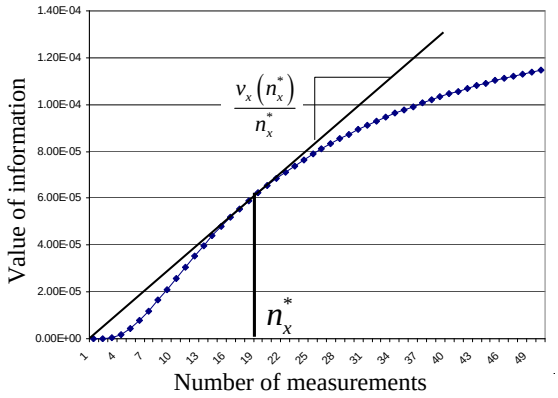


Figure 4.3 The KG(*) policy, which maximizes the average value of a series of experiments of a single alternative.

4.3.2 The KG(*) policy

A variant of the KG policy is called the KG(*) policy, which finds the number of experiments n_x^* which produces the highest *average* value of each observation, computed using

$$n_x^* = \arg\max_{n_x>0} \frac{\nu_x^{KG}(n_x)}{n_x}. \tag{4.14}$$

Figure 4.3 illustrates this policy. If the value of information is concave, then $n_x^* = 1$. If the function is non-concave, then it is very easy to find the value of n_x^* that solves equation (4.14).

4.3.3 A tunable lookahead policy

The KG(*) policy implicitly assumes that our experimental budget is large enough to sample alternative x roughly n_x^* times. There are many applications where this is just not going to be true. We can mimic the basic idea of KG(*), but replace n_x^* with a tunable parameter. Begin by defining

$\nu^{KG,n}(\theta^{KG}) =$ The knowledge gradient computed using a precision $\beta^1 = \beta^0 + \theta^{KG}\beta^W$.

We then define our tunable KG policy as

$$X^{KG}(\theta^{KG}) = \arg\max_x \nu_x^{KG,n}(\theta^{KG})$$

We now have to tune our policy to find the best value of θ^{KG} (see section 3.7 for our discussion on tuning). Here is where our policy adapts to our learning budget, as well as other problem characteristics that we choose to model.

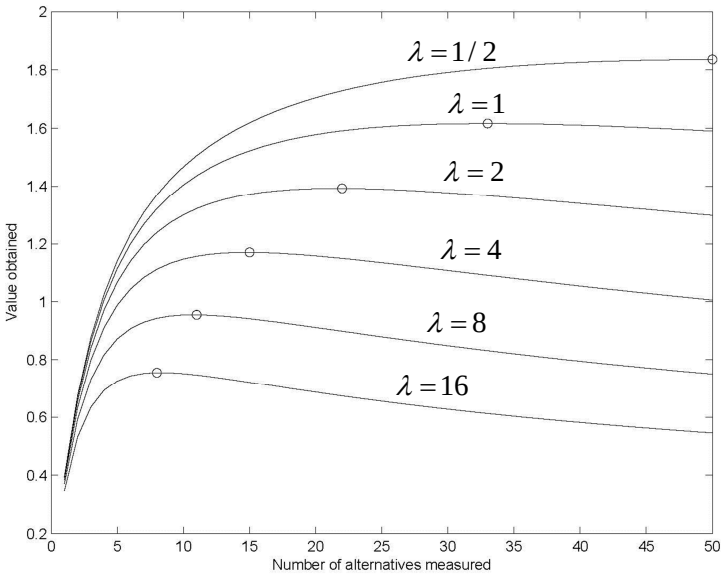


Figure 4.4 The value of spreading a budget of 50 experiments over $M \leq 50$ alternatives (from Frazier & Powell 2010).

4.3.4 The problem of too many choices

There are times when we simply have too many alternatives to evaluate. Imagine, for example, that we have 50 players to evaluate when forming a basketball team. We can let each take n_x shots and evaluate their shooting ability. Imagine that we have enough time to let all of them take a total of N shots. It is easy to imagine that if we have 50 players to consider, but only have enough time to let them take a total of 50 shots, we are not going to learn anything if each player gets to take a single shot.

We often find ourselves with too many choices. Consider the last time you visited a supermarket, or had to find a restaurant in a large city (even if you live there). Ultimately, we have to find a way to limit our choice set, and then find the best within that set.

Figure 4.4 illustrates the value of spreading a budget of 50 experiments uniformly over $N \leq 50$ alternatives. If the experimental noise λ is small (in the figure, this corresponds to $\lambda = 1/2$), we do the best if we can observe all 50 alternatives exactly once, after which we pick what appears to be the best choice. Since our evaluation is very accurate, we do the best when we have the most choices. As the experimental noise increases, we get more value if we focus our experimental budget over a subset of the alternatives. For example, if $\lambda = 16$, we do best if we arbitrarily choose eight alternatives out of our population of 50, and then focus on finding the best among these eight.

This behavior can help explain why some choices tend to be biased in favor of cosmetic differences. If you have to choose the best baseball player, there is a bias toward choosing tall, strong athletes, since these are attributes that are easy to identify

and help to narrow the field. People are biased toward brands that are well known, and companies tend to work with established vendors. Simply landing near the top of an Internet search can also be a way of pruning a set of choices. However it is done, there are many examples where there is a need to use an initial filter to reduce the set of choices to one that can be evaluated using a fixed experimental budget.

4.4 THE FOUR DISTRIBUTIONS OF LEARNING

Perhaps one of the most subtle aspects of learning is understanding the different types of uncertainty, which depend on when you are asking the question. There are four different distributions that arise in the learning process. We illustrate these in the context of learning problems where all random variables are represented by normal distributions.

1) The *prior distribution* $N(\bar{\mu}_x^0, \bar{\sigma}_x^{2,0})$, which gives our initial distribution of belief about the mean.

2) The *sampling distribution* of the random variable W, which is given by $N(\mu, \sigma_W^2)$, where μ is the true mean and σ_W^2 is the sample variance.

3) The *posterior distribution* (after n experiments), given by $N(\bar{\mu}_x^n, \bar{\sigma}_x^{2,n})$, which reflects the noise in the experimental outcome W. Here, we are modeling the distribution of μ after n experiments, but when we are at time $n = 0$ (that is, we have not run any experiments yet).

4) The conditional distribution of $\bar{\mu}_x^{n+1}$ given our beliefs at time n. This is also known as the *predictive distribution*. If we have seen $W^1, ..., W^n$, then $\bar{\mu}_x^{n+1}$ is random before we have observed W^{n+1}, which means that $\bar{\mu}_x^{n+1} \sim \mathcal{N}(\bar{\mu}_x^n, \tilde{\sigma}_x^{2,n})$.

An understanding of these distributions is useful because it helps to highlight the different types of uncertainties we are trying to resolve by collecting information.

4.5 KNOWLEDGE GRADIENT FOR CORRELATED BELIEFS

There are many problems where updating our belief about one alternative tells us something about other alternatives. Some examples include

- We are trying to find the best set of features for a laptop. We try one laptop with 12G of RAM, a 2.8 Ghz processor, a 14" screen and weighing 3.2 pounds. We then offer a second laptop with 8G of RAM, but everything else the same. A third laptop has 10G of RAM, a 3.2 Ghz processor, a solid state internal disk drive, a 13" screen and weighs 2.5 lbs (with a much higher price). We start by selling the first laptop, and find that we are getting sales higher than expected. Given the similarities with the second laptop, it is reasonable to assume that the sales of this laptop will also be higher than expected.

- Now we are trying to find the best price for our laptop. We start with an initial guess of the sales volume we anticipate for prices in $100 increments from $800 to $1500. We start at $1100, and sales are much lower than we expected. This would lower our beliefs about sales at $1000 and $1200.
- We are trying to find the best of several paths to use for routing a bus. Each time the bus finishes a trip, we record the start and end times, but not the times for individual components of the trip (the bus driver does not have the time for this). If the travel time for one path is higher than expected, this would increase our estimates for other paths that have shared links.
- We are estimating who has a disease. If a concentration of a disease is higher than expected in one part of the population, then we would expect that people who are nearby, or otherwise have a reason to interact, will also have a higher likelihood of having the disease.

Correlations are particularly important when the number of possible experiments is much larger than the experimental budget. The experiment design might be continuous (where in the United States should we test birds for bird flu), or there may simply be a very large number of choices (such as websites relevant to a particular issue). The number of choices to measure may be far larger than our budget to measure them in a reliable way.

We begin our discussion by reviewing the updated formulas for correlated beliefs, first presented in section 2.3.2. We then describe how to compute the knowledge gradient using correlated beliefs, and close with a discussion of the benefits.

4.5.1 Updating with correlated beliefs

As we pointed out in section 3.5, we can handle correlations with any policy if we use this structure to update beliefs *after* we have decided which experiment to make. However, the knowledge gradient is able to imbed our knowledge of correlated beliefs when we compute the value of information.

We assume that we have a covariance matrix (or function) that tells us how experiments of x and x' are correlated. If x is a scalar, we might assume that the covariance of μ_x and $\mu_{x'}$ is given by

$$Cov(x, x') = \sigma_x \sigma_{x'} e^{-\beta|x-x'|},$$

where σ_x and $\sigma_{x'}$ is the standard deviation of our belief about μ_x and $\mu_{x'}$, respectively. Here, β controls the relationship between μ_x and $\mu_{x'}$ as the distance between x and x' changes. Or, we just assume that there is a known covariance matrix Σ with element $\sigma_{xx'}$. For now, we continue to assume that the alternatives x are discrete, an assumption we relax later.

There is a way of updating our estimate of $\bar{\mu}^n$ which gives us a more convenient analytical form than what is given in Section 2.3.2. To simplify the algebra a bit, we let $\lambda^W = \sigma_W^2 = 1/\beta^W$. As we did in Section 4.2, we need the *change* in the variance of our belief due to the results of an experiment. Following the development

in Chapter 2, let $\Sigma^{n+1}(x)$ be the updated covariance matrix given that we have chosen to measure alternative x, and let $\tilde{\Sigma}^n(x)$ be the change in the covariance matrix due to evaluating x, which is given by

$$\tilde{\Sigma}^n(x) = \Sigma^n - \Sigma^{n+1} \tag{4.15}$$

$$= \frac{\Sigma^n e_x (e_x)^T \Sigma^n}{\Sigma^n_{xx} + \lambda^W}, \tag{4.16}$$

where e_x is a column vector of 0's with a 1 in the position corresponding to x. We note that (4.15) closely parallels (4.8) for the case of independent beliefs. Now define the column vector $\tilde{\sigma}^n(x)$, which gives the change in our belief about each alternative x' resulting from measuring alternative x. This is calculated using

$$\tilde{\sigma}^n(x) = \frac{\Sigma^n e_x}{\sqrt{\Sigma^n_{xx} + \lambda^W}}. \tag{4.17}$$

Keep in mind that $\tilde{\sigma}^n(x)$ is a vector with element $\tilde{\sigma}^n_i(x)$ for a given experiment $x = x^n$.

Also let $\tilde{\sigma}_i(x)$ be the component $(e_i)^T\tilde{\sigma}(x)$ of the vector $\tilde{\sigma}(x)$. Let $Var^n(\cdot)$ be the variance given what we know after n experiments. We note that if we measure alternative x^n, then

$$Var^n\left[W^{n+1}_{x^n} - \bar{\mu}^n_{x^n}\right] = Var^n\left[\mu_{x^n} + \varepsilon^{n+1} - \bar{\mu}^n_{x^n}\right] \tag{4.18}$$

$$= Var^n\left[\mu_{x^n} + \varepsilon^{n+1}\right] \tag{4.19}$$

$$= \Sigma^n_{x^n x^n} + \lambda^W. \tag{4.20}$$

We obtain equation (4.18) using $W^{n+1}_{x^n} = \mu_{x^n} + \varepsilon^{n+1}$, since $W^{n+1}_{x^n}$ is a noisy observation of the truth μ_{x^n} (which is random because of our Bayesian belief model). We drop $\bar{\mu}^n_{x^n}$ to obtain (4.18) because we are conditioning on what we know after n experiments, which means that $\bar{\mu}^n_{x^n}$ is deterministic (that is, it is just a number).

Next define the random variable Z^{n+1}

$$Z^{n+1} = \frac{(W^{n+1}_x - \bar{\mu}^n_x)}{\sqrt{Var^n\left[W^{n+1}_x - \bar{\mu}^n_x\right]}},$$

where $x = x^n$. The random variable W^{n+1}_x is normally distributed with mean $\bar{\mu}^n_x$ and variance $Var^n\left[W^{n+1}_x - \bar{\mu}^n_x\right] = Var^n\left[W^{n+1}_x\right]$, which means that Z^{n+1} is normally distributed with mean 0 and variance 1.

We use the notation $\bar{\mu}^{n+1}(x^n)$ to represent the updated vector of estimates of the mean $\bar{\mu}^n$, with element $\bar{\mu}^{n+1}_i(x^n)$ after running experiment x^n. The updating equation for correlated beliefs (first presented in section 2.3.2) is given by

$$\bar{\mu}^{n+1}(x^n) = \bar{\mu}^n + \frac{W^{n+1}_{x^n} - \bar{\mu}^n_{x^n}}{\lambda^W + \Sigma^n_{x^n x^n}} \Sigma^n e_{x^n}. \tag{4.21}$$

We can rewrite (4.21) as

$$\bar{\mu}^{n+1}(x^n) = \bar{\mu}^n_{x^n} + \tilde{\sigma}^n(x^n) Z^{n+1}, \tag{4.22}$$

where $\bar{\mu}^n_{x^n}$ and $\tilde{\sigma}^n(x^n)$ are vectors, while Z^{n+1} is a scalar.

Equation (4.22) nicely brings out the definition of $\tilde{\sigma}^n(x^n)$ as the vector of variances of the future estimates $\bar{\mu}^{n+1}_{x^n}$ given what we know at time n (which makes $\bar{\mu}^n$ deterministic). Essentially, even though a single experiment may change our beliefs about every alternative, the "randomness" in this change comes from a scalar observation, so the conditional distribution is expressed in terms of the scalar Z^{n+1}. This is a useful way of representing $\bar{\mu}^{n+1}_{x^n}$, especially for problems with correlated beliefs, as we see next.

4.5.2 Computing the knowledge gradient

We now face the challenge of computed the knowledge gradient. The knowledge gradient policy for correlated beliefs is computed using

$$\begin{aligned} X^{KG}(s) &= \arg\max_x \mathbb{E}\left[\max_i \bar{\mu}_i^{n+1}(x^n) \mid S^n = s, x^n = x\right] \\ &= \arg\max_x \mathbb{E}\left[\max_i \left(\bar{\mu}_i^n + \tilde{\sigma}_i^n(x^n) Z^{n+1}\right) \mid S^n, x^n = x\right]. \end{aligned} \tag{4.23}$$

where Z is a one-dimensional standard normal random variable. The problem with this expression is that the expectation is hard to compute. We encountered the same expectation when experiments are independent, but in this case we just have to do an easy computation involving the normal distribution. When the experiments are correlated, the calculation becomes more difficult.

There is a way to compute the expectation exactly. We start by defining

$$h(\bar{\mu}^n, \tilde{\sigma}^n(x)) = \mathbb{E}\left[\max_i \left(\bar{\mu}_i^n + \tilde{\sigma}_i^n(x^n) Z^{n+1}\right) \mid S^n, x^n = x\right]. \tag{4.24}$$

Substituting (4.24) into (4.23) gives us

$$X^{KG}(s) = \arg\max_x h(\bar{\mu}^n, \tilde{\sigma}^n(x)). \tag{4.25}$$

Now let $h(a, b) = \mathbb{E}\max_i(a_i + b_i Z)$, where $a = \bar{\mu}^n_i$, $b = \tilde{\sigma}^n_i(x^n)$ and Z is our standard normal deviate. Both a and b are M-dimensional vectors. We next sort the elements b_i so that $b_1 \leq b_2 \leq \ldots$ so that we get a sequence of lines with increasing slopes. If we plot the lines, we get a series of cuts similar to what is depicted in Figure 4.5. We see there are ranges for z over which the line for alternative 1 is higher than any other line; there is another range for z over which alternative 2 dominates. But there is no range over which alternative 3 dominates; in fact, alternative 3 is dominated by every other alternative for any value of z. What we need to do is to identify these dominated alternatives and drop them from further consideration.

To do this we start by finding the points where the lines intersect. If we consider the lines $a_i + b_i z$ and $a_{i+1} + b_{i+1} z$, we find they intersect at

$$z = c_i = \frac{a_i - a_{i+1}}{b_{i+1} - b_i}.$$

For the moment, we are going to assume that $b_{i+1} > b_i$ (that is, no ties). If $c_{i-1} < c_i < c_{i+1}$, then we can generally find a range for z over which a particular

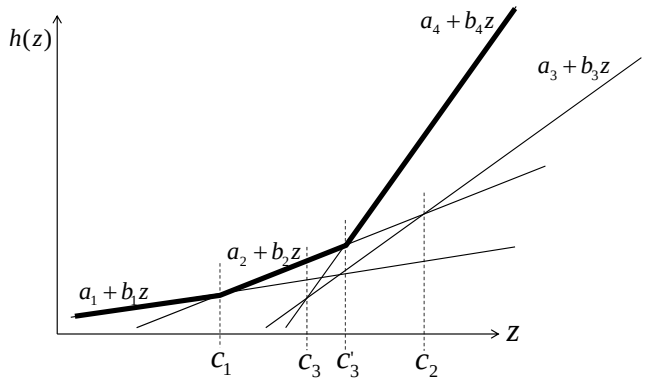


Figure 4.5 Regions of z over which different choices dominate. Choice 3 is always dominated.

choice dominates, as depicted in Figure 4.5. We can identify dominated alternatives such as alternative 3 in the figure when $c_{i+1} < c_i$. When this happens, we simply drop it from the set. Thus, instead of using c_2, we would use c_3' to capture the intersection between the lines for choices 2 and 4.

Once we have the sequence c_i in hand, we can compute (4.23) using

$$h(a,b) = \sum_{i=1}^{M} (b_{i+1} - b_i) f(-|c_i|),$$

where as before, $f(z) = z\Phi(z) + \phi(z)$. Of course, the summation has to be adjusted to skip any choices i that were found to be dominated.

Figure 4.6 illustrates the use of the correlated knowledge gradient algorithm when it is applied to a 2-dimensional surface represented using a lookup table with correlated beliefs to capture the smoothness of the function. The figure shows the function after one experiment (figure 4.6(a)), four experiments (b), five experiments (c), and six experiments (d). These plots help to illustrates two behaviors: First, how the knowledge gradient tends to pursue regions that are farthest from previous experiments (where the uncertainty is higher), and second, the ability to generalize the results of experiment using correlated beliefs.

4.5.3 The value of a KG policy with correlated beliefs

It is clear that we should use the structure of the correlations to update our beliefs after an experiment, leaving the question: how much value do we obtain by incorporating correlations within the knowledge gradient calculation? The knowledge gradient with independent beliefs is much easier to calculate, so it would be nice if we could use the simpler policy but still use correlations when updating beliefs.

Figure 4.7 addresses this question by plotting the knowledge gradient for two policies: the knowledge gradient using independent beliefs, and the knowledge gradient

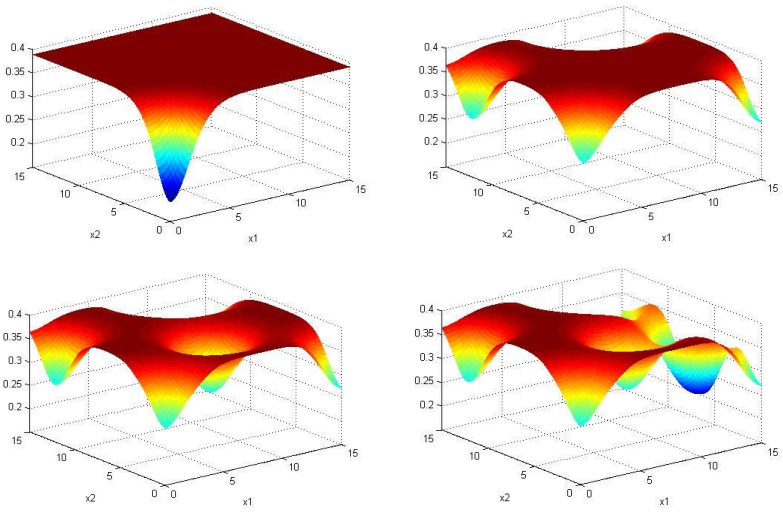


Figure 4.6 Evolution of belief about a 2-dimensional surface, showing the effect of experiments on our belief about the behavior of other experiments in the same proximity.

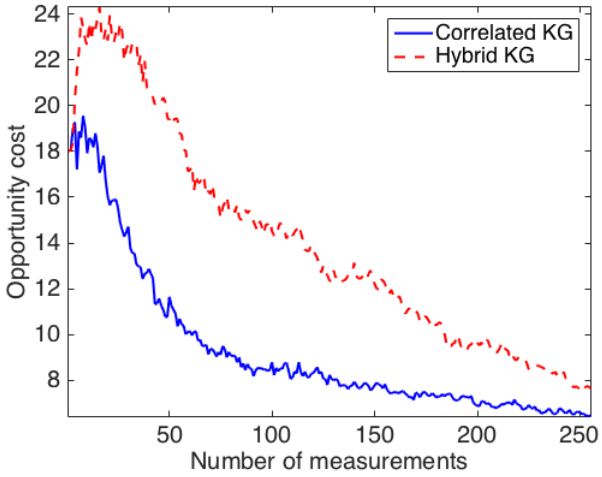


Figure 4.7 Performance of the knowledge gradient using independent or correlated beliefs when estimating the value of information.

incorporating correlated beliefs in the lookahead policy. Both policies are simulated using correlations when updating beliefs in the simulator, so this is a pure measure of the value of incorporating correlations when estimating the value of information. This plot suggests that there is considerable value capturing correlations within the policy, suggesting that the assumption of independence when estimating the value of information is not going to be efective.

Handling correlations between experiments has tremendous practical value. When we assumed independent experiments, it was necessary to measure each option at least once. There are applications where the number of potential experiments is far greater than the number of experiments that we can actually make. If we have information about correlations, we can find good designs even when the experimental budget is much smaller than the number of alternatives.

4.6 ANTICIPATORY VS. EXPERIENTIAL LEARNING

In section 3.5, we made the point that no matter what the policy does, we can capture complex belief models, along with complex dynamics of the information process, when we step forward from experiment n to $n+1$. Thus, all of the heuristic policies introduced in section 3.2 can capture correlated beliefs as part of the learning process.

Then, we just saw that we could include correlated beliefs in our model of the future as part of the process of deciding what experiment to run.

These are two very different ways of incorporating correlated beliefs (or, for that matter, any type of belief model). We distinguish these two modes using the following terms:

Anticipatory learning - This is the learning that we *anticipate* doing as part of our decision rule, when we are simulating decisions in the future.

Experiential learning - This is learning that we would do as we *experience* data from real observations from the field.

Anticipatory learning occurs within the decision function $X^\pi(S_t)$ when decisions require calculations that approximate in some way how new information may change our underlying belief. In our discussion of decision trees in Section 1.6, Section 1.6.1 presented a basic decision tree where outcomes did not change downstream probabilities. This is a case of a policy that does not use anticipatory learning. Sections 1.6.2 and 1.6.3 provided examples where observations of information were used to change downstream distributions within the decision tree used to make the decision. These sections provide examples of policies that use anticipatory learning.

Experiential learning is what happens after a decision has been made, and we then observe an actual outcome. Section 3.2.2 introduced the transition function $S_{t+1} = S^M(S_t, x_t, W_{t+1})$ which describes the updated belief state given an observation W_{t+1}. In some cases, W_{t+1} comes from a real physical system where the new information might be observed sales, or an observed response to a drug treatment. It is entirely possible that the distribution generating observed values of W_{t+1} is quite different than a distribution $f_W(w)$ that we are assuming for W_{t+1} within our policy $X^\pi(S_t)$. If this is the case, then it is natural to use observations of W_{t+1} to update our estimate of the density $f_W(w)$. This is experiential learning, and it is assumed to be part of the transition function $S^M(\cdot)$.

There are, however, many applications where $S^M(\cdot)$ represents a computer simulation, and W_{t+1} is generated from an assumed probability distribution $f_W(w)$. If

$f_W(w)$ is the distribution we use within our policy $X^\pi(S_t)$, then we are not actually learning anything from an observation W_{t+1} that comes from $f_W(w)$. This is quite common in many stochastic simulations, where the focus is on the uncertainty in W, and not in the uncertainty about the distribution $f_W(w)$ of W. An exception would be a simulation that is focusing explicitly on policies for collecting information. In such simulations, it is natural to make observations of W_{t+1} as we step forward using $S_{t+1} = S^M(S_t, x_t, W_{t+1})$ which are drawn from a truth that is unknown to the policy $X^\pi(S_t)$.

With these concepts in mind, we can describe three modeling strategies based on how new information is used.

1) No anticipatory or experiential learning. This is the default strategy in both deterministic and stochastic optimization, where we assume that observations of random variables are realizations from a known distribution, and as a result would not change our belief about the distribution. This situation is analogous to the "independent KG" curve in Figure 4.7.

2) Experiential learning without anticipatory learning. Here we use actual observations of random variables to update our beliefs, but we ignore the fact that we are going to do this when we make a decision. Learning is thus reduced to a statistical problem, with no optimization component. This is exactly the "hybrid KG" situation in Figure 4.7.

3) Experiential learning with anticipatory learning. This is where we use learning within the policy, as in equation (4.25), and in the transition function, as in equations (2.18)-(2.19). Such a situation corresponds to the "correlated KG" curve in Figure 4.7.

We ignore the combination of using anticipatory learning without experiential learning, because it does not make sense to anticipate the effect of observations on beliefs, but then not use real observations to update beliefs.

Anticipatory learning spans any policy (optimal or heuristic) which uses some representation of the uncertainty in our belief about the value of each choice. We did this heuristically in Chapter 3 with policies such as mixed exploitation-exploration, epsilon-greedy exploration, and Boltzmann exploration. In this chapter, the knowledge gradient, expected improvement and linear loss policies all represent methods which use some form of explicit learning within the learning policy.

4.7 THE KNOWLEDGE GRADIENT FOR SOME NON-GAUSSIAN DISTRIBUTIONS

It is important to remember that, while the definition of the knowledge gradient given in (4.3) is general, and can be written down for almost any learning problem, the KG formula given in (4.13) is geared specifically to the ranking and selection problem with independent alternatives and a normal-normal learning model (normally distributed prior, normally distributed experimental outcomes). This is not particularly surprising, because the formula uses the functions ϕ and Φ that characterize a

Gaussian distribution. We might expect other learning models to yield completely different formulas.

In this section, we give expressions for the marginal value of a single experiment for some of the non-Gaussian learning models presented in Chapter 2. They have certain notable differences from the normal-normal case, but all of them are based on the relationship between our beliefs about alternative x, and our beliefs about "the best alternative other than x." In all of these examples, we assume that the alternatives are independent, for the simple reason that there are no convenient Bayesian conjugate priors for any distribution other than normal.

4.7.1 The beta-Bernoulli model

We begin with the beta-Bernoulli model both for its simplicity which captures problems where outcomes are binary (e.g. success/failure, purchase a product or not) and where the uncertainty is the probability of success. This model will also highlight an important limitation of the basic knowledge gradient which tries to learn from a single outcome.

A popular application of the beta-bernoulli model arises in the setting of clinical trials. Our alternatives may correspond to different experimental medical treatments. A treatment can result in either success or failure, and we are interested in finding the treatment with the highest success probability. The challenge comes when we have a relatively small number of clinical trials that we can perform. We must decide which treatments to test in order to find the highest success probability most efficiently.

As in Chapter 2, we assume that the success probability ρ_x of treatment x follows a beta distribution with parameters α_x^0 and β_x^0. If we decide to test treatment x^n at time n, the result $W_{x^n}^{n+1}$ of the test is 1 with probability ρ_{x^n} and 0 with probability $1-\rho_{x^n}$. Our beliefs about the treatment are updated using the equations

$$\begin{aligned}
\alpha_x^{n+1} &= \begin{cases} \alpha_x^n + W_x^{n+1} & \text{if } x^n = x \\ \alpha_x^n & \text{otherwise,} \end{cases} \\
\beta_x^{n+1} &= \begin{cases} \beta_x^n + \left(1 - W_x^{n+1}\right) & \text{if } x^n = x \\ \beta_x^n & \text{otherwise.} \end{cases}
\end{aligned}$$

Our estimate of ρ_x given S^n is $\mathbb{E}\left(\rho_x \,|\, S^n\right) = \frac{\alpha_x^n}{\alpha_x^n + \beta_x^n}$. Recall that α_x^n roughly corresponds to the number of successes that we have observed in n trials, whereas β_x^n represents the number of failures. As usual, the KG factor is defined as

$$\nu_x^{KG,n} = \mathbb{E}\left[\max_{x'} \frac{\alpha_{x'}^{n+1}}{\alpha_{x'}^{n+1} + \beta_{x'}^{n+1}} - \max_{x'} \frac{\alpha_{x'}^n}{\alpha_{x'}^n + \beta_{x'}^n} \,\middle|\, S^n\right].$$

Unlike some of the other models discussed in this section, the predictive distribution in the beta-Bernoulli model is very simple. Given S^n and $x^n = x$, the conditional distribution of α_x^{n+1} is essentially Bernoulli. There are only two possible outcomes, whose probabilities are given by

$$\mathbb{P}\left(\alpha_x^{n+1} = \alpha_x^n + 1\right) = \frac{\alpha_x^n}{\alpha_x^n + \beta_x^n}, \qquad \mathbb{P}\left(\alpha_x^{n+1} = \alpha_x^n\right) = \frac{\beta_x^n}{\alpha_x^n + \beta_x^n}.$$

Choice	α^n	β^n	$\frac{\alpha^n}{\alpha^n+\beta^n+1}$	$\frac{\alpha^n}{\alpha^n+\beta^n}$	$\frac{\alpha^n+1}{\alpha^n+\beta^n+1}$	C^n	KG
1	1	13	0.0667	0.0714	0.1333	0.8874	0
2	2	11	0.1429	0.1538	0.2143	0.8874	0
3	1	20	0.0455	0.0476	0.0909	0.8874	0
4	67	333	0.1671	0.1675	0.1696	0.8874	0
5	268	34	0.8845	0.8874	0.8878	0.1675	0

Table 4.2 Calculations of the KG formula for the beta-Bernoulli model

The predictive distribution of β_x^{n+1} is also Bernoulli, but the probabilities are reversed. That is, $\mathbb{P}\left(\beta_x^{n+1} = \beta_x^n + 1\right)$ is now $\frac{\beta_x^n}{\alpha_x^n+\beta_x^n}$.

Letting $C_x^n = \max_{x' \neq x} \frac{\alpha_{x'}^n}{\alpha_{x'}^n+\beta_{x'}^n}$ as usual, we can derive the KG formula

$$\nu_x^{KG,n} = \begin{cases} \frac{\alpha_x^n}{\alpha_x^n+\beta_x^n}\left(\frac{\alpha_x^n+1}{\alpha_x^n+\beta_x^n+1} - C_x^n\right) & \text{if } \frac{\alpha_x^n}{\alpha_x^n+\beta_x^n} \leq C_x^n < \frac{\alpha_x^n+1}{\alpha_x^n+\beta_x^n+1} \\ \frac{\beta_x^n}{\alpha_x^n+\beta_x^n}\left(C_x^n - \frac{\alpha_x^n}{\alpha_x^n+\beta_x^n+1}\right) & \text{if } \frac{\alpha_x^n}{\alpha_x^n+\beta_x^n+1} \leq C_x^n < \frac{\alpha_x^n}{\alpha_x^n+\beta_x^n} \\ 0 & \text{otherwise.} \end{cases} \tag{4.26}$$

Observe that

$$\frac{\alpha_x^n}{\alpha_x^n+\beta_x^n+1} \leq \frac{\alpha_x^n}{\alpha_x^n+\beta_x^n} \leq \frac{\alpha_x^n+1}{\alpha_x^n+\beta_x^n+1},$$

and the KG factor depends on where C_x^n falls in relation to these quantities. In this case, we do not measure x if we believe ρ_x to be too low or too high. We will only benefit from measuring x if our beliefs about ρ_x are reasonably close to our beliefs about the other success probabilities. There is a certain symmetry to this formula (both low and high estimates result in knowledge gradients of zero), and in fact, it has the same symmetric quality of the KG formula for the normal-normal model. If our objective is to find the smallest success probability rather than the largest, (4.26) remains the same; the only change is that we replace the maximum in the definition of C_x^n by a minimum.

An unexpected consequence of this structure is that it is entirely possible for all the KG factors to be zero in the beta-Bernoulli problem. Table 4.2 illustrates one possible instance where this might happen. Intuitively, it can occur when we are already quite certain about the solution to the problem. In the problem shown in the table, it seems clear that alternative 5 is the best, with $268 + 34 = 302$ trials yielding a very high success probability. Among the other alternatives, there is some competition between 2 and 4, but neither is close to alternative 5. As a result, our beliefs about 2 and 4 are too low for one experiment to change their standing with respect to 5. Similarly, our beliefs about alternative 5 are too high for one experiment to change its standing with respect to any of the others. Thus, all the KG factors are zero, which essentially means that the KG method does not really care which alternative to measure (we can choose any one at random).

Of course, it is conceivable (though it may seem unlikely) that alternative 5 is not really the best. For instance, if we could measure alternative 2 several hundred times, we might find that it actually has a higher success probability, and we were

simply unlucky enough to observe 11 failures in the beginning. Unfortunately, the KG method only looks ahead one time step into the future, and thus is unable to consider this possibility. This hints at possible limitations of the knowledge gradient approach in a situation where the observations are discrete, similar to the S-curve effect we saw in Section 4.3.

4.7.2 The gamma-exponential model

Suppose that we have M exponentially distributed random variables with means $\lambda_1, ..., \lambda_M$, and we wish to find the one with the largest rate. For instance, we might be looking at a number of servers, with the goal of finding the one with the highest (fastest) service rate. Alternately, we might have a number of products, each with an exponential daily demand. In this setting, we might want to discover which product has the lowest demand (corresponding to the largest exponential rate), so that we might take this product out of production. A final example is the problem of network routing, where we have a number of possible routes for sending packets, and the objective is to find the route with the lowest ping time. In all of these problems, the observations (service time, daily demand, network latency) are positive, which means that the normal-normal model is not a good fit.

Instead, we use the gamma-exponential model (which we first saw in Section 2.6.1). We start by assuming that the rate λ_x of each alternative is uncertain and follows a gamma distribution with parameters a_x^0 and b_x^0. When we choose to measure alternative x^n at time n, we make a random observation $W_{x^n}^{n+1} \sim Exp\left(\lambda_{x^n}\right)$, and update our beliefs according to the equations

$$
\begin{aligned}
a_x^{n+1} &= \begin{cases} a_x^n + 1 & \text{if } x^n = x \\ a_x^n & \text{otherwise,} \end{cases} \\
b_x^{n+1} &= \begin{cases} b_x^n + W_x^{n+1} & \text{if } x^n = x \\ b_x^n & \text{otherwise.} \end{cases}
\end{aligned}
$$

We still have an independent ranking and selection problem, as in (3.2) and (3.3), but the specific updating mechanism for the alternative we measure is taken from the gamma-exponential model in Chapter 2.

Recall that, given the beliefs a_x^n and b_x^n, our estimate of λ_x is a_x^n/b_x^n, the mean of the gamma distribution. The KG factor of alternative x at time n is now given by

$$
\nu_x^{KG,n} = \mathbb{E}\left[\max_{x'} \frac{a_{x'}^{n+1}}{b_{x'}^{n+1}} - \max_{x'} \frac{a_{x'}^n}{b_{x'}^n} \mid S^n\right]. \tag{4.27}
$$

We omit the steps in the computation of this expectation; interested readers can follow the procedure in Section 4.12.1 and compute the formula for themselves. However, we point out one interesting detail. As in the normal-normal case, the computation of $\nu_x^{KG,n}$ requires us to find the conditional distribution, given S^n and $x^n = x$, of the parameter b_x^{n+1}. This is known as the *predictive distribution* of b_x^n. It is easy to

see that

$$\begin{aligned}\mathbb{P}\left(b_x^{n+1} > y \,|\, S^n, x^n = x\right) &= \mathbb{P}\left(W^{n+1} > y - b_x^n \,|\, S^n, x^n = x\right) \\ &= \mathbb{E}\left[\mathbb{P}\left(W^{n+1} > y - b_x^n \,|\, \lambda_x\right) \,|\, S^n, x^n = x\right] \\ &= \mathbb{E}\left[e^{-\lambda_x (y - b_x^n)} \,|\, S^n, x^n = x\right] \\ &= \left(\frac{b_x^n}{y}\right)^{a_x^n}.\end{aligned}$$

We get from the second line to the third line by using the cumulative distribution (cdf) of the exponential distribution, since W_x^{n+1} is exponential given λ_x. The last line uses the moment-generating function of the gamma distribution. From this calculation, it follows that the density of b_x^{n+1} is given by

$$f(y) = \frac{a_x^n \left(b_x^n\right)^{a_x^n}}{y^{a_x^n + 1}},$$

which is precisely the Pareto density with parameters a_x^n and b_x^n.

Once we have this fact, computing (4.27) is a matter of taking an expectation of a certain function over the Pareto density. Define $C_x^n = \max_{x' \neq x} \frac{a_x^n}{b_x^n}$. As before, this quantity represents the "best of the rest" of our estimates of the rates. We then compute

$$\tilde{\nu}_x^n = \frac{\left(b_x^n\right)^{a_x^n} \left(C_x^n\right)^{a_x^n + 1}}{\left(a_x^n + 1\right)^{a_x^n + 1}} \tag{4.28}$$

which is a sort of baseline knowledge gradient. However, depending on certain conditions, we may subtract an additional penalty from this quantity when we compute the final KG formula.

It can be shown that the KG formula is given by

$$\nu_x^{KG,n} = \begin{cases} \tilde{\nu}_x^n & \text{if } x = \arg\max_{x'} \frac{a_{x'}^n}{b_{x'}^n} \\ \tilde{\nu}_x^n - \left(C_x^n - \frac{a_x^n}{b_x^n}\right) & \text{if } \frac{a_x^n + 1}{b_x^n} > \max_{x'} \frac{a_{x'}^n}{b_{x'}^n} \\ 0 & \text{otherwise.} \end{cases} \tag{4.29}$$

This particular formula is skewed towards exploitation. If $x \neq \arg\max_{x'} \frac{a_{x'}^n}{b_{x'}^n}$, we subtract an additional value $C_x^n - \frac{a_x^n}{b_x^n}$ from the KG factor of alternative x. However, if $x = \arg\max_{x'} \frac{a_{x'}^n}{b_{x'}^n}$, there is no additional penalty. Furthermore, if our estimate of λ_x is low enough that $\frac{a_x^n + 1}{b_x^n} \leq \max_{x'} \frac{a_{x'}^n}{b_{x'}^n}$, the KG factor is automatically zero. Since $\nu_x^{KG,n} \geq 0$ for all x, just like in the normal-normal case, this means that we won't even consider an alternative if our beliefs about it are too low. So, there is a more pronounced slant in favor of alternatives with high estimates than we saw in the normal-normal setting.

Table 4.3 illustrates this issue for a problem with five alternatives. Based on our beliefs, it seems that alternative 5 is the best. This alternative also has the highest KG factor. Although our estimate of λ_3 is very close to our estimate of λ_5 (they differ

Choice	a^n	b^n	a^n/b^n	C^n	Too low?	$\tilde{\nu}_x^n$	Penalty	Final KG
1	1.0	7.2161	0.1386	0.3676	yes			0
2	4.0	12.1753	0.3285	0.3676	no	0.0472	0.0390	0.0081
3	3.0	8.1802	0.3667	0.3676	no	0.0390	0.0008	0.0382
4	5.0	19.3574	0.2583	0.3676	yes			0
5	2.0	5.4413	0.3676	0.3667	no	0.0540	0	0.0541

Table 4.3 Calculations of the KG formula for the gamma-exponential model

by less than 0.001), the KG factor for alternative 5 is nearly 1.5 times larger than the KG factor for alternative 3. Of the three remaining alternatives, only one is believed to be good enough to warrant a (very small) non-zero KG factor.

One should not take this numerical example too close to heart, however. If we keep the same numbers, but let $a_3^n = 1$ and $b_3^n = 2.78$, then alternative 3 will have the largest KG factor, even though our estimate of it would actually be 0.3597, lower than in Table 4.3. Just as in the normal-normal problem, the KG method weighs our estimate of a value against the variance of our beliefs. All other things being equal, alternatives with lower values of a_x^n (that is, alternatives that we have measured fewer times) tend to have higher KG factors, so the KG method will occasionally be moved to explore an alternative that does not currently seem to be the best.

As a final detail in our discussion, let us consider the case where our goal is to find the alternative with the smallest rate, rather than the largest. So, instead of looking for the product with the lowest demand (highest rate) in order to take it out of production, we are now looking for the product with the highest demand (lowest rate) in order to determine the most promising line of research for new products. In this case, the KG factor of alternative x is defined to be

$$\nu_x^{KG,n} = \mathbb{E}\left[\min_{x'} \frac{a_{x'}^n}{b_{x'}^n} - \min_{x'} \frac{a_{x'}^{n+1}}{b_{x'}^{n+1}} \mid S^n\right].$$

The predictive distribution of b_x^{n+1}, given S^n and $x^n = x$, is still Pareto with parameters a_x^n and b_x^n. As before, let $C_x^n = \min_{x' \neq x} \frac{a_{x'}^n}{b_{x'}^n}$, and let $\tilde{\nu}_x^n$ be as in (4.28). The KG formula resulting from taking the appropriate expectation over the Pareto density is

$$\nu_x^{KG,n} = \begin{cases} \tilde{\nu}_x^n & \text{if } x \neq \arg\max_{x'} \frac{a_{x'}^n}{b_{x'}^n} \\ \tilde{\nu}_x^n - \left(C_x^n - \frac{a_x^n}{b_x^n}\right) & \text{if } \frac{a_x^n+1}{b_x^n} > C_x^n \\ 0 & \text{otherwise.} \end{cases} \tag{4.30}$$

This formula is the mirror image of (4.29). It has a similar appearance, but its effect is precisely the opposite: it rewards exploration. In this case, we never impose a penalty on any alternative *except* for $\arg\max_{x'} \frac{a_{x'}^n}{b_{x'}^n}$. In fact, if our beliefs about the best alternative are too good (i.e. our estimate of the corresponding λ_x is too small), the KG factor of this alternative is zero, and we do not measure it. Even if we believe that the best alternative is relatively close to the second-best, we still penalize the one that we think is the best.

Recall that, in the normal-normal case, the KG formula was the same regardless if we were looking for the alternative with the largest or the smallest value. However, in the gamma-exponential model, there is a clear difference between minimizing and maximizing. The reason is because the gamma and exponential distributions are not symmetric. Our use of the gamma prior means that our estimate of λ_x could potentially take on any arbitrarily high value, but it can never go below zero. Thus, roughly speaking, our beliefs about λ_x are more likely to be low than to be high. To put it another way, the true value λ_x can always be higher than we think. Thus, if we are looking for the lowest value of λ_x, we need to push ourselves to explore more, so as not to get stuck on one alternative that seems to have a low rate. However, if we are looking for the highest value of λ_x, it is often enough to stick with an alternative that seems to have a high rate: if the rate is not really as high as we think, we will discover this quickly.

4.7.3 The gamma-Poisson model

Let us now consider a ranking and selection problem where our observations are discrete. For instance, we might consider the problem of finding the product with the highest average demand, assuming that the individual daily demands are integer-valued. Then, the daily demand for product x can be modeled as a Poisson random variable with rate λ_x. We assume that λ_x follows a gamma distribution with parameters a_x^0 and b_x^0. If we choose to measure the demand for product x^n after n days, our beliefs are updated according to the equations

$$\begin{aligned} a_x^{n+1} &= \begin{cases} a_x^n + N_x^{n+1} & \text{if } x^n = x \\ a_x^n & \text{otherwise,} \end{cases} \\ b_x^{n+1} &= \begin{cases} b_x^n + 1 & \text{if } x^n = x \\ b_x^n & \text{otherwise,} \end{cases} \end{aligned}$$

where N_x^{n+1} is the number of units of product x ordered on the next day. We assume that $N_x^{n+1} \sim Poisson\left(\lambda_x\right)$. If we are looking for the largest rate, the definition of the KG factor is once again

$$\nu_x^{KG,n} = \mathbb{E}\left[\max_{x'} \frac{a_{x'}^{n+1}}{b_{x'}^{n+1}} - \max_{x'} \frac{a_{x'}^n}{b_{x'}^n} \,\middle|\, S^n\right].$$

The problem looks deceptively similar to the gamma-exponential problem. However, the first difference is that the predictive distribution of a_x^{n+1} is now discrete, since our observation is Poisson. In fact, it can be shown that

$$\mathbb{P}\left(a_x^{n+1} = a_x^n + k \,|\, S^n, x^n = x\right) = \frac{\Gamma\left(a_x^n + k\right)}{\Gamma\left(a_x^n\right)\Gamma\left(k+1\right)} \left(\frac{b_x^n}{b_x^n + 1}\right)^{a_x^n} \left(\frac{1}{b_x^n + 1}\right)^k$$

for $k = 0, 1, 2, \ldots$. We can view this as a sort of generalization of the negative binomial distribution. In fact, if a_x^n is an integer, then

$$\frac{\Gamma\left(a_x^n + k\right)}{\Gamma\left(a_x^n\right)\Gamma\left(k+1\right)} = \binom{a_x^n + k - 1}{a_x^n - 1}$$

and the predictive distribution of a_x^{n+1} is the classic negative binomial distribution, with a_x^{n+1} representing the total number of Bernoulli trials that take place before a_x^n failures occur, with $\frac{1}{b_x^n+1}$ being the success probability.

There is no closed-form expression for the cdf of a negative binomial distribution; however, because the distribution is discrete, the cdf can always be evaluated exactly by computing and adding the appropriate terms of the probability mass function. Let $F_a(y) = \mathbb{P}(Y_a \leq y)$, where Y_a has the negative binomial distribution for a failures, with success probability $\frac{1}{b_x^n+1}$. The basic KG quantity equals

$$\tilde{\nu}_x^n = C_x^n F_{a_x^n}\left(C_x^n\left(b_x^n+1\right)\right) - \frac{a_x^n}{b_x^n} F_{a_x^n+1}\left(C_x^n\left(b_x^n+1\right)+1\right),$$

and it can be shown that the KG factor is given by

$$\nu_x^{KG,n} = \begin{cases} \tilde{\nu}_x^n - \left(C_x^n - \frac{a_x^n}{b_x^n}\right) & \text{if } x \neq \arg\max_{x'} \frac{a_{x'}^n}{b_{x'}^n} \\ \tilde{\nu}_x^n & \text{if } \frac{a_x^n}{b_x^n+1} \leq C_x^n \\ 0 & \text{otherwise.} \end{cases} \tag{4.31}$$

Interestingly, if our estimate of λ_x is too high, we will not measure x at all, but if we measure an alternative that does not seem to have the highest rate, the KG factor has an extra penalty.

4.7.4 The Pareto-uniform model

Let us consider the problem of finding the product with the largest demand from a different angle. Suppose now that the demand for product x is uniformly distributed on the interval $[0, B_x]$, where B_x is unknown. We assume that B_x follows a Pareto distribution with parameters $\alpha^0 > 1$ and $b^0 > 0$. When we choose to perform a market study on product x at time n, we update our beliefs about x according to

$$b_x^{n+1} = \begin{cases} \max\left(b_x^n, W_x^{n+1}\right) & \text{if } x^n = x \\ b_x^n & \text{otherwise,} \end{cases}$$

$$\alpha_x^{n+1} = \begin{cases} \alpha_x^n + 1 & \text{if } x^n = x \\ \alpha_x^n & \text{otherwise.} \end{cases}$$

The random variable W_x^{n+1} is our observation of the demand for product x, and is uniform on $[0, B_x]$.

The goal is to find the product with the largest possible demand $\max_x B_x$. Recall that $\mathbb{E}[B_x \mid S^n] = \frac{\alpha_x^n b_x^n}{\alpha_x^n - 1}$. Then, the knowledge gradient is defined to be

$$\nu_x^{KG,n} = \mathbb{E}\left[\max_{x'} \frac{\alpha_{x'}^{n+1} b_{x'}^{n+1}}{\alpha_{x'}^{n+1} - 1} - \max_{x'} \frac{\alpha_{x'}^n b_{x'}^n}{\alpha_{x'}^n - 1} \mid S^n\right].$$

The procedure for computing this expectation is the same as in the other models that we have considered. One thing that makes it a bit messier than usual is that the predictive distribution of b_x^{n+1}, given S^n and $x^n = x$, is neither discrete nor

continuous, but a mixture of both. This is because, if we measure x, then b_x^{n+1} is the maximum of a constant and our observation, which has the effect of "folding" part of the continuous density of the observation. It can be shown that

$$\mathbb{P}\left(b_x^{n+1} = b_x^n \,|\, S^n, x^n = x\right) = \frac{\alpha_x^n}{\alpha_x^n + 1}.$$

For $y > b_x^n$, however, the predictive distribution has a scaled Pareto density

$$f\left(y \,|\, S^n, x^n = x\right) = \frac{1}{\alpha_x^n + 1} \frac{\alpha_x^n \left(b_x^n\right)^{\alpha_x^n}}{y^{\alpha_x^n + 1}}.$$

This mixed distribution complicates the computation slightly, but the principle is the same. As usual, let

$$C_x^n = \max_{x' \neq x} \frac{\alpha_{x'}^n b_{x'}^n}{\alpha_{x'}^n - 1}$$

denote the "best of the rest." In this case, the basic KG quantity is

$$\tilde{\nu}_x^n = \frac{1}{\alpha_x^n \left(\alpha_x^n - 1\right)} \frac{\left(\alpha_x^n + 1\right)^{\alpha_x^n - 1} \left(b_x^n\right)^{\alpha_x^n}}{\left(\alpha_x^n\right)^{\alpha_x^n - 1} \left(C_x^n\right)^{\alpha_x^n - 1}}$$

and the KG formula is

$$\nu_x^{KG,n} = \begin{cases} \tilde{\nu}_x^n & \text{if } x \neq \arg\max_{x'} \frac{\alpha_{x'}^n b_{x'}^n}{\alpha_{x'}^n - 1} \\ \tilde{\nu}_x^n - \left(\frac{\alpha_x^n b_x^n}{\alpha_x^n - 1} - C_x^n\right) & \text{if } \frac{(\alpha_x^n + 1) b_x^n}{\alpha_x^n} \leq C_x^n \\ 0 & \text{otherwise.} \end{cases} \tag{4.32}$$

Thus, if we think that x has the largest possible demand, and our estimate of B_x is too much larger than our other estimates, we do not measure x. If x seems to have the largest demand, but the actual estimate is relatively close to our other estimates, the KG factor of x is non-zero, but has an extra penalty term. Otherwise, the KG factor is equal to $\tilde{\nu}_x^n$, with no additional penalty. This particular model promotes exploration.

4.7.5 The normal distribution as an approximation

While it is interesting to see the knowledge gradient computed for different distributions, most of the time we are going to use the normal distribution as an approximation for both the prior and posterior, regardless of the distribution of W. The reason, which we first addressed in section 2.6.6, is the central limit theorem. When we combine estimates from even a relatively small number of experiments (possibly as low as five), the normal distribution becomes a very good approximation, even when our experimental outcomes W are highly non-normal (e.g. binary).

The research community recognizes the value of using an approximation for the posterior, and has given it the name of *assumed density filtering* (or ADF). This is nothing more than taking a posterior distribution (density), and replacing it with a simpler one (such as the normal), which is a kind of filter.

Sampling distribution	Prior distribution	Predictive distribution
Normal	Normal	Normal
Exponential	Gamma	Pareto
Poisson	Gamma	Negative binomial
Uniform	Pareto	Mixed discrete/Pareto
Bernoulli	Beta	Bernoulli
Multinomial	Dirichlet	Multinomial
Normal	Normal-gamma	Student's t-distribution

Table 4.4 Table showing the distinctions between sampling, prior and predictive distributions for different learning models.

4.7.6 Discussion

Our examination of ranking and selection with non-Gaussian learning models underscores several interesting issues that were not as clear in the basic normal-normal problem. For one thing, there is now a real distinction between the prior, sampling and predictive distributions. In the normal-normal model, all three of these distributions were normal, but in other problems, all three can come from different families.

So far, we have managed to derive knowledge gradient formulas in all of these cases. It is clear, however, that a knowledge gradient formula depends not only on the particular type of learning model that we are using, but also on the objective function. In the gamma-exponential model, the KG algorithm uses two different formulas depending on whether we are looking for the largest exponential parameter or the smallest. Even in the basic normal-normal model, we can change the KG formula completely by changing the objective function. Exercises 4.17 and 4.19 give two examples of problems where this occurs.

The power and appeal of the KG method come from our ability to write the definition of the knowledge gradient in (4.3) in terms of some arbitrary objective function. Every new objective function requires us to recompute the KG formula, but as long as we are able to do this, we can create algorithms for problems where the objective function is very complicated. This allows us to go beyond the simple framework of ranking and selection, where the goal is always to pick the alternative with the highest value. Later on in this book, we will create knowledge gradient methods for problems where the alternatives are viewed as components that make up a large system, and the objective is a complicated function of our beliefs about the alternatives that somehow expresses the value of the entire system.

4.8 EXPECTED IMPROVEMENT

The expected improvement (EI) policy grew out of a class of learning problems where, instead of dealing with a finite set of alternatives, we have a continuous spectrum. For example, a semiconductor manufacturer uses liquid argon to provide an environment for sputtering, or depositing thin layers of metal, on semiconductor wafers. Argon is

expensive and has to be purchased from a chemical manufacturing company, so the exact amount x that is needed may require some fine-tuning. We study this problem class in much more detail in Chapter 14. For now, we can examine EI in the context of the basic ranking and selection problem with M alternatives.

EI defines improvement in the following way. At time n, our current estimate of the largest value is given by $\max_{x'} \bar{\mu}^n_{x'}$. We would like to find alternatives x whose true value μ_x is greater than this estimated quantity. To put it another way, we prefer x such that μ_x is more likely to exceed (or improve upon) our current estimate. We then define the EI factor of x as

$$\nu^{EI,n}_x = \mathbb{E}\left[\max\left\{\mu_x - \max_{x'} \bar{\mu}^n_{x'}, 0\right\} \,\middle|\, S^n, x^n = x\right]. \tag{4.33}$$

We can interpret the EI factor as the value of collecting a single observation about x vs. doing nothing. According to EI, collecting information about x is only valuable if $\mu_x > \max_{x'} \bar{\mu}^n_{x'}$, that is, if μ_x brings about an improvement. Otherwise, the value is zero. Since μ_x is unknown at time n (but $\max_{x'} \bar{\mu}^n_{x'}$ is known), we take an expected value of the improvement over the distribution of μ_x. At time n, this is $\mathcal{N}\left(\bar{\mu}^n_x, \left(\sigma^n_x\right)^2\right)$. Like KG, the EI expectation in (4.33) leads to an explicit formula

$$\nu^{EI,n}_x = \sigma^n_x f\left(\frac{\bar{\mu}^n_x - \max_{x'} \bar{\mu}^n_{x'}}{\sigma^n_x}\right), \tag{4.34}$$

which closely resembles (4.13), with some differences that we highlight below. The policy then makes the decision using

$$X^{EI,n} = \arg\max_x \nu^{EI,n}_x.$$

There are two main differences between EI and KG. First, $\nu^{KG,n}_x$ is calculated based on how likely $\bar{\mu}^{n+1}_x$ is to exceed $\max_{x' \neq x} \bar{\mu}^n_{x'}$; that is, x is compared to the best of the other alternatives. On the other hand, EI simply uses the current best estimate $\max_{x'} \bar{\mu}^n_{x'}$, the maximum over all alternatives, as the reference point. Second, EI uses the prior variance σ^n_x instead of the one-period variance reduction $\tilde{\sigma}^n_x$. One way to interpret this is that EI essentially assumes $\sigma^2_W = 0$. That is, EI considers what would happen if we could learn μ_x exactly in a single experiment.

We can still use EI in problems with noisy experiments, but the noise is not explicitly considered by the EI calculation. The literature on EI sometimes goes so far as to assume that $\sigma^2_W = 0$ in the underlying learning problem. This is not very interesting in ranking and selection, because we could find the exact optimal solution just by measuring every alternative once. However, in problems with continuous decisions, finding the best x is still challenging even when observations are exact. We return to EI in Chapter 14, where it is known as efficient global optimization or EGO.

4.9 THE PROBLEM OF PRIORS

Perhaps one of the most important yet difficult challenges in learning is constructing a reasonable prior. Sometimes a domain expert may have a reasonable understanding

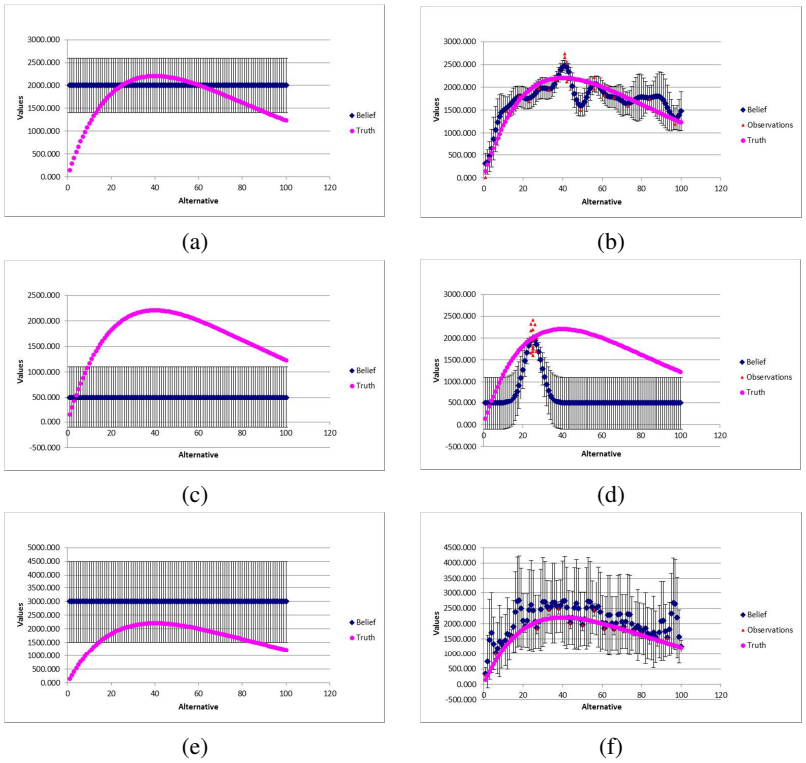


Figure 4.8 The effect of the prior on the search process. (a) shows an unbiased prior; (b) illustrates the resulting outcomes from the experiments. (c) shows a prior that is biased low; (d) illustrates that this produces a set of experiments focused around whatever point is chosen first. (e) shows a prior that is biased high; (f) illustrates that a high prior produces a search that evaluates the entire function.

about the shape of a function. The domain expert may refer to a person or to information derived from the Internet or prior studies. In the worst case, we may have to resort to doing an initial sample using randomly chosen observations. If we insist on using a Bayesian model, this last strategy is referred to as *empirical Bayes*.

The biggest problem arises when we have a prior, but it is not a very good one. Furthermore, we may be willing to assume that the precision of our prior belief is higher than it really is. Figure 4.8 presents a thought experiment with several possible scenarios. In Figure 4.8(a), we have an unknown function, a constant prior over the interval and a confidence interval that reasonably captures the spread of the function. This is a fairly accurate statement that we have no idea where the optimum is, but we do have an idea of the range of the function. Figure 4.8(b) is our estimate of the function after a series of experiments using the knowledge gradient, which shows that we do a very good job of finding the true optimum, with a very high level of confidence (note that we do not produce a precise estimate of the value of the function at the optimum).

Figure 4.8(c) illustrates a prior that is too low, and where we do not accurately represent the precision of our belief. 4.8(d) then shows that we start by sampling the

function at a random point, and not surprisingly the observed value is much higher than our prior. As a result, we tend to focus our search near this point, since our prior suggests that there is no value in sampling points far from the initial sample. If we were to repeat the exercise with a different random starting point, we would have focused our entire search close to that point. The result is an unreliable estimate of the optimum.

Figure 4.8(e) then depicts a prior that is too high. When this is the case, Figure 4.8(f) shows that we end up sampling the entire function, since every observation produces an estimate that is very low relative to our prior estimate for the rest of the function. We then proceed to sample any point that has not been sampled before (and which is not close to a previously sampled point). This works if our budget is large enough to sample the entire function, but it would work poorly if this were not the case.

Perhaps the most visible lesson from this illustration is that it is important to be honest about the uncertainty in the prior. Figures 4.8(c) and 4.8(e) both display confidence intervals that do not cover the function. A narrow confidence bound is particularly problematic if the prior is biased low, because we tend to focus all of our energy around any point that we sample.

Of course, from the Bayesian point of view, there is no one fixed function. Rather, our prior encodes our beliefs about a wide range of possible true functions. The message of the preceding example can be taken to mean that we should make sure that this range is in some way "wide enough."

Care should be used to ensure that the prior does not exclude portions of the function that may be quite good. For this reason, it is better to be biased high, and to use care not to overstate the precision of your prior (that is, where your confidence interval is too narrow). However, using a prior that is too high, and/or a confidence interval that is too large, may simply result in a lot of unnecessary exploration. In other words, the better your prior is, the better your search will be. This may be a way of saying that there is no free lunch in information collection.

4.10 VALUE OF INFORMATION WITH UNKNOWN VARIANCE*

An important simplifying assumption that we made for the knowledge gradient is that we know the variance (or precision) of an experimental outcome. In this section we consider the more difficult case of unknown variance, which was developed under the name "linear loss." Not surprisingly, this is a more difficult problem, with a corresponding increase in complexity.

Two policies have been proposed in the literature under the names $LL(1)$ and LL_1, which might leave the impression that they are the same policy. Despite having very similar names, $LL(1)$ and LL_1 are actually two different policies. Both are examples of "linear-loss" or LL methods, close analogs to KG that were developed for ranking and selection with unknown means and variances. We discussed learning in this setting in Section 2.6.5.

Supposing that alternative x has value μ_x, and our observations of μ_x also have an unknown precision β_x^W, we can represent our beliefs about the mean and precision

using a normal-gamma prior. Recall that, when we say that $\left(\mu_x, \beta_x^W\right)$ follows a normal-gamma distribution with parameters $\bar{\mu}_x^0, \tau_x^0, a_x^0, b_x^0$, we mean that β_x^W is gamma with parameters a_x^0 and b_x^0, and the conditional distribution of μ_x given that $\beta^W = r$ is normal with mean $\bar{\mu}_x^0$ and precision $\tau_x^0 r$. For each observation of μ_x that we collect, we use (2.42)-(2.45) to update our beliefs.

Both $LL(1)$ and LL_1 originate from a technique called $LL(N)$. This procedure was designed for a version of ranking and selection where, instead of collecting observations one at a time, we collect N of them at once, in a batch. Instead of making sequential decisions, where we first choose an alternative, then collect an observation, update our beliefs and make a new decision based on the new information, we make only one decision. Before collecting the batch of information, we choose how many observations (out of N total) should be collected for each alternative. For example, if we have three alternatives and $N = 10$, we might make a decision to collect $k_1 = 3$ observations of μ_1, $k_2 = 5$ observations of μ_2, and $k_3 = 2$ observations of μ_3. We make this decision based on the beliefs we have prior to collecting the batch. For example, if we have a lot of uncertainty about μ_2 (corresponding to a low value of τ_2^0), we might want to assign more observations to alternative 2.

Our goal is to choose an allocation vector $k = (k_1, ..., k_M)$ with $\sum_{i=1}^M k_i = N$ and $k_i \geq 0$ to maximize the usual objective function (**??**). In this setting, we can write the objective as

$$\max_k \mathbb{E} F^k \left(\mu, \beta^W, W\right) \tag{4.35}$$

with $F^k \left(\mu, \beta^W, W\right) = \max_x \bar{\mu}_x^n$. Notice that β^W is now included as part of the "truth," since it is also a vector of unknown parameters and affects the observations we get. The name "linear loss" is due to the fact that maximizing $\mathbb{E}F^k$ is the same as minimizing an equivalent expression $\mathbb{E}\left[\mathbb{E}^N \left(\max_x \mu_x - \max_x \bar{\mu}_x^n\right)\right]$, the expected loss or difference between the true best value and the estimated best value.

We cannot compute (4.35) directly, but we can approximate it. Let $x^* = \arg\max_x \bar{\mu}_x^0$ be the alternative that seems to be the best initially, before any information is collected. Because x^* does not depend on the choice of allocation k, we can rewrite (4.35) as

$$\max_k \mathbb{E}\left(\max_x \bar{\mu}_x^n - \bar{\mu}_{x^*}^n\right). \tag{4.36}$$

Although (4.35) and (4.36) have different optimal values, they have the same optimal solution (that is, the same k maximizes both functions). Define an indicator function

$$I_x^N = \begin{cases} 1 & \text{if } x = \arg\max_{x'} \bar{\mu}_{x'}^n \\ 0 & \text{otherwise,} \end{cases}$$

that equals 1 if and only if x is believed to be the best alternative after N experiments have been made. Also define a second indicator

$$\tilde{I}_x^N = \begin{cases} 1 & \text{if } \bar{\mu}_x^n \geq \bar{\mu}_{x^*}^n \\ 0 & \text{otherwise,} \end{cases}$$

to show whether x is believed to be better than x^* (but not necessarily better than all other alternatives) at time N. Then,

$$\begin{aligned}\mathbb{E}\left(\max_x \bar{\mu}_x^n - \bar{\mu}_{x^*}^n\right) &= \mathbb{E}\sum_{x=1}^{M} I_x^N \cdot (\bar{\mu}_x^n - \bar{\mu}_{x^*}^n) \\ &\leq \mathbb{E}\sum_{x=1}^{M} \tilde{I}_x^N \cdot (\bar{\mu}_x^n - \bar{\mu}_{x^*}^n) \\ &= \sum_{x=1}^{M} P\left(\bar{\mu}_x^n \geq \bar{\mu}_{x^*}^n\right) \mathbb{E}\left(\bar{\mu}_x^n - \bar{\mu}_{x^*}^n \mid \bar{\mu}_x^n \geq \bar{\mu}_{x^*}^n\right). \end{aligned} \tag{4.37}$$

If we replace our objective function by the upper bound in (4.37), we have replaced the objective function $\max_x \bar{\mu}_x^n$ with a sum of pairwise comparisons between individual $\bar{\mu}_x^n$ and a single value $\bar{\mu}_{x^*}^n$. This is more tractable because we can express the predictive distribution of $\bar{\mu}_x^n$ at time 0, given that we will allocate k_x observations to alternative x, in terms of Student's t-distribution. Specifically, $\sqrt{\frac{\tau_x^0 a_x^0 (\tau_x^0 + k_x)}{b_x^0 k_x}}\left(\bar{\mu}_x^n - \bar{\mu}_x^0\right)$ follows a t-distribution with $2a_x^0$ degrees of freedom. We can then compute the expectation in (4.37) for a particular choice of k, and maximize k by setting the derivative with respect to k_x equal to zero. After a few more approximations, the LL method arrives at the allocation

$$k_x = \frac{N + \sum_{x'=1}^{M} \tau_{x'}^0}{\sum_{x'=1}^{M} \sqrt{\frac{b_{x'}^0 \eta_{x'}^0}{b_x^0 \eta_x^0}}} - \tau_x^0 \tag{4.38}$$

where

$$\eta_x^0 = \begin{cases} \sqrt{\lambda_{x,x^*}^0} \frac{2a_x^0 + \lambda_{x,x^*}^0 \left(\bar{\mu}_{x^*}^0 - \bar{\mu}_x^0\right)^2}{2a_x^0 - 1} \phi_{2a_x^0}\left(\sqrt{\lambda_{x,x^*}^0}\left(\bar{\mu}_{x^*}^0 - \bar{\mu}_x^0\right)\right) & x \neq x^* \\ \sum_{x' \neq x^*} \eta_{x'}^0 & x = x^*, \end{cases}$$

the function $\phi_{2\alpha}$ is the density of the t-distribution with 2α degrees of freedom, and

$$\lambda_{x,x^*}^0 = \left(\frac{b_x^0}{\tau_x^0 a_x^0} + \frac{b_{x^*}^0}{\tau_{x^*}^0 a_{x^*}^0}\right)^{-1}. \tag{4.39}$$

The procedure uses several approximations, such as allowing k_x to be continuous when computing the optimal allocation, so it is possible for (4.38) to produce negative numbers. In this case, we need to adjust the allocation manually by rounding k_x up or down while ensuring that the allocations still add up to N.

What happens when we leave the batch setting and return to our usual world of sequential experiments? One easy solution is to run the $LL(N)$ method for $N = 1$ at every time step $n = 0, 1, 2, ..., N-1$. This is precisely what is meant by the $LL(1)$ technique. As a result, we will obtain a vector k^n of the form e_{x^n}, a vector of zeroes with a single 1 corresponding to the alternative that we need to measure. We then measure the alternative, update our beliefs, and repeat the same procedure assuming a batch of size 1.

The similarly-named LL_1 procedure works along the same lines. The main difference is that, in the batch setting, LL_1 assumes that all N samples will be allocated to a

single alternative, and then finds the most suitable alternative under these conditions. This significantly simplifies computation, because the batch is guaranteed to change only a single set of beliefs. Setting $N = 1$ inside the policy then produces a straightforward rule for making decisions at time n,

$$X^{LL_1,n} = \arg\max_x \nu_x^{LL_1,n},$$

where

$$\nu_x^{LL_1,n} = \sqrt{\lambda_{x,\tilde{x}}^n}\Psi_{2a_x^n}\left(\sqrt{\lambda_{x,\tilde{x}}^n}\,|\bar{\mu}_x^n - \bar{\mu}_{\tilde{x}}^n|\right), \tag{4.40}$$

for $\tilde{x} = \arg\max_{x' \neq x} \bar{\mu}_x^n$, and $\Psi_d(z) = \frac{d+z^2}{d-1}\phi_d(z) - z\Phi_d(-z)$, with ϕ_d, Φ_d being the pdf and cdf of the standard t-distribution with d degrees of freedom. The quantity $\lambda_{x,\tilde{x}}^n$ is computed exactly as in (4.39), but using the time-n beliefs. Notice that this procedure compares our beliefs about x to our beliefs about $\tilde{x}$, the best alternative other than x. This recalls the behavior of the KG policy, and indeed, LL_1 is the exact analog of KG for ranking and selection with unknown precision in the noise of an experiment.

The computation required for LL_1 is simpler and requires fewer approximations than $LL(1)$. It is less suitable for the batch setting, where we would most likely choose to divide our observations among many alternatives, but better suited to the sequential setting considered in this chapter. Additionally, some experiments in the simulation literature suggest that we can learn more effectively and discover better alternatives by using LL_1 to make decisions sequentially or with very small batches, rather than by running $LL(N)$ to allocate our entire learning budget at once.

4.11 DISCUSSION

The appeal of the knowledge gradient is that it is a simple idea that can be applied to many settings. A particularly powerful feature of the knowledge gradient is that it can capture the important dimension of correlated beliefs. In fact, it is useful to review the list of applications given in Section 1.2 where you will see that almost all of these are characterized by correlated beliefs. Later in the volume, we consider more complex sets of alternatives such as finding the best subset, or the best value of a continuous, multidimensional parameter vector.

The knowledge gradient (as with any policy based on the expected value of a single experiment) is vulnerable to the nonconcavity of information. Indeed, if you have a problem where the value of information is nonconcave, then you have to address the issues discussed in Section 4.3, regardless of your choice of learning policy. However, if the value of a single experiment is nonconcave, then this simply means that you have to think about taking repeated experiments. This behavior would almost always be present if the information W^n is binomial, which means that we should be thinking about the value of multiple trials.

We have presented results for situations other than the normal-normal model, but we suspect that most applications will lend themselves reasonably well to the normal-normal model, for two reasons. First, while the initial prior may not be normal, the central limit theorem generally means that estimates of parameters after a few

observations are likely to be described by a normal distribution. Second, while a single observation W^n may be non-normal, if we have to use repeated observations (in a single trial) to overcome the nonconcavity of information, then the combined effect of multiple observations is likely to be accurately described by a normal distribution.

4.12 WHY DOES IT WORK?*

4.12.1 Derivation of the knowledge gradient formula

It is not necessary to know how to derive the knowledge gradient formula in order to be able to use it effectively, but sometimes it is nice to go past the "trust me" formulas and see the actual derivation. The presentation here is more advanced (hence the * in the section title), but it is intended to be tutorial in nature, with additional steps that would normally be excluded from a traditional journal article.

The knowledge gradient method is characterized by simplicity and ease of use. In every time step, we can compute $\nu_x^{KG,n}$ for each alternative x by plugging the current values of $\bar{\mu}_x^n$ and $\bar{\sigma}_x^{2,n}$ into the formulas (4.10) and (4.11), and then applying (4.13). After that, our experimental decision is given by $X^{KG,n} = \arg\max_x \nu_x^{KG,n}$, the alternative with the largest knowledge gradient value.

However, it is worthwhile to go through the derivation of the knowledge gradient formula at least once. Not only does this make the KG formulas look less unwieldy, by showing how they originate from the definition of the knowledge gradient, it also gives a sense of how we might go about creating a knowledge gradient method in other optimal learning problems. Later on in this chapter, we derive KG formulas for ranking and selection problems that use some of the non-Gaussian learning models from Chapter 2. This can be done using the same approach as for the independent normal case, but the resulting KG formulas will be very different from the formulas for the normal-normal model.

The expression in (4.4) gives a generic definition of a KG policy, in terms of the expected improvement made by running an experiment. We can write down this expectation in many different settings, but the way we compute it (if, indeed, we can compute it at all) will vary from problem to problem. Often, deriving a computable form for the KG policy poses a computational challenge in research. Thus, the KG approach is a very general idea, but the algorithmic realization of that idea is heavily problem-specific.

We now show how $\nu_x^{KG,n}$ is derived for a particular choice of alternative x at time n. At time n, the estimates $\bar{\mu}_x^n$ and $\bar{\sigma}_x^{2,n}$ are known to us for all x. However, the future estimates $\bar{\mu}_x^{n+1}$ are still random, because we have not yet decided on the $(n+1)$st experiment. It is important to remember that, because the problem is sequential, each new time step changes what is random and what is known. For example, the nth estimate $\bar{\mu}_x^n$ is a random variable from the point of view of any time $n' < n$, but it is a constant from the point of view of time n, when the first n observations have been irrevocably made.

Our goal is to compute (4.3), which requires us to find

$$\mathbb{E}\left[V^{n+1}(S^n) \,\middle|\, S^n, x^n = x\right] = \mathbb{E}\left[\max_{x'} \bar{\mu}_{x'}^{n+1} \,|\, S^n, x^n = x\right].$$

We assume that we measure x at time n, and examine how this experiment will affect our beliefs about the best alternative. Fortunately, we can simplify our expression for $\max_{x'} \bar{\mu}_{x'}^{n+1}$. Recall from the updating equations (3.2) and (3.3) that $\bar{\mu}_{x'}^{n+1} = \bar{\mu}_{x'}^{n}$ for any $x' \neq x^n$. Thus, we can rewrite

$$\mathbb{E}\left[\max_{x'} \bar{\mu}_{x'}^{n+1} \,\middle|\, S^n, x^n = x\right] = \mathbb{E}\left[\max\left(\max_{x' \neq x} \bar{\mu}_{x'}^{n}, \bar{\mu}_{x}^{n+1}\right) \,\middle|\, S^n, x^n = x\right]. \tag{4.41}$$

From the point of view of time n, the quantity $\max_{x'} \bar{\mu}_{x'}^{n+1}$ is merely a maximum of a single random variable and a constant. It is typically much easier to compute the expected value of such a quantity than the maximum of multiple random variables.

Next, we consider the conditional distribution of $\bar{\mu}_x^{n+1}$ given S^n and $x^n = x$. From the updating equations, we know that

$$\bar{\mu}_x^{n+1} = \frac{\beta_x^n}{\beta_x^n + \beta_x^W} \bar{\mu}_x^n + \frac{\beta_x^W}{\beta_x^n + \beta_x^W} W_x^{n+1}, \tag{4.42}$$

a weighted average of a constant $\bar{\mu}_x^n$ and a random variable W_x^{n+1}. Given our beliefs at time n, the conditional distribution of the true value μ_x of x is normal with mean $\bar{\mu}_x^n$ and variance $\bar{\sigma}_x^{2,n}$. Then, given μ_x, the observation W_x^{n+1} is itself conditionally normal with mean μ_x and variance σ_x^2. What we need, however, is the conditional distribution of W_x^{n+1} given our beliefs at time n, but not given the true value μ_x, and we can find this distribution by computing the moment-generating function

$$\begin{aligned}
\mathbb{E}\left(e^{-rW_x^{n+1}}\right) &= \mathbb{E}\left(\mathbb{E}\left(e^{-rW_x^{n+1}} \,\middle|\, \mu_x\right)\right) \\
&= \mathbb{E}\left(e^{-r\mu_x} e^{\frac{1}{2}\sigma_x^2 r^2}\right) \\
&= e^{\frac{1}{2}\sigma_x^2 r^2} \mathbb{E}\left(e^{-r\mu_x}\right) \\
&= e^{\frac{1}{2}\sigma_x^2 r^2} e^{-r\bar{\mu}_x^n} e^{\frac{1}{2}\bar{\sigma}_x^{2,n} r^2} \\
&= e^{-r\bar{\mu}_x^n} e^{\frac{1}{2}\left(\sigma_x^2 + \bar{\sigma}_x^{2,n}\right) r^2}.
\end{aligned}$$

This is clearly the moment-generating function of the normal distribution with mean $\bar{\mu}_x^n$ and variance $\sigma_x^2 + \bar{\sigma}_x^{2,n}$. The variance can also be written in precision notation as $1/\beta_x^W + 1/\beta_x^n$.

It follows that the conditional distribution of $\bar{\mu}_x^{n+1}$ given S^n and $x^n = x$ is also normal, since $\bar{\mu}_x^{n+1}$ is a linear function of W_x^{n+1} by (4.42). We can find the mean and variance of this distribution by computing

$$\begin{aligned}
\mathbb{E}\left[\bar{\mu}_x^{n+1} \,|\, S^n, x^n = x\right] &= \frac{\beta_x^n}{\beta_x^n + \beta_x^W} \bar{\mu}_x^n + \frac{\beta_x^W}{\beta_x^n + \beta_x^W} \mathbb{E}\left[W_x^{n+1} \,|\, S^n, x^n = x\right] \\
&= \frac{\beta_x^n}{\beta_x^n + \beta_x^W} \bar{\mu}_x^n + \frac{\beta_x^W}{\beta_x^n + \beta_x^W} \bar{\mu}_x^n \\
&= \bar{\mu}_x^n
\end{aligned}$$

and

$$\begin{aligned} Var\left[\bar{\mu}_x^{n+1} \mid S^n, x^n = x\right] &= \left(\frac{\beta_x^W}{\beta_x^n + \beta_x^W}\right)^2 Var\left[W_x^{n+1} \mid S^n, x^n = x\right] \\ &= \left(\frac{\beta_x^W}{\beta_x^n + \beta_x^W}\right)^2 \left(\frac{1}{\beta_x^W} + \frac{1}{\beta_x^n}\right) \\ &= \frac{\beta_x^W}{\beta_x^n\left(\beta_x^W + \beta_x^n\right)}. \end{aligned}$$

Using the definition of the precision again, we can write

$$\begin{aligned} \frac{\beta_x^W}{\beta_x^n\left(\beta_x^W + \beta_x^n\right)} &= \frac{\bar{\sigma}_x^{2,n}}{\sigma_x^2\left(\frac{1}{\sigma_x^2} + \frac{1}{\bar{\sigma}_x^{2,n}}\right)} \\ &= \frac{\bar{\sigma}_x^{2,n}}{1 + \sigma_x^2/\bar{\sigma}_x^{2,n}} \end{aligned}$$

which is precisely $\tilde{\sigma}_x^{2,n}$ by (4.9).

We have found that the conditional distribution of $\bar{\mu}_x^{n+1}$, given S^n and $x^n = x$, is normal with mean $\bar{\mu}_x^n$ and variance $\tilde{\sigma}_x^{2,n}$. In words, when we measure x at time n, we expect that the next observation will be equal to $\bar{\mu}_x^n$, that is, our beliefs $\bar{\mu}_x^n$ are accurate on average. As a consequence of this experiment, the variance of our beliefs about μ_x will decrease by an amount equal to the variance of the next observation.

Now, we can return to (4.41) and rewrite the right-hand side as

$$\mathbb{E}\left[\max\left(\max_{x' \neq x} \bar{\mu}_{x'}^n, \bar{\mu}_x^{n+1}\right) \,\middle|\, S^n, x^n = x\right] = \mathbb{E}\left[\max\left(\max_{x' \neq x} \bar{\mu}_{x'}^n, \bar{\mu}_x^n + \tilde{\sigma}_x^n Z\right)\right]$$

where Z is a standard normal (mean 0, variance 1) random variable. Our goal is now to compute an expectation of a function of Z, which looks far more tractable than the expression we started out with.

Let $C_x^n = \max_{x' \neq x} \bar{\mu}_{x'}^n$, and observe that

$$\max\left(C_x^n, \bar{\mu}_x^n + \tilde{\sigma}_x^n Z\right) = \begin{cases} C_x^n & \text{if } Z \leq \frac{C - \bar{\mu}_x^n}{\tilde{\sigma}_x^n} \\ \bar{\mu}_x^n + \tilde{\sigma}_x^n & \text{otherwise.} \end{cases}$$

Thus,

$$\begin{aligned} \mathbb{E}\left[\max\left(C_x^n, \bar{\mu}_x^n + \tilde{\sigma}_x^n Z\right)\right] &= \int_{-\infty}^{\frac{C_x^n - \bar{\mu}_x^n}{\tilde{\sigma}_x^n}} C_x^n \phi(z)\, dz \\ &\quad + \int_{\frac{C_x^n - \bar{\mu}_x^n}{\tilde{\sigma}_x^n}}^{\infty} \left(\bar{\mu}_x^n + \tilde{\sigma}_x^n z\right) \phi(z)\, dz \end{aligned}$$

where

$$\int_{-\infty}^{\frac{C_x^n - \bar{\mu}_x^n}{\tilde{\sigma}_x^n}} C_x^n \phi(z)\, dz = C_x^n \Phi\left(\frac{C_x^n - \bar{\mu}_x^n}{\tilde{\sigma}_x^n}\right)$$

and

$$\int_{\frac{C_x^n-\bar{\mu}_x^n}{\tilde{\sigma}_x^n}}^{\infty}(\bar{\mu}_x^n+\tilde{\sigma}_x^n z)\,\phi(z)\,dz=\bar{\mu}_x^n\Phi\left(-\frac{C_x^n-\bar{\mu}_x^n}{\tilde{\sigma}_x^n}\right)+\tilde{\sigma}_x^n\phi\left(\frac{C_x^n-\bar{\mu}_x^n}{\tilde{\sigma}_x^n}\right).$$

The first term in the second equation uses the symmetry of the normal distribution, and the second term is due to the fact that

$$\int_y^{\infty} z\phi(z)\,dz=\phi(y),$$

which can be verified by a back-of-the-envelope calculation.

Observe now that, due to the symmetry of the normal density,

$$\phi\left(\frac{C_x^n-\bar{\mu}_x^n}{\tilde{\sigma}_x^n}\right)=\phi\left(-\frac{C_x^n-\bar{\mu}_x^n}{\tilde{\sigma}_x^n}\right)=\phi(\zeta_x^n),$$

which gives us one of the terms that make up the KG formula. To obtain the other term, we consider two cases. If $C_x^n\leq\bar{\mu}_x^n$, then $\bar{\mu}_x^n=\max_{x'}\bar{\mu}_{x'}^n$ and

$$\begin{aligned}C_x^n\Phi\left(\frac{C_x^n-\bar{\mu}_x^n}{\tilde{\sigma}_x^n}\right)+\bar{\mu}_x^n\Phi\left(-\frac{C_x^n-\bar{\mu}_x^n}{\tilde{\sigma}_x^n}\right) &= C_x^n\Phi(\zeta_x^n)+\bar{\mu}_x^n-\bar{\mu}_x^n\Phi(\zeta_x^n)\\ &= \bar{\mu}_x^n+\tilde{\sigma}_x^n\left(\frac{C_x^n-\bar{\mu}_x^n}{\tilde{\sigma}_x^n}\right)\Phi(\zeta_x^n)\\ &= \bar{\mu}_x^n+\tilde{\sigma}_x^n\zeta_x^n\Phi(\zeta_x^n).\end{aligned}$$

However, if $\bar{\mu}_x^n<C_x^n$, then $C_x^n=\max_{x'}\bar{\mu}_{x'}^n$ and

$$\begin{aligned}C_x^n\Phi\left(\frac{C_x^n-\bar{\mu}_x^n}{\tilde{\sigma}_x^n}\right)+\bar{\mu}_x^n\Phi\left(-\frac{C_x^n-\bar{\mu}_x^n}{\tilde{\sigma}_x^n}\right) &= C_x^n-C_x^n\Phi(\zeta_x^n)+\bar{\mu}_x^n\Phi(\zeta^n)x)\\ &= C_x^n+\tilde{\sigma}_x^n\zeta_x^n\Phi(\zeta_x^n).\end{aligned}$$

Either way,

$$C_x^n\Phi\left(\frac{C_x^n-\bar{\mu}_x^n}{\tilde{\sigma}_x^n}\right)+\bar{\mu}_x^n\Phi\left(-\frac{C_x^n-\bar{\mu}_x^n}{\tilde{\sigma}_x^n}\right)=\max_{x'}\bar{\mu}_{x'}^n+\tilde{\sigma}_x^n\zeta_x^n\Phi(\zeta_x^n).$$

Putting all the terms together,

$$\begin{aligned}\mathbb{E}\left[\max(C_x^n,\bar{\mu}_x^n+\tilde{\sigma}_x^n Z)\right] &= \max_{x'}\bar{\mu}_{x'}^n+\tilde{\sigma}_x^n(\zeta_x^n\Phi(\zeta_x^n)+\phi(\zeta_x^n))\\ &= \max_{x'}\bar{\mu}_{x'}^n+\tilde{\sigma}_x^n f(\zeta_x^n),\end{aligned}$$

whence

$$\begin{aligned}\nu_x^{KG,n} &= \mathbb{E}\left[\max_{x'}\bar{\mu}_{x'}^{n+1}-\max_{x'}\bar{\mu}_{x'}^n\,\middle|\,S^n,x^n=x\right]\\ &= \mathbb{E}\left[\max\left(\max_{x'\neq x}\bar{\mu}_{x'}^n,\bar{\mu}_x^{n+1}\right)\,\middle|\,S^n,x^n=x\right]-\max_{x'}\bar{\mu}_{x'}^n\\ &= \tilde{\sigma}_x^n f(\zeta_x^n),\end{aligned}$$

which is precisely the KG formula that we introduced earlier.

The above derivation relies on our assumption of independent alternatives, as well as our use of a normal-normal learning model. However, in the process, we followed a set of steps that can be used to derive knowledge gradient formulas for other learning problems, as well. In particular, we use the exact same approach to derive KG formulas for ranking and selection problems with non-Gaussian learning models later in this chapter. The steps are:

1) We calculate the conditional distribution, given S^n and $x^n = x$, of the value function $V^{n+1}\left(S^{n+1}(x)\right)$. In ranking and selection with independent alternatives, this reduces to the problem of finding the conditional distribution of $\bar{\mu}_x^{n+1}$ (in the above derivation, this was normal). Recall that this is called the predictive distribution of $\bar{\mu}_x^{n+1}$, because it is what we predict about time $n+1$ at time n.

2) We calculate the conditional expectation of $V^{n+1}\left(S^{n+1}(x)\right)$ over the predictive distribution. This is especially simple in ranking and selection with independent alternatives, because it is simply an expected value of a function of a one-dimensional random variable (in the above derivation, this was the standard normal random variable Z).

3) We subtract the quantity $V^n(S^n)$, which is viewed as deterministic at time n.

In most of the problems discussed in this book, the real challenge lies in the second step. For example, if we introduce correlations into the problem, the predictive distribution of the vector $\bar{\mu}^{n+1}$ is still fairly simple, but it is a bit more difficult to calculate the expectation of the value function $V^{n+1}\left(S^{n+1}(x)\right)$ over this distribution. However, there are many learning problems that seem complicated, but that actually yield simple closed-form expressions for the knowledge gradient. Later on, we encounter many more varied examples of such problems.

4.13 BIBLIOGRAPHIC NOTES

Section 4.2 - The idea of collecting information based on the expected value of a single experiment was first introduced by Gupta & Miescke (1994) and Gupta & Miescke (1996). The concept of the knowledge gradient was developed in greater depth in Frazier et al. (2008). We connect the KG concept to the idea of information economics from Chapter **??**; in fact, Chick & Gans (2009) describes a KG-like approach as the "economic approach to simulation selection."

Section 4.3 - The material on the non-concavity of information is based on Frazier & Powell (2010). Additional discussion on the nonconcavity of information can be found in Weibull et al. (2007), which also presents conditions where it is not always optimal to choose the alternative that appears to be best.

Section 4.5 - The knowledge gradient for correlated beliefs was first introduced in Frazier et al. (2009).

Section 4.7 - The development of the knowledge gradient for non-Gaussian distributions in this section is mostly new. A version of KG for the gamma-exponential model was presented in Ryzhov & Powell (2011*c*).

Section **??** - The expected improvement algorithm was proposed by Jones et al. (1998*a*) for problems with continuous decisions. This algorithm is also sometimes known as efficient global optimization or EGO in this setting. Some more recent work on EI can be found in Gramacy & Lee (2011). The $LL(N)$ methodology was originally put forth by Chick & Inoue (2001), with extensive empirical validation undertaken by Inoue et al. (1999) and Branke et al. (2005). A general overview of the LL approach is available in Chick (2006). The LL_1 procedure is a more recent development laid out in Chick et al. (2010).

PROBLEMS

4.1 Your estimate of the long-run performance of a mutual fund was that it returns 8 percent, but your distribution of belief around this number is normally distributed with a standard deviation of 3 percent. You update this each year (assume that successive years are independent), and from history you estimate that the standard deviation of the return in a particular year is 6 percent. At the end of the most recent year, the mutual fund returned -2 percent.

a) Use Bayesian updating to update the mean and standard deviation.

b) What do we mean when we say that the "normal distribution is conjugate"?

c) What will be the precision of my estimate of the long-run performance in four years (after four experiments, starting with the current belief state)?

4.2 You have three places that serve takeout food around your area and you want to maximize the quality of the total food you intake over time. The three restaurants are:

1. Vine Garden (VG)
2. Wise Sushi (WS)
3. Food Village (FV)

You assume a normal prior with (μ_x, β_x) on the quality of the food in these places and the experiments are normally distributed with precision $\beta^W = 1$.

1. Define the expected opportunity cost (EOC) for this setting (assume that the discount factor, $\gamma = 1$).
2. For a single ω, Table 4.5 below contains your prior, the truth and the outcomes of your observations until the third time step $n = 3$. Write down the empirical opportunity cost after the third observation (note that you will need to update your priors).
3. If you are valuing your current utility from food higher than future time periods $(\gamma < 1)$, how would you expect the behavior of the optimal policy to change as opposed to having $\gamma = 1$?

4.3 Table 4.8 shows the priors $\bar{\mu}^n$ and the standard deviations σ^n for five alternatives.

Iteration	VG	WS	FV
Prior (μ_x, β_x)	$(6, 2)$	$(7, 1)$	$(8, 1)$
Truth for μ_x	5	8	7
1	4		
2		9	
3			8.5

Table 4.5 Priors and observations

a) Compute the knowledge gradient for each alternative in a spreadsheet. Create a plot with the mean, standard deviation and the knowledge gradient for each alternative.

b) Three of the alternatives have the same standard deviation, but with increasing priors. Three have the same prior, but with increasing standard deviations. From these two (overlapping) sets of alternatives, describe how the knowledge gradient changes as we vary priors and the standard deviation of our belief.

Choice	$\bar{\mu}^n$	σ^n
1	3.0	8.0
2	4.0	8.0
3	5.0	8.0
4	5.0	9.0
5	5.0	10.0

Table 4.6 Calculations illustrating the knowledge gradient index

4.4 Assume that we have a standard normal prior about a true parameter μ which we assume is normally distributed with mean $\bar{\mu}^0$ and variance $(\sigma^0)^2$.

a) Given the observations $W^1, \ldots, W^n$, is $\bar{\mu}^n$ deterministic or random?

b) Given the observations $W^1, \ldots, W^n$, what is $\mathbb{E}(\mu|W^1, \ldots, W^n)$ (where μ is our truth)? Why is μ random given the first n experiments?

c) Given the observations $W^1, \ldots, W^n$, what is the mean and variance of $\bar{\mu}^{n+1}$? Why is $\bar{\mu}^{n+1}$ random?

4.5 As a venture capitalist specializing in energy technologies, you have to decide to invest in one of three strategies for converting solar power to electricity. A major concern is the efficiency of a solar panel, which tends to run around 11 to 12 percent. You are at the point where you are running field experiments, but each field experiment produces an estimate of the efficiency which has a standard deviation of 4.0. The first

technology appears to have an efficiency of 11.5 percent, but the standard deviation in your distribution of belief around this number is 2.0. The second technology has an estimated efficiency of 11.0 with a standard deviation of 3.5, while the third has an estimated efficiency of 12.0, with a standard deviation of 1.5. You want to choose the technology with the highest efficiency.

a) Use the knowledge gradient to tell you which technology you should experiment with next.

b) If you are only going to do one last experiment, is this the optimal choice? Explain.

c) If you did not make this last investment, you would choose technology 3 with an efficiency of 12.0. What is the expected efficiency of the technology that would be chosen as best after the last investment?

4.6 Consider the problem of finding the best person to serve as the lead-off hitter on a baseball team. The lead-off hitter is evaluated primarily for his ability to get on base. If x is the hitter, his outcome would be recorded as a Bernoulli random variable W_x^n, where $W_x^n = 1$ if he gets on base, and $W_x^n = 0$ otherwise. We are going to conduct these experiments during spring training, where we are primarily focused on finding the best lead-off hitter (we do not really care about his performance while we are collecting the information). Learning is accomplished using the beta-Bernoulli model. Each alternative x has an unknown success probability ρ_x, and our goal is to find the alternative with the highest success probability. We begin with a prior belief $\rho_x \sim Beta\left(\alpha_x^0, \beta_x^0\right)$. Supposing that we measure alternative x^n at time n, our beliefs are updated according to the equations

$$\begin{aligned}
\alpha_x^{n+1} &= \begin{cases} \alpha_x^n + W_x^{n+1} & \text{if } x^n = x \\ \alpha_x^n & \text{otherwise,} \end{cases} \\
\beta_x^{n+1} &= \begin{cases} \beta_x^n + \left(1 - W_x^{n+1}\right) & \text{if } x^n = x \\ \beta_x^n & \text{otherwise,} \end{cases}
\end{aligned}$$

where the observation W^{n+1} is equal to 1 with probability ρ_x and 0 with probability $1 - \rho_x$. Recall that, under this model, our estimate of the uncertain truth ρ_x at time n is $\mathbb{E}\left(\rho_x \,|\, S^n\right) = \frac{\alpha_x^n}{\alpha_x^n + \beta_x^n}$.

a) Suppose that we measure alternative x at time n. Show (by conditioning on the truth) that

$$\mathbb{P}\left(W_x^{n+1} = 1 \,|\, S^n\right) = \frac{\alpha_x^n}{\alpha_x^n + \beta_x^n}, \qquad \mathbb{P}\left(W_x^{n+1} = 0 \,|\, S^n\right) = \frac{\beta_x^n}{\alpha_x^n + \beta_x^n}.$$

b) Use the definition of the knowledge gradient to write out an expression for the knowledge gradient for this problem. You do not have to reduce the expression in any way (for example, you will have an expectation, but you do not have to reduce it to a convenient expression).

4.7 Garrett Jones was a minor leaguer in baseball trying to break into the major leagues. He was called up to play in a few major league games, where he made one hit

in eight at bats. After this weak performance, he was sent back to the minor leagues. The major league club that was evaluating him is looking for someone who can hit at a certain level against an existing major league hitter. Think of this as choosing between an uncertain minor leaguer, and a more certain major leaguer (so this is a case with two alternatives).

a) It is reasonable to assume that no-one would ever make a decision based on a single at bat. Assume that our minor leaguer will be given at least 10 at bats, and that we will now assume that our prior belief about his batting average is normally distributed with mean 0.250 and standard deviation 0.20. Further assume that our belief about the major leaguer is also normally distributed with mean 0.267 and standard deviation of 0.10. Finally assume that we are going to approximate the observed batting average from at least 10 at bats as normally distributed with mean:

$$W_{minor} = \frac{H}{m}$$

where H is the number of hits and m is the number of at bats. The variance of W_{minor} is given by

$$\sigma_W^2 = \rho_{minor}(1 - \rho_{minor})/m$$

where $\rho_{minor} = .235$ is the expected batting average of the minor leaguer (the true batting average is a random variable). Give the expression for the knowledge gradient resulting from m at bats and compute the knowledge gradient.

b) Assume that the knowledge gradient for a single at bat is very small. Without actually computing the knowledge gradient, plot what is likely the general shape of the value of observing m at bats as a function of m. If you were going to use the KG(*) policy, what would you do? What are the implications of this shape in terms of how a coach should evaluate different minor leaguers?

4.8 Consider a ranking and selection problem with exponential observations and gamma priors. That is, if we choose to measure alternative x at time n, we observe $W_x^{n+1} \sim Exp(\lambda_x)$. The rate λ_x is unknown, but we start with the assumption that $\lambda_x \sim Gamma\left(\alpha_x^0, \beta_x^0\right)$ and update these beliefs as we run experiments. Our beliefs about the alternatives are independent. Thus, if we measure alternative x at time n, we update $\alpha_x^{n+1} = \alpha_x^n + 1$ and $\beta_x^{n+1} = \beta_x^n + W_x^{n+1}$, while keeping $\alpha_y^{n+1} = \alpha_y^n$ and $\beta_y^{n+1} = \beta_y^n$ for all $y \neq x$.

a) Suppose that our objective is to find the largest rate λ_x. Define the knowledge gradient of alternative x at time n as

$$\nu_x^{KG,n} = \mathbb{E}\left[\max_y \frac{\alpha_y^{n+1}}{\beta_y^{n+1}} - \max_y \frac{\alpha_y^n}{\beta_y^n} \mid S^n, x^n = x\right].$$

Argue that

$$\mathbb{E}\left[\max_y \frac{\alpha_y^{n+1}}{\beta_y^{n+1}} \mid S^n, x^n = x\right] = \mathbb{E}\left[\max\left(C_x^n, \frac{\alpha_x^n + 1}{Y}\right)\right]$$

where $C_x^n = \max_{y \neq x} \frac{\alpha_y^n}{\beta_y^n}$ and $Y \sim Pareto\,(\alpha_x^n, \beta_x^n)$.

(Remember that the $Pareto\,(a,b)$ density is $g\,(t) = \frac{ab^a}{t^{a+1}}$ for $t > b$ and zero elsewhere.)

b) Suppose that $\frac{\alpha_x^n+1}{\beta_x^n} \leq C_x^n$. Show that

$$\mathbb{E}\left[\max\left(C_x^n, \frac{\alpha_x^n+1}{Y}\right)\right] = C_x^n.$$

c) Suppose that $\frac{\alpha_x^n+1}{\beta_x^n} > C_x^n$. Show that

$$\mathbb{E}\left[\max\left(C_x^n, \frac{\alpha_x^n+1}{Y}\right)\right] = \frac{\alpha_x^n}{\beta_x^n} + \frac{(\beta_x^n)^{\alpha_x^n}(C_x^n)^{\alpha_x^n+1}}{(\alpha_x^n+1)^{\alpha_x^n+1}}.$$

d) Based on parts b) and c), show that

$$\nu_x^{KG,n} = \begin{cases} \frac{(\beta_x^n)^{\alpha_x^n}(C_x^n)^{\alpha_x^n+1}}{(\alpha_x^n+1)^{\alpha_x^n+1}} & \text{if } x = \arg\max_y \frac{\alpha_y^n}{\beta_y^n} \\ \frac{(\beta_x^n)^{\alpha_x^n}(C_x^n)^{\alpha_x^n+1}}{(\alpha_x^n+1)^{\alpha_x^n+1}} - \left(\max_y \frac{\alpha_y^n}{\beta_y^n} - \frac{\alpha_x^n}{\beta_x^n}\right) & \text{if } x \neq \arg\max_y \frac{\alpha_y^n}{\beta_y^n}, \\ & \frac{\alpha_x^n+1}{\beta_x^n} > \max_y \frac{\alpha_y^n}{\beta_y^n} \\ 0 & \text{otherwise.} \end{cases}$$

4.9 We again consider a ranking and selection problem with exponential observations and gamma priors. There are five alternatives, and the belief state for a certain time step n is given in table 4.7. The objective is to find the largest rate.

x	α_x^n	β_x^n
1	2	18
2	3	17
3	1	7
4	2	15
5	3	14

Table 4.7 Priors for exercise 4.9

4.10 Consider a ranking and selection problem with independent alternatives, exponential observations and gamma priors.

a) Suppose that we want to find the alternative with the highest rate. Derive the KG formula given in (4.29). (Hint: Consider the three cases given in the formula separately before trying to take integrals.)

b) Repeat your analysis for the case where our goal is to find the lowest rate. Show that the KG formula is given by (4.30).

4.11 Table 4.8 shows the priors $\bar{\mu}^n$ and the standard deviations σ^n for five alternatives.

a) Compute the knowledge gradient for each alternative in a spreadsheet. Create a plot with the mean, standard deviation and the knowledge gradient for each alternative.

b) Three of the alternatives have the same standard deviation, but with increasing priors. Three have the same prior, but with increasing standard deviations. From these two (overlapping) sets of alternatives, describe how the knowledge gradient changes as we vary priors and the standard deviation of our belief.

Choice	$\bar{\mu}^n$	σ^n
1	3.0	8.0
2	4.0	8.0
3	5.0	8.0
4	5.0	9.0
5	5.0	10.0

Table 4.8 Calculations illustrating the knowledge gradient index

a) Compute $\nu_x^{KG,n}$ for all x. Which alternative will the KG policy measure at time n?

b) Suppose now that $\alpha_2^n = 1$ and $\beta_2^n = 6$, while our beliefs about the other alternatives remain unchanged. How does this change your answer to part a)? Why did KG change its decision, even though the estimate $\frac{\alpha_2^n}{\beta_2^n}$ is actually smaller now than what it was in part a)?

4.12 Table 4.9 shows the priors $\bar{\mu}^n$ and the standard deviations σ^n for five alternatives.

a) Compute the knowledge gradient for each in a spreadsheet.

b) You should observe that the knowledge gradients are fairly small. Provide a plain English explanation for why this would be the case.

Choice	$\bar{\mu}^n$	σ^n
1	3.0	4.0
2	4.0	6.0
3	20.0	3.0
4	5.0	5.0
5	6.0	7.0

Table 4.9 Priors for exercise 4.12

4.13 In Section 4.12.1, we showed that $\mathbb{E}\left[\bar{\mu}_x^{n+1} \mid S^n, x^n = x\right] = \bar{\mu}_x^n$ in the normal-normal model. Verify that this also holds for our estimates of the unknown parameters in other learning models:

a) Show that $\mathbb{E}\left[\frac{\alpha_x^{n+1}}{\beta_x^{n+1}} \,\middle|\, S^n, x^n = x\right] = \frac{\alpha_x^n}{\beta_x^n}$ in the gamma-exponential model.

b) Repeat part a) for the gamma-Poisson model.

c) Show that $\mathbb{E}\left[\frac{\alpha_x^{n+1}}{\alpha_x^{n+1}-1} b_x^{n+1} \,\middle|\, S^n, x^n = x\right] = \frac{\alpha_x^n}{\alpha_x^n - 1} b_x^n$ in the Pareto-uniform model.

d) Show that $\mathbb{E}\left[\frac{\alpha_x^{n+1}}{\alpha_x^{n+1}+\beta_x^{n+1}} \,\middle|\, S^n, x^n = x\right] = \frac{\alpha_x^n}{\alpha_x^n+\beta_x^n}$ in the beta-Bernoulli model.

In each of these cases, our estimates of the unknown parameters (the rate λ_x, the upper endpoint B_x, and the success probability ρ_x) are expected to stay the same on average. That is, given that we are at time n, we expect that the estimate will not change on average between time n and time $n+1$. This is called the *martingale property*.

4.14 Consider a ranking and selection problem with independent alternatives, Poisson observations and gamma priors. Suppose that the objective is to find the alternative with the highest Poisson rate. Show that the KG formula is given by (4.31).

4.15 Consider a ranking and selection problem with independent alternatives, uniform observations and Pareto priors. Suppose that the objective is to find the largest upper endpoint among the uniform distributions.

a) Show that the predictive distribution of b_x^{n+1} given S^n and $x^n = x$ is a mixed discrete/continuous distribution given by

$$\mathbb{P}\left(b_x^{n+1} = b_x^n \mid S^n, x^n = x\right) = \frac{\alpha_x^n}{\alpha_x^n + 1}$$

and

$$f\left(y \mid S^n, x^n = x\right) = \frac{1}{\alpha_x^n + 1} \frac{\alpha_x^n \left(b_x^n\right)^{\alpha_x^n}}{y^{\alpha_x^n+1}}, \qquad y > b_x^n.$$

b) Show that the KG formula for alternative x in this problem is given by (4.32).

4.16 Consider a ranking and selection problem with independent alternatives, Bernoulli observations and beta priors. Suppose that the objective is to find the alternative with the largest success probability. Show that the KG formula is given by (4.26), and verify that this formula remains the same (aside from changing the maximum in the definition of C_x^n to a minimum) if we change the objective to finding the lowest success probability instead of the highest.

4.17 Consider a ranking and selection problem with independent alternatives, normal observations and normal priors. However, instead of the usual objective function $F^\pi = \max_x \bar{\mu}_x^n$, we use

$$F^\pi = \sum_x \mathbb{E}\left[\left(\mu_x - \bar{\mu}_x^n\right)^2 \,\middle|\, S^N\right].$$

That is to say, instead of trying to find the alternative with the highest value, our goal is now to run experiments in such a way as to reduce (on average) the sum of squared errors of our final estimates at time N of all the values. Derive the KG formula for this problem.

4.18 Suppose that we have four different products. The profit margin of product x, $x = 1, 2, 3, 4$ is represented by μ_x. We can choose to perform a market study on product x to get an observation W_x of its profit margin. The observation is normally distributed with mean μ_x and variance $(\sigma_x^W)^2$. Table 4.10 below gives the true values of μ_x and $(\sigma_x^W)^2$:

x	μ_x	$\left(\sigma_x^W\right)^2$
1	15	4
2	10	3
3	12	5
4	11	2

Table 4.10 Priors for exercise 4.18.

However, we do not know the true values of μ_x. We describe our beliefs about them using a multivariate Gaussian prior with the following mean vector and covariance matrix:

$$\bar{\mu}^0 = \begin{bmatrix} 12 \\ 14 \\ 13 \\ 10 \end{bmatrix}, \qquad \Sigma^0 = \begin{bmatrix} 12 & 0 & 6 & 3 \\ 0 & 7 & 4 & 2 \\ 6 & 4 & 9 & 0 \\ 3 & 2 & 0 & 8 \end{bmatrix}.$$

Assume that we choose to observe product 3 and we observe $W_3^1 = 15$. Show how our beliefs would change using the updating equations

$$\bar{\mu}^1 = \bar{\mu}^0 + \frac{W_x^1 - \bar{\mu}_x^0}{\left(\sigma_x^W\right)^2 + \Sigma_{xx}^0}\Sigma^0 e_x, \qquad \Sigma^1 = \Sigma^0 - \frac{\Sigma^0 e_x e_x^T \Sigma^0}{\left(\sigma_x^W\right)^2 + \Sigma_{xx}^n} \tag{4.43}$$

where $x = 3$ is the particular product that you are considering, and e_x is a column vector of zeroes with a single 1 in the xth coordinate. Report the resulting values of $\bar{\mu}^1$ and Σ^1. (Equation (4.43) gives the "convenient" version of the updating equations, where you don't have to compute an inverse. You do not have to derive these equations.)

4.19 Consider a ranking and selection problem with independent alternatives, normal observations and normal priors. However, instead of the usual objective function $F^\pi = \max_x \bar{\mu}_x^n$, we use

$$F^\pi = \max_x |\bar{\mu}_x^n|.$$

That is, we want to find the alternative with the largest absolute value. Show that the KG factor of alternative x is given by

$$\nu_x^{KG,n} = \tilde{\sigma}_x^n \left(f\left(\zeta_x^n\right) + f\left(\delta_x^n\right)\right),$$

where

$$\zeta_x^n = -\left|\frac{\bar{\mu}_x^n - \max_{x' \neq x} |\bar{\mu}_{x'}^n|}{\tilde{\sigma}_x^n}\right|, \qquad \delta_x^n = -\frac{\bar{\mu}_x^n + \max_{x' \neq x} |\bar{\mu}_{x'}^n|}{\tilde{\sigma}_x^n.}$$

4.20 The revenue generated by an online advertisement has an exponential distribution with parameter λ (thus the mean revenue is $\frac{1}{\lambda}$). We do not know λ, so we use a gamma

prior and assume $\lambda \sim Gamma(a, b)$ for $a > 1$. Recall that the density of the gamma distribution is given by

$$f(x) = \frac{b(bx)^{a-1}e^{-bx}}{\Gamma(a)},$$

where $\Gamma(a) = (a-1)!$ if a is integer. The mean of $f(x)$ is a/b and the variance is a/b^2.

a) What is the current belief about the value of λ? That is, if you had to guess the value of λ, what would you say, and why? If you assume that λ is exactly equal to this belief, what is the mean revenue generated by the advertisement?

b) Now take an expectation of the mean revenue over the entire distribution of belief. That is, compute $\mathbb{E}(\frac{1}{\lambda})$ for $\lambda \sim Gamma(a, b)$.

c) Why are your answers to (a) and (b) different? Which one should you actually use as your estimate of the mean reward, and why?

4.21 Consider a ranking and selection problem with normal observations and normal priors (and independent beliefs).

a) Create a MATLAB file called `kg.m` which implements the KG policy. As a template, you can use the code that was first introduced in exercise 3.2 which can be downloaded from

`http://optimallearning.princeton.edu/exercises/exploration.m`

`http://optimallearning.princeton.edu/exercises/explorationRun.m`

b) Set $N = 5000$, $M = 50$ and report the confidence interval.

c) Set $N = 1$, $M = 1$ and run the policy 100 times. How often does KG find the best alternative?

4.22 You would like to find the price of a product that maximizes revenue. Unknown to you, the demand for the product is given by

$$D(p) = 100e^{-.02p}.$$

Total revenue is given by $R(p) = pD(p)$. Assume prices are integers between 1 and 100.

You set a price and watch it for a week. Assume that the observed revenue R^n in week n is given by

$$R^n = R(p) + \epsilon^n$$

where $\epsilon^n \sim \mathcal{N}(0, 400^2)$. However, since you believe that the function is continuous, you realize that your beliefs are correlated. Assume that your belief about $R(p)$ and $R(p')$ is correlated with covariance function

$$Cov^0(R(p), R(p')) = 400^2 e^{-0.03|p-p'|}.$$

So, $Cov^0(R(p), R(p)) = Var^0(R(p)) = 400^2$, and $Cov^0(R(20), R(30)) = 400^2 \times 0.7408$. Use this to create your prior covariance matrix Σ^0. Assume that your initial estimate of $R(p)$ is $\bar{\mu}_p^0 = 2000$ for each p (this is known as a "uniform prior," and represents a situation where you have no idea which price is the best.) Note that we are using online learning for this exercise.

a) Write out the updating formulas for updating your estimate $\bar{\mu}_p^n$ giving the estimated revenue when you charge price p, and the updating formula for the covariance matrix Σ^n.

b) Implement the algorithm for computing the knowledge gradient in the presence of correlated beliefs (call this algorithm KGCB), using as a starting point

`http://optimallearning.princeton.edu/exercises/KGCorrBeliefs.m`

An example illustration of the KGCB algorithm is given in

`http://optimallearning.princeton.edu/exercises/KGCorrBeliefsEx.m`

Verify your algorithm first by running it with a diagonal covariance matrix, and showing that the independent and correlated KG algorithms give you the same numbers. Note that you may find that for some prices, the knowledge gradient is too small to compute (for example, you get a very negative exponent).

c) Next use the initial covariance matrix described above. Plot the log of the knowledge gradient for prices between 1 and 100 (again, be careful with large negative exponents), and compare your results to the log of the knowledge gradient assuming independent beliefs. How do they compare?

d) Now we are going to compare policies. Please do the following:

i) Run your KGCB algorithm for 10 experiments, and plot after each experiment the opportunity cost, which means you take what you think is the best price based on your current set of estimates, and compare it to the revenue you would get if you knew the best price (hint: it is \$50). Repeat this exercise 20 times, and report the average opportunity cost, averaged over the 20 iterations. The goal here is to get a sense of the variability of the performance of a learning policy.

ii) We want to compare KGCB against pure exploration, pure exploitation and interval estimation using $z_\alpha = 1.96$ (a standard default). For each policy, perform 20 sample paths and plot the average opportunity cost over all 20 sample paths. Compare all three policies. [Please make sure that you are resetting the covariance matrix to its initial structure after you are done with a sample path (once you have gone over a single truth). If you skip this part, the KG algorithm will assume it has very precise priors for the second run, which of course is not true.]

e) The knowledge gradient policy can be quite sensitive to the prior. Instead of an initial prior of 2000, now assume that we start with a uniform prior of 500 (same standard deviation). If you edit the prior (column B), the package regenerates the truth. You are going to have to re-enter the formula for the truth after changing the prior. After doing this, perform 3 repetitions of KG, pure exploration and pure exploitation, and contrast their performance.

CHAPTER 5

ONLINE LEARNING

In this chapter we turn to the problem where we have to learn as we go. This means that we no longer have the luxury of making mistakes within our experimental budget before we have to decide which design to implement. Now, every decision is an implementation decision where we have to live with the consequences, but simultaneously learning as we go.

On the surface, the difference between the offline learning problem we introduced in chapter 3, and the online learning problem we address in this chapter, is the objective function. With the offline problem, our objective could be stated as

$$\max_{\pi} \mathbb{E}_{\mu}\mathbb{E}_{W^1,\ldots,W^N|\mu}\mu_{x^{\pi,N}}, \tag{5.1}$$

where $x^{\pi,N}$ is the alternative we think is best given our estimates $\bar{\mu}_x^N$, given by

$$x^{\pi,N} = \arg\max_x \bar{\mu}_x^N.$$

This means we are only interested in the performance $\mu_{x^{\pi,N}}$ of the final design $x^{\pi,N}$ that we learned by following learning policy π after exhausting our experimental budget N.

For an online learning problem, we have to learn as we go. So, if $X^\pi(S)$ is our policy that returns a decision $x^n = X^\pi(S^n)$ when our belief state is S^n. We then receive a

reward W^{n+1}. This means we have to find the policy that solves

$$\max_{\pi} \mathbb{E}_{\mu}\mathbb{E}_{W^1,\ldots,W^N|\mu} \sum_{n=0}^{N-1} W_{x^n}^{n+1}, \tag{5.2}$$

where $x^n = X^\pi(S^n)$. If we assume that we have access to the true mean μ_x for the purpose of testing policies (as we did in equation (5.1)), we could write our objective as

$$\max_{\pi} \mathbb{E}_{\mu}\mathbb{E}_{W^1,\ldots,W^N|\mu} \sum_{n=0}^{N-1} \mu_{X^\pi(S^n)}. \tag{5.3}$$

However we calculate the objective, we see that our offline problem focuses on the final (or terminal) reward or performance, while our online problem focuses on the cumulative reward.

Viewed this way, the difference between offline (terminal reward) and online (cumulative reward) seems modest, and for some classes of policies, it is. However, online learning problems have a long and rich history where it has evolved as the *multiarmed bandit problem*.

The multi-armed bandit problem (or MAB, as it is often known) is a venerable topic in optimal learning, and has inspired some of the pioneering work in the field. The story that was originally used to motivate the problem (and gave the problem its name) is not really an important application, but is useful for understanding the basic idea behind the problem. The term "one-armed bandit" refers to a slot machine operated by pulling a lever (or "arm") on its side. A multi-armed bandit, then, is a collection of slot machines, each with a different winning probability (or different average winnings).

Suppose that we have M slot machines each with a different but unknown reward distribution (yes, that is the story). If we play slot machine x at time n, we receive random winnings W_x^{n+1}. Suppose that the expected value of these winnings, given by μ_x, is unknown. We would like to estimate μ_x, but the only way to do this is by putting money into the machine and collecting a random observation. For this reason, we must balance our desire to find the slot machine with the highest value with our desire to achieve good results on every play.

There are numerous examples of online learning problems. Some examples are

---

EXAMPLE 5.1

You need to find the best path to work, but the only way to evaluate a path is to actually try it.

EXAMPLE 5.2

A physician has to find the best diabetes medication for a patient, but the only way to see how the patient responds is to try the medication on the patient.

■ **EXAMPLE 5.3**

A company needs to place bids to have its ad appear on Google. The company has to try different bids to find the one that strikes the best balance between cost and response.

■ **EXAMPLE 5.4**

An internet movie provider has to decide which movies to show to maximize the likelihood that a user logging into her account will pick one of the movies to watch.

■ **EXAMPLE 5.5**

An internet retailer has to experiment with different prices for a product to find the price that maximizes revenue.

Both offline and online learning problems require balancing exploration and exploitation, but the tradeoff is more apparent with online problems.

As early as 1970, we have known that we could characterize an optimal policy by setting up Bellman's equation

$$V^n(S^n) = \max_x \left(\bar{\mu}_x^n + \mathbb{E}_W V^{n+1}(S^{n+1})\right),\ n = 0, \ldots, N-1 \tag{5.4}$$

$$V^N(S^N) = \max_x \bar{\mu}_x^N, \tag{5.5}$$

where $S^{n+1} = S^M(S^n, x, W_x^{n+1})$. Here, $S^n = (\bar{\mu}_x^n, \beta_x^n)_{x \in \mathcal{X}}$ is the belief state, x is a decision of which "arm" to try, W_x^{n+1} is what we learn, and $S^M(S, x, W)$ captures the equations for updating the belief model. The problem is that if we have a normally distributed belief model and M alternatives, then S has $2M$ continuous dimensions. As a result, solving equation (5.5) is completely intractable.

Given this, it was quite a breakthrough when J.C. Gittins found, in 1974, that instead of solving a $2M$ dimensional problem across all M alternatives, it was possible to find an optimal policy by solving M individual dynamic programs. Each of these dynamic programs returned an index, later known as the *Gittins index*, which required that we simply find the alternative with the highest index. This became known as an *index policy*. This spawned a steady stream of papers looking for optimal index policies for variations of the basic bandit problem.

The development of Gittins indices generated considerable enthusiasm in the applied probability community, although they still remained a computational challenge in that while they were computable, they were not easy to compute. We illustrate the basic idea of Gittins indices, which also serves to highlight the relative complexity, especially compared to the policies we have seen so far for the offline problem.

A completely different approach was developed by Lai and Robbins in 1985, who found that a very simple idea called upper confidence bounding (which we first saw in chapter 3) produced a policy where it is possible to bound how often it chose the wrong arm. While this does not mean that UCB policies (as they are known) are

optimal, it provided evidence that they might be quite good. In addition, they are trivial to compute, which makes them attractive for high-speed internet applications.

Reflecting the combined contributions of these two communities, the multi-armed bandit problem has attracted a substantial following that has grown past the core niche communities that developed these ideas. In fact, these communities have collectively spawned the study of a wide range of "bandit problems" that has introduced novel problem variations that have been overlooked by other communities working on stochastic optimization problems.

We are ultimately going to see that there are many policies that were designed originally for online learning problems, but which still need to be tuned, and which can be tuned for either online or offline applications. We are going to cover three major classes of policies:

Excitation policy Excitation policies use a simple rule, such as a decision that appears to be best, and introduces some noise to encourage exploration.

Upper confidence bounding UCB policies are trivial to compute and enjoy bounds that suggest that they should work well, although they still have to be tuned in practice.

Gittins indices Gittins indices determine the best decision based on the value of information from the decision over the entire horizon, computed using dynamic programming. These are more difficult to compute, but provide a rare example of an optimal policy when certain assumptions are satisfied.

Lookahead policies While Gittins indices using dynamic programming to compute the value of being in a belief state, another approach is to try to model decisions and learning in the future to help make a decision now. We have already seen this in section 1.6 when we used decision trees to model decisions and information over some horizon. In this chapter we are going to illustrate some different strategies for approximating the future.

5.1 AN EXCITATION POLICY

Consider the problem of pricing a product where we have to set a price x in the hope of maximizing revenue. We might assume that we can approximate the demand $D(p)$ as a function of price p over a reasonable range of prices using a linear function

$$D(p|\theta) = \theta_1 - \theta_2 p.$$

We would like to maximize the revenue $R(p) = pD(p)$. If we knew $\theta = (\theta_1, \theta_2)$ we would just take the derivative of $R(p)$, set it equal to zero and solve for the optimal price, giving us

$$p^* = \frac{\theta_1}{2\theta_2}.$$

The problem is that we do not know θ. Let $\bar{\theta}^n$ be our estimate of θ after n samples. This means that we would compute our price p^n using

$$p^n = \frac{\bar{\theta}_1^n}{2\bar{\theta}_2^n}.$$

Further assume that after setting a price p^n we observe a revenue $\hat{R}^{n+1}$. The revenue $\hat{R}^{n+1}$ presumably comes from

$$\hat{R}^{n+1} = \theta_1 - \theta_2 p^n + \varepsilon^{n+1},$$

where θ_1 and θ_2 are the true (but unknown) parameters, and ε^{n+1} represents noise in the process of generating revenue.

Since we only have an estimate $\bar{\theta}^n$, we would expect to use the data $(\hat{R}^{n+1}, p^n)$ to obtain an updated estimate $\bar{\theta}^{n+1}$ (we learn how to do this with a linear belief model in chapter 7).

This seems reasonable, except that this method will tend to test prices in a fairly narrow range. If we think $\bar{\theta}^n$ is a good estimate of the true θ, this is fine. But what if we do not have a lot of confidence in this estimate? Sampling points in a narrow range is not a good way of generating data to estimate a model.

A simple strategy that is widely used in some communities is to simply add a noise term, ε^p, to induce exploration. Assume that $\varepsilon^p \sim N(0, \sigma_\epsilon^2)$, where the standard deviation σ_ϵ is a tunable parameter. We refer to the resulting policy as an *excitation policy* which is written as

$$P^{exc}(S^n|\sigma_\epsilon) = \frac{\bar{\theta}_1^n}{2\bar{\theta}_2^n} + \varepsilon^{p,n+1}. \tag{5.6}$$

We can think of the rule in equation (5.6) as a type of policy where we use some formula for making a decision, such as choosing the decision that appears to be best, and then adding noise (of course, this only works if the decision is continuous). The variance σ_ϵ^2 is a tunable parameter, where larger values encourage more exploration.

5.2 UPPER CONFIDENCE BOUNDING

A class of policies that has received considerable interest is known as *upper confidence bounding* or UCB policies. These policies are quite simple to implement, and different variants have been developed using this approach for many types of reward distributions. For example, imagine we have a problem where all the rewards are in the interval $[0, 1]$, e.g. if we are using a beta-Bernoulli model. In this setting, one possible UCB policy defines the index of alternative x to be

$$\nu_x^{UCB1,n} = \bar{\mu}_x^n + \sqrt{\frac{2\log n}{N_x^n}}, \tag{5.7}$$

where N_x^n is the number of times we have played arm x up to and including time n. The policy is somewhat analogous to interval estimation: we take our current

estimate of μ_x and add an uncertainty bonus. Just as in interval estimation, this particular uncertainty bonus represents the half-width of a confidence interval. In a sense, (5.7) represents a probabilistic guess of the largest possible value that μ_x could realistically take on. We choose to measure, not the alternative with the highest estimated value, but rather the alternative that *could* potentially have the largest value.

The policy in (5.7) is geared toward the specific case where the rewards are in $[0, 1]$. A UCB policy designed for the normal-normal model defines the index of x as

$$\nu_x^{UCB1-Normal,n} = \bar{\mu}_x^n + 4\sigma_W \sqrt{\frac{\log n}{N_x^n}}. \tag{5.8}$$

There are also many other versions in the literature for both beta-Bernoulli and normal-normal problems, but (5.7) and (5.8) are two of the most prominent. Problems with bounded rewards turn out to be particularly attractive for the UCB approach because, in these settings, a particular proof technique can be applied to create UCB policies with provable bounds on the regret.

There is also a UCB policy for the gamma-exponential model, where W_x follows an exponential distribution with parameter λ_x, and our beliefs about λ_x are represented by a gamma distribution with parameters a_x^n and b_x^n. In this case, we compute

$$\nu_x^{UCB-Exp,n} = \frac{b_x^n}{a_x^n - 1} + \theta \min\left(\sqrt{2\frac{\log n + 2\log\log n}{N_x^n}}, 1\right), \tag{5.9}$$

with θ being a tunable parameter.

The reasons why UCB policies have attracted attention is a) they are easy to compute and b) because they have an optimality property of sorts. If we are able to make N experiments, and we make them by following a UCB-type policy, then the average number of times we will play a sub-optimal alternative (an alternative with $\mu_x < \max_{x'} \mu_{x'}$) can be bounded above by $C \log N$, where C is some constant. This is known as a *regret bound*. Thus, the number of times that we choose any particular sub-optimal alternative is on the order of $\log N$. It has been proven that this is the best possible bound (up to the choice of C) on the number of times a sub-optimal machine is played. Each of the UCB policies given above have this property.

The structure of the policies (5.7), (5.8) and (5.9) all have the general form

$$X^{UCB}(S^n|\theta) = \arg\max_x \left(\bar{\mu}_x^n + \theta B_x^{unc,n}\right), \tag{5.10}$$

where $B_x^{unc,n}$ is often referred to as the *uncertainty bonus*. The trick is in the design of this bonus. We are looking for a function that starts large (which encourages the logic to try an alternative) and shrinks as the number of times we test an alternative grows. The trick is in designing this bonus. For example, we could use

$$B_x^{unc,n} = 4\sigma_W \sqrt{\frac{\log n}{N_x^n}}, \tag{5.11}$$

or

$$B_x^{unc,n} = 4\sigma_W \frac{\log n}{N_x^n}. \tag{5.12}$$

Both of these exhibit the same behavior of starting with a large bonus, declining as we test alternative x. So why do we use (5.11) instead of (5.12)? The answer is simply that (5.11) enjoys a tighter regret bound than (5.12). At the same time, the coefficient $4\sigma_W$ is typically discarded in practice, and replaced with a tunable parameter, producing a policy of the form

$$X^{UCB}(S^n|\theta^{UCB}) = \arg\max_x \left(\bar{\mu}_x^n + \theta^{UCB}\sqrt{\frac{\log n}{N_x^n}}\right). \tag{5.13}$$

We have simply replaced the coefficient $4\sigma^W$ with a tunable parameter θ^{UCB}, which is typically tuned in an ad hoc way. We return to the issue of tuning later.

5.3 GITTINS INDICES

The idea behind Gittins indices works as follows. Assume that we are playing a single slot machine, and that we have the choice of continuing to play the slot machine or stopping and switching to a process that pays a reward r . If we choose not to play, we receive r, and then find ourselves in the same state (since we did not collect any new information). If we choose to play, we earn a random amount W, plus we earn $\mathbb{E}\{V(S^{n+1}, r)|S^n\}$, where S^{n+1} represents our new belief state resulting from our observed winnings. For reasons that will become clear shortly, we write the value function as a function of the state S^{n+1} and the stopping reward r.

The value of being in state S^n, then, can be written as

$$V(S^n, r) \quad = \quad \max\left[r + \gamma V(S^n, r), \mathbb{E}\left\{W^{n+1} + \gamma\, V(S^{n+1}, r)\middle|\, S^n\right\}\right]. \tag{5.14}$$

The first choice represents the decision to receive the fixed reward r, while in the second, we get to observe W^{n+1} (which is random when we make the decision). When we have to choose x^n, we will use the expected value of our return if we continue playing, which is computed using our current belief state. For example, in the Bayesian normal-normal model, $\mathbb{E}\{W^{n+1}|S^n\} = \bar{\mu}^n$, which is our estimate of the mean of W given what we know after the first n experiments.

If we choose to stop playing at iteration n, then S^n does not change, which means we earn r and face the identical problem again for our next play. In this case, once we decide to stop playing, we will never play again, and we will continue to receive r (discounted) from now on. The infinite horizon, discounted value of this reward is $r/(1-\gamma)$. This means that we can rewrite our optimality recursion as

$$V(S^n, r) = \max\left[\frac{r}{1-\gamma}, \mathbb{E}\left\{W^{n+1} + \gamma\, V(S^{n+1}, r)\middle|\, S^n\right\}\right], \tag{5.15}$$

Here is where we encounter the magic of Gittins indices. We compute the value of r that makes us indifferent between stopping and accepting the reward r (forever), versus continuing to play the slot machine. That is, we wish to solve the equation

$$\frac{r}{1-\gamma} = \mathbb{E}\left\{W^{n+1} + \gamma\, V(S^{n+1}, r)\middle|\, S^n\right\} \tag{5.16}$$

for r. The Gittins index $I^{Gitt,n}$ is the particular value of r that solves (5.16). This index depends on the state S^n. If we use a Bayesian perspective and assume normally distributed rewards, we would use $S^n = (\bar{\mu}^n, \beta^n)$ to capture our distribution of belief about the true mean μ. If we use a frequentist perspective, our state variable would consist of our estimate $\bar{\theta}^n$ of the mean, our estimate $\hat{\sigma}^{2,n}$ of the variance, and the number N^n of observations (this is equal to n if we only have one slot machine).

If we have multiple slot machines, we consider every machine separately, as if it were the only machine in the problem. We would find the Gittins index $I_x^{Gitt,n}$ for every machine x. Gittins showed that, if $N \to \infty$, meaning that we are allowed to make infinitely many experiments, it is optimal to play the slot machine with the highest value of $I_x^{Gitt,n}$ at every time n. Notice that we have not talked about how exactly (5.16) can be solved. In fact, this is a major issue, but for now, assume that we have some way of computing $I_x^{Gitt,n}$.

Recall that, in ranking and selection, it is possible to come up with trivial policies that are asymptotically optimal as the number of experiments goes to infinity. For example, the policy that measures every alternative in a round-robin fashion is optimal for ranking and selection: if we have infinitely many chances to measure, this policy will measure every alternative infinitely often, thus discovering the true best alternative in the limit. However, in the multi-armed bandit setting, this simple policy is likely to work extremely badly. It may discover the true best alternative in the limit, but it will do poorly in the early iterations. If $\gamma < 1$, the early iterations are more important than the later ones, because they contribute more to our objective value. Thus, in the online problem, it can be more important to pick good alternatives in the early iterations than to find the true best alternative. The Gittins policy is the only policy with the ability to do this optimally.

5.3.1 Gittins indices in the beta-Bernoulli model

The Gittins recursion in (5.15) cannot be solved using conventional dynamic programming techniques. Even in the beta-Bernoulli model, one of the simplest learning models we have considered, the number of possible states S^n is uncountably infinite. In other models like the normal-normal model, S^n is also continuous. However, in some models, the expectation in the right-hand side of (5.15) is fairly straightforward, allowing us to get a better handle on the problem conceptually.

Let us consider the beta-Bernoulli model for a single slot machine. Each play has a simple 0/1 outcome (win or lose), and the probability of winning is ρ. We do know this probability exactly, so we assume that ρ follows a beta distribution with parameters α^0 and β^0. Recall that the beta-Bernoulli model is conjugate, and the updating equations are given by

$$\begin{aligned} \alpha^{n+1} &= \alpha^n + W^{n+1}, \\ \beta^{n+1} &= \beta^n + \left(1 - W^{n+1}\right), \end{aligned}$$

where the distribution of W^{n+1} is Bernoulli with success probability ρ. After n plays, the distribution of ρ is beta with parameters α^n and β^n. The belief state for a

single slot machine is simply $S^n = (\alpha^n, \beta^n)$. Consequently,

$$\begin{aligned}\mathbb{E}\left(W^{n+1} \,|\, S^n\right) &= \mathbb{E}\left[\mathbb{E}\left(W^{n+1} \,|\, S^n, \rho\right) \,|\, S^n\right] \\ &= \mathbb{E}\left(\rho \,|\, S^n\right) \\ &= \frac{\alpha^n}{\alpha^n + \beta^n}.\end{aligned}$$

Then, writing $V\left(S^n, r\right)$ as $V\left(\alpha^n, \beta^n, r\right)$, we obtain

$$\begin{aligned}\mathbb{E}\left\{W^{n+1} + \gamma\, V(S^{n+1}, r) \middle| S^n\right\} &= \frac{\alpha^n}{\alpha^n + \beta^n} + \gamma \frac{\alpha^n}{\alpha^n + \beta^n} V\left(\alpha^n + 1, \beta^n, r\right) \\ &\quad + \gamma \frac{\beta^n}{\alpha^n + \beta^n} V\left(\alpha^n, \beta^n + 1, r\right). \qquad (5.17)\end{aligned}$$

For fixed α and β, the quantity $V\left(\alpha, \beta, r\right)$ is a constant. However, if the observation W^{n+1} is a success, we will transition to the belief state $(\alpha^n + 1, \beta^n)$, and if it is a failure, the next knowledge will be $(\alpha^n, \beta^n + 1)$. Given S^n, the conditional probability of success is $\frac{\alpha^n}{\alpha^n + \beta^n}$.

From (5.17), it becomes clear why Gittins indices are difficult to compute exactly. For any value of r and any α, β, we need to know $V\left(\alpha + 1, \beta, r\right)$ as well as $V\left(\alpha, \beta + 1, r\right)$ before we can compute $V\left(\alpha, \beta, r\right)$. But there is no limit on how high α and β are allowed to go. These parameters represent roughly the tallies of successes and failures that we have observed, and these numbers can take on any integer value if we assume an infinite horizon.

However, it is possible to compute $V\left(\alpha, \beta, r\right)$ approximately. For all α and β such that $\alpha + \beta$ is "large enough," we could assume that $V\left(\alpha, \beta, r\right)$ is equal to some value, perhaps zero. Then, a backwards recursion using these terminal values would give us approximations of $V\left(\alpha, \beta, r\right)$ for small α and β.

The quality of such an approximation would depend on how many steps we would be willing to perform in the backwards recursion. In other words, the larger the value of $\alpha + \beta$ for which we cut off the recursion and set a terminal value, the better. Furthermore, the approximation would be improved if these terminal values were themselves as close to the actual value functions as possible.

One way of choosing terminal values is the following. First, fix a value of r. If $\alpha + \beta$ is very large, it is reasonable to suppose that

$$V\left(\alpha, \beta, r\right) \approx V\left(\alpha + 1, \beta, r\right) \approx V\left(\alpha, \beta + 1, r\right).$$

Then, we can combine (5.15) with (5.17) to approximate the Gittins recursion as

$$V\left(\alpha, \beta, r\right) = \max\left[\frac{r}{1-\gamma}, \frac{\alpha}{\alpha + \beta} + \gamma V\left(\alpha, \beta, r\right)\right]. \qquad (5.18)$$

In this case, it can be shown that (5.18) has the solution

$$V\left(\alpha, \beta, r\right) = \frac{1}{1-\gamma} \max\left(r, \frac{\alpha}{\alpha + \beta}\right),$$

and the solution to (5.16) in this case is simply

$$r^* = \frac{\alpha}{\alpha + \beta}.$$

β \ α	1	2	3	4	5	6	7
1	0.7029	0.8001	0.8452	0.8723	0.8905	0.9039	0.9141
2	0.5001	0.6346	0.7072	0.7539	0.7869	0.8115	0.8307
3	0.3796	0.5163	0.6010	0.6579	0.6996	0.7318	0.7573
4	0.3021	0.4342	0.5184	0.5809	0.6276	0.6642	0.6940
5	0.2488	0.3720	0.4561	0.5179	0.5676	0.6071	0.6395
6	0.2103	0.3245	0.4058	0.4677	0.5168	0.5581	0.5923
7	0.1815	0.2871	0.3647	0.4257	0.4748	0.5156	0.5510

Table 5.1 Gittins indices for the beta-Bernoulli model with $\alpha, \beta = 1, ..., 7$ for $\gamma = 0.9$.

We can use this result to approximate Gittins indices for a desired α^n and β^n. First, we choose some large number N. If $\alpha + \beta \geq N$, we assume that $V(\alpha, \beta, r) = \frac{1}{1-\gamma}\frac{\alpha}{\alpha+\beta}$ for all r. Then, we can use (5.15) and (5.17) to work backwards and compute $V(\alpha^n, \beta^n, r)$ for a particular value of r. Finally, we can use a search algorithm to find the particular value r^* that makes the two components of the maximum in the expression for $V(\alpha^n, \beta^n, r)$ equal.

The computational cost of this method is high. If N is large, the backwards recursion becomes more expensive for each value of r, and we have to repeat it many times to find the value r^*. However, the recursion (for fixed r) is simple enough to be coded in a spreadsheet, and r can then be varied through trial and error (see Exercise 5.3). Such an exercise allows one to get a sense of the complexity of the problem.

When all else fails, the monograph by Gittins (1989) provides tables of Gittins indices for several values of γ and $\alpha, \beta = 1, 2, ..., 40$. A few of these values are given in Tables 5.1 and 5.2 for $\gamma = 0.9, 0.95$. The tables allow us to make several interesting observations. First, the Gittins indices are all numbers in the interval $[0, 1]$. In fact, this is always true in the beta-Bernoulli model (but not for other models, as we shall see in the next section). Second, the indices are increasing in the number of successes, and decreasing in the number of failures. This is logical; if the number of successes is low, and the number of trials is high, we can be fairly sure that the success probability of the slot machine is low, and therefore the fixed-reward process should give us smaller rewards.

5.3.2 Gittins indices in the normal-normal model

Let us now switch to the normal-normal model. Instead of winning probabilities, we deal with average winnings, and we assume that the winnings in each play follow a normal distribution. The quantity μ_x represents the unknown average winnings of slot machine x. Every observation W_x^n is normal with mean μ_x and known precision β^W, and every unknown mean μ_x is normally distributed with prior mean $\bar{\mu}_x^0$ and

β \ α	1	2	3	4	5	6	7
1	0.7614	0.8381	0.8736	0.8948	0.9092	0.9197	0.9278
2	0.5601	0.6810	0.7443	0.7845	0.8128	0.8340	0.8595
3	0.4334	0.5621	0.6392	0.6903	0.7281	0.7568	0.7797
4	0.3477	0.4753	0.5556	0.6133	0.6563	0.6899	0.7174
5	0.2877	0.4094	0.4898	0.5493	0.5957	0.6326	0.6628
6	0.2439	0.3576	0.4372	0.4964	0.5440	0.5830	0.6152
7	0.2106	0.3172	0.3937	0.4528	0.4999	0.5397	0.5733

Table 5.2 Gittins indices for the beta-Bernoulli model with $\alpha, \beta = 1, ..., 7$ for $\gamma = 0.95$.

prior precision β_x^0. In every time step, we select an alternative x, observe a random reward W_x^{n+1}, and apply (3.2)-(3.3) to obtain a new set of beliefs $\left(\bar{\mu}^{n+1}, \beta^{n+1}\right)$.

The Gittins index of slot machine x at time n can be written as the function $I_x^{Gitt,n}\left(\bar{\mu}_x^n, \sigma_x^n, \sigma_W, \gamma\right)$. Observe that this quantity only depends on our beliefs about slot machine x, and not on our beliefs about any other slot machines $y \neq x$. This is the key feature of any index policy. However, the index does depend on the problem parameters σ_W and γ. We find it convenient to write the index in terms of the variance rather than the precision, for reasons that will become clear below.

Gittins showed that $I_x^{Gitt,n}$ can be simplified using

$$I_x^{Gitt,n}\left(\bar{\mu}_x^n, \sigma_x^n, \sigma_W, \gamma\right) = \bar{\mu}_x^n + \sigma_W \cdot I_x^{Gitt,n}\left(0, \frac{\sigma_x^n}{\sigma_W}, 1, \gamma\right). \tag{5.19}$$

This equation is reminiscent of the well-known property of normal random variables. Just as any random variable can be written as a function of a standard normal random variable, so a Gittins index can be written in terms of a "standard normal" Gittins index, as long as we are using a normal-normal learning model.

For notational convenience, we can write

$$I_x^{Gitt,n}\left(0, \frac{\sigma_x^n}{\sigma_W}, 1, \gamma\right) = G\left(\frac{\sigma_x^n}{\sigma_W}, \gamma\right).$$

Thus, we only have to compute Gittins indices for fixed values of the prior mean and experimental noise, and then use (5.19) to translate it to our current beliefs. Table 5.3 gives the values of $G\left(s, \gamma\right)$ for $\gamma = 0.95, 0.99$ and $s = 1/\sqrt{k}$ for $k = 1, 2, ...$ This corresponds to a case where the experimental noise is equal to 1, and our beliefs about alternative x have the variance $1/k$ if we make k experiments of x.

The table reveals several interesting facts about Gittins indices. First, $G\left(1/\sqrt{k}, \gamma\right)$ is increasing in γ. If γ is larger, this means that we have a larger effective time horizon. Essentially, a larger portion of our time horizon "matters," more and more of the rewards we collect remain large enough after discounting to have a notable

	Discount factor					
k	0.5	0.7	0.9	0.95	0.99	0.995
1	0.2057	0.3691	0.7466	0.9956	1.5758	1.8175
2	0.1217	0.2224	0.4662	0.6343	1.0415	1.2157
3	0.0873	0.1614	0.3465	0.4781	0.8061	0.9493
4	0.0683	0.1272	0.2781	0.3878	0.6677	0.7919
5	0.0562	0.1052	0.2332	0.3281	0.5747	0.6857
6	0.0477	0.0898	0.2013	0.2852	0.5072	0.6082
7	0.0415	0.0784	0.1774	0.2528	0.4554	0.5487
8	0.0367	0.0696	0.1587	0.2274	0.4144	0.5013
9	0.0329	0.0626	0.1437	0.2069	0.3608	0.4624
10	0.0299	0.0569	0.1313	0.1899	0.3529	0.4299
20	0.0155	0.0298	0.0712	0.1058	0.2094	0.2615
30	0.0104	0.0202	0.0491	0.0739	0.1520	0.1927
40	0.0079	0.0153	0.0375	0.0570	0.1202	0.1542
50	0.0063	0.0123	0.0304	0.0464	0.0998	0.1292
60	0.0053	0.0103	0.0255	0.0392	0.0855	0.1115
70	0.0045	0.0089	0.0220	0.0339	0.0749	0.0983
80	0.0040	0.0078	0.0193	0.0299	0.0667	0.0881
90	0.0035	0.0069	0.0173	0.0267	0.0602	0.0798
100	0.0032	0.0062	0.0156	0.0242	0.0549	0.0730

Table 5.3 Gittins indices $G(s, \gamma)$ for the case where $s = 1/\sqrt{k}$, from Gittins (1989).

effect on our objective value. This means that we can afford to do more exploration, because one low reward early on will not harm our objective value as much. Hence, the Gittins indices are higher, encouraging us to explore more.

Second, $G\left(1/\sqrt{k}, \gamma\right)$ is decreasing in k. This is fairly intuitive. The standard Gittins index $G(s, \gamma)$ represents the uncertainty bonus, and does not depend on our estimate of the value of an alternative. As the variance of our beliefs goes down, the uncertainty bonus should go down as well, and we should only continue to measure the alternative if it provides a high reward.

Unfortunately, even the seminal work by Gittins does not give the values of $G(s, \gamma)$ for all possible s and γ. In general, computing these values is a difficult problem in and of itself. For this reason, a minor literature on Gittins approximation has arisen in the past ten years. To obtain a practical algorithm, it is necessary to examine this literature in more depth.

5.3.3 Approximating Gittins indices

Finding Gittins indices is somewhat like finding the cdf of the standard normal distribution. It cannot be done analytically, and requires instead a fairly tedious numerical calculation. We take for granted the existence of nice functions built into most programming languages for computing the cumulative standard normal distribution, for which extremely accurate polynomial approximations are available. In Excel, this is available using the function `NORMINV`.

As of this writing, such functions do not exist for Gittins indices. However, in the case of the normal-normal model, there is a reasonably good approximation that results in an easily computable algorithm. First, it can be shown that

$$G(s,\gamma) = \sqrt{-\log\gamma}\cdot b\left(-\frac{s^2}{\log\gamma}\right),$$

where the function b must be approximated. The best available approximation of Gittins indices is given by

$$\tilde{b}(s) = \begin{cases} \frac{s}{\sqrt{2}} & s \le \frac{1}{7}, \\ e^{-0.02645(\log s)^2 + 0.89106\log s - 0.4873} & \frac{1}{7} < s \le 100, \\ \sqrt{s}\left(2\log s - \log\log s - \log 16\pi\right)^{\frac{1}{2}} & s > 100. \end{cases}$$

Thus, the approximate version of (5.19) is

$$I_x^{Gitt,n} \approx \bar{\mu}_x^n + \sigma_W\sqrt{-\log\gamma}\cdot\tilde{b}\left(-\frac{\bar{\sigma}_x^{2,n}}{\sigma_W^2\log\gamma}\right). \tag{5.20}$$

Figure 5.1 gives us an idea of the quality of this approximation. In a few select cases where the exact Gittins indices are known (see Table 5.3), we can evaluate the approximation against the exact values. We see that the approximation gives the most error for small values of k, but steadily improves as k increases. The approximation for $\gamma = 0.9$ tends to be slightly more accurate than the one for $\gamma = 0.99$. In general, the approximation is more accurate for lower values of γ.

5.4 THE KNOWLEDGE GRADIENT FOR ONLINE LEARNING

The knowledge gradient approach that we introduced in Chapter 4 is a particularly simple and elegant strategy for collecting information. In the ranking and selection setting, it produces an easily computable algorithm. What is more, it can be adapted to handle correlated beliefs, as well as non-Gaussian learning models. A natural question, then, is whether it can be adapted for the multi-armed bandit problem, which is the online version of ranking and selection.

In this section, we develop a simple relationship between the knowledge gradient for offline and online settings, which also allows us to consider problems with correlated beliefs. We present a few experimental comparisons that seem to suggest that this works quite well for online problems. We then discuss applications to problems with non-normal belief models.

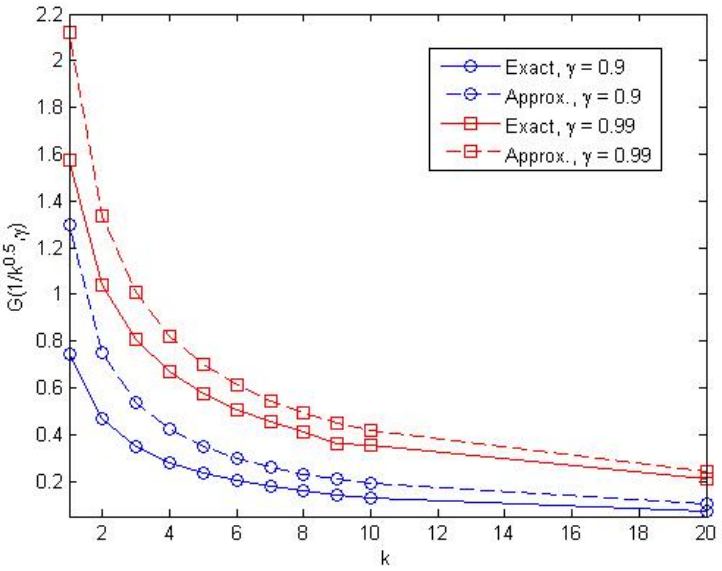


Figure 5.1 Comparison of approximate and exact Gittins indices for $\gamma = 0.9, 0.99$.

5.4.1 The basic idea

Once again, consider the normal-normal Bayesian learning model. Suppose that we can run N experiments, and that $\gamma = 1$. Furthermore, suppose that we have already run n experiments, and have constructed estimates $\bar{\mu}_x^n$ and β_x^n for each alternative x. Now, as a thought experiment, let us imagine that we will suddenly cease learning, starting at time n. We will still continue to collect rewards, but we will no longer be able to use the updating equations (3.2) and (3.3) to change our beliefs. We are stuck with our time-n beliefs until the end of the time horizon.

If this were to occur, the best course of action would be to choose $x^{n'} = \arg\max_x \bar{\mu}_x^n$ for all times $n \leq n' \leq N$. Since we cannot change our beliefs anymore, all we can really do is choose the alternative that seems to be the best, based on the information that we managed to collect up to this point. The expected total reward that we will collect by doing this, from time n to time N, is given by

$$V^{Stop,n}(S^n) = (N - n + 1) \max_x \bar{\mu}_x^n, \tag{5.21}$$

simply because there are $N - n + 1$ rewards left to collect. Because $\gamma = 1$, each reward is weighted equally. For instance, in the example given in Table 5.4, this quantity is $V^{Stop,n}(S^n) = 6 \cdot 5.5 = 33$.

Consider a different thought experiment. We are still at time n, but now our next decision will change our beliefs as usual. Assume we run experiment $x^n = x$ at time n, giving us updated estimates $\bar{\mu}_{x'}^{n+1}(x)$. However, starting at time $n + 1$, we will cease to learn, and from there on we will be in the situation described above. This means that, starting at time $n + 1$, we will always measure the alternative given by

$\arg\max_{x'} \bar{\mu}_{x'}^{n+1}(x)$. The problem thus reduces to choosing one single decision x^n to maximize the expected total reward we collect, starting at time n.

This idea is essentially the knowledge gradient concept from a slightly different point of view. In ranking and selection, we chose each decision to maximize the incremental improvement (obtained from a single experiment) in our estimate of the best value. Essentially, we treated each decision as if it were the last time we were allowed to learn. We made each decision in such a way as to get the most benefit out of that single experiment. In the online setting, we do the same thing, only "benefit" is now expressed in terms of the total reward that we can collect from time n to the end of the time horizon.

The KG decision for the bandit problem is given by

$$X^{KG,n} = \arg\max_x \mathbb{E}\left[\mu_x + V^{Stop,n+1}\left(S^{n+1}\right) \mid S^n, x^n = x\right] \tag{5.22}$$

$$= \arg\max_x \bar{\mu}_x^n + (N-n)\,\mathbb{E}\left[\max_{x'} \bar{\mu}_{x'}^{n+1}(x) \mid S^n, x^n = x\right] \tag{5.23}$$

$$= \arg\max_x \bar{\mu}_x^n + (N-n)\,\mathbb{E}\left[\max_{x'} \bar{\mu}_{x'}^{n+1}(x) - \max_{x'} \bar{\mu}_{x'}^n \mid S^n, x^n = x\right] \tag{5.24}$$

$$= \arg\max_x \bar{\mu}_x^n + (N-n)\,\nu_x^{KG,n}, \tag{5.25}$$

where $\nu_x^{KG,n}$ is simply the knowledge gradient for ranking and selection, given by (4.13). We start with the basic Bellman equation in (5.22). The downstream value is given by $V^{Stop,n+1}$ because we assume that we will cease to learn starting at time $n+1$. Next, we use the fact that $\mathbb{E}\left(\mu_x \mid S^n\right) = \bar{\mu}_x^n$, together with the definition of $V^{Stop,n+1}$ from (5.21) to obtain (5.23). Because the quantity $\max_{x'} \bar{\mu}_{x'}^n$ is constant given S^n, and does not depend on x, we can put it into the expression without changing the arg max, thus arriving at (5.24). Finally, we apply the definition of the knowledge gradient from (4.3) to obtain (5.25).

This line of reasoning has given us a simple and easily computable algorithm for the multi-armed bandit problem. At first, the expression $\bar{\mu}_x^n + (N-n)\,\nu_x^{KG,n}$ that we compute for alternative x may look very similar to the index policies we discussed earlier. Like interval estimation, Gittins indices, and other methods, KG takes $\bar{\mu}_x^n$ and adds an uncertainty bonus $(N-n)\,\nu_x^{KG,n}$. Just as in the other index policies, the uncertainty bonus gets smaller as σ_x^n gets smaller: thus, if the level of uncertainty is zero, the uncertainty bonus is zero as well. Furthermore, all other things being equal, the uncertainty bonus is larger if n is smaller, reflecting the fact that it is more important to learn in the early stages of the problem, while we have more remaining time steps in which we can potentially use the information we collect.

However, the KG policy is not an index policy. Crucially, the knowledge gradient $\nu_x^{KG,n}$ depends not only on $\bar{\mu}_x^n$, but also on $\max_{x' \neq x} \bar{\mu}_{x'}^n$. This cannot happen in an index policy, where the index of x is only allowed to depend on our beliefs about x. The knowledge gradient policy does not decompose the multi-armed bandit problem into many one-armed bandit problems. It considers each alternative relative to the others.

Table 5.4 shows the computations performed by the online KG policy for a particular problem with five alternatives and $N - n = 5$ experiments remaining in the time horizon. This example illustrates the distinction between the online KG policy and

Choice	$\bar{\mu}^n$	β^n	$\tilde{\sigma}^n$	ζ^n	$\nu^{KG,n}$	$\bar{\mu}^n + 5 \cdot \nu^{KG,n}$
1	2	1/2	1.1547	-3.0311	0.0004	2.0020
2	4	1/2	1.1547	-1.2990	0.0527	4.2634
3	3	2/3	0.9487	-2.6352	0.0012	3.0062
4	5.5	1/2	1.1547	-0.8660	0.1234	6.1168
5	4.5	1/3	1.5	-0.6667	0.2267	5.6333

Table 5.4 Calculations for the online KG policy in a bandit problem with $M = 5$, $N - n = 5$, and $\beta_x^W = 1$ for all x.

the offline KG policy from Chapter 4. If this were a ranking and selection problem, we would measure the alternative with the highest KG factor, namely alternative 5. However, even though alternative 4 has a smaller KG factor, our estimate $\bar{\mu}_4^n$ is sufficiently larger than $\bar{\mu}_5^n$ to make the online KG policy choose alternative 4. Thus, the online KG policy favors exploitation more than the offline KG policy.

At the same time, if $N - n = 50$ in the same example, then the online KG policy would prefer alternative 5 to alternative 4, thus agreeing with the offline KG policy. Unlike the offline KG policy, online KG is what is known as a *nonstationary policy*. This means that the decision made by online KG depends on n as well as on S^n. The exact same belief state can lead online KG to measure different alternatives depending on the current time.

The formulation of KG for bandit problems is quite versatile. Suppose that we have a discount factor $\gamma < 1$. It is a simple matter to repeat the above reasoning and arrive at the decision rule

$$X^{KG,n} = \arg\max_x \bar{\mu}_x^n + \gamma \frac{1 - \gamma^{N-n}}{1 - \gamma} \nu_x^{KG,n}.$$

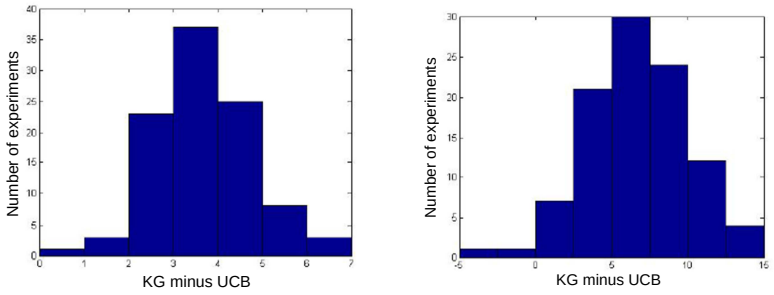
Taking $N \to \infty$, we obtain the knowledge gradient rule for infinite-horizon bandit problems,

$$X^{KG,n} = \arg\max_x \bar{\mu}_x^n + \frac{\gamma}{1 - \gamma} \nu_x^{KG,n}.$$

This is substantially easier to compute than Gittins indices. Keep in mind also that Gittins indices are only designed for infinite-horizon problems. If N is finite, the Gittins policy becomes a heuristic with γ serving as a tunable parameter. On the other hand, the KG policy can be defined for both finite- and infinite-horizon problems, and requires no tuning in either case. Of course, it is not an optimal policy, but it is able to respond to these different environments without the need for any tunable parameters.

5.4.2 Tunable variations

As with other policies, we can introduce tunable parameters that can produce better results when tuned for specific environments. We start with the tunable version of the knowledge gradient we first introduced for the offline (terminal reward) problem in section 4.3.3. There, we made the assumption that we would evaluate an alternative



(a) (b)

Figure 5.2 Histogram showing the average difference in the total discounted rewards collected by KG and approximate Gittins indices across 100 bandit problems with $M = 100$ and $\gamma = 0.9$. (a) uses truth from the prior, and (b) uses truth from an unbiased uniform distribution (from Ryzhov et al. 2011).

with precision $\theta_1 \beta^W$, where θ_1 is a positive integer that can be interpret as repeat factor which anticipates that we might perform the same experiment multiple times. Thus, our updated precision from an experiment would be modeled as

$$\beta_x^{n+1} = \beta_x^n + \theta_1 \beta^W.$$

We let $\nu_x^{KG,n}(\theta_1)$ be the offline KG formula using θ_1 as the repeat factor for the precision β^W. Then, we introduce a second parameter θ_2 which replaces the horizon $N - n$ in our online KG formula. Our tunable online KG formula is now given by

$$X^{OLKG}(\theta) = \arg\max_x \left(\bar{\mu}_x^n + \theta_2 \nu_x^{KG,n}(\theta_1)\right),$$

where $\theta = (\theta_1, \theta_2)$ captures our tunable parameters.

Tuning can have a major impact for problems where the noise of a measurement is quite high, and/or where the learning budget is relatively small.

5.4.3 Some experimental comparisons

It is useful to see how the knowledge gradient adapted for online problems compares against some of the popular policies that have been proposed for this problem class. Figure 5.2 shows a histogram of the performance of infinite-horizon KG minus the performance of a policy based on the Gittins approximation from Section 5.3.3 across 100 bandit problems with 100 alternatives. The numbers represent differences in the total discounted rewards (with $\gamma = 0.9$) collected by the two policies. We can see that all the numbers are positive, meaning that KG outperformed the approximate policy in every problem. To be sure, one can create problems where this is not the case, but it does indicate that KG can be competitive with the best existing approximation of the optimal policy.

The main advantage of the KG method, however, is that its nature as a non-index policy makes it well suited to the case where our beliefs about the alternatives are

tb

Comparison	Average difference	Standard error
KG minus approximate Gittins	0.7076	0.0997
KG minus interval estimation	-0.0912	0.0857
KG minus UCB	44.4305	0.6324
KG minus UCB1	1.2091	0.1020
KG minus pure exploitation	5.5413	0.1511

Table 5.5 Comparison between knowledge gradient and competing policies for 100 truth-from-prior experiments (from Ryzhov et al. 2011).

correlated. Suppose that we begin with a multivariate normal prior $\mu \sim N\left(\bar{\mu}^0, \Sigma^0\right)$, and use (2.18) and (2.19) to update our beliefs, but we keep the bandit objective function from (**??**). In this setting, index policies are inherently unsuitable: an index policy depends on our ability to decompose the problem and consider every alternative as if it were the only alternative in the problem, but the whole point of correlations is that the alternatives are inextricably related. Thus, while we can still use index policies such as Gittins and UCB as heuristics, they automatically lose their nice optimality properties in the correlated setting.

However, we can still define a knowledge gradient method in the correlated case. In fact, the KG decision rule is still given by (5.25), with the only change that we replace $\nu_x^{KG,n}$ by $h\left(\bar{\mu}^n, \tilde{\sigma}\left(\Sigma^n, x\right)\right)$ from (4.24). This quantity is then computed exactly as in Chapter 4. As of this writing, KG is the first algorithm that is able to consider bandit problems with multivariate normal priors.

Table 5.5 summarizes the results of a series of experiments on online problems with 100 alternatives with correlated beliefs. The knowledge gradient outperformed approximate Gittins, UCB, UCB1 and pure exploration for all 100 sample realizations. Only interval estimation proved to be competitive, and actually outperformed the knowledge gradient policy 77 percent of the time (although the difference in the average performance was not statistically significant). Recall that interval estimation uses the policy of maximizing the index

$$\nu^{IE,n} = \bar{\mu}_x^n + z_\alpha \sigma_x^n,$$

where $\bar{\mu}_x^n$ is our current estimate of the value of alternative x and σ_x^n is the standard deviation of our estimate $\bar{\mu}_x^n$.

The performance of interval estimation seems surprising, given that the knowledge gradient policy is taking advantage of a covariance structure (which we assume is known in advance), while IE has no such ability. However, IE has a tunable parameter z_α, and it is important that this parameter be tuned carefully. Figure 5.3 shows the behavior of interval estimation as a function of z_α, along with the knowledge gradient (which of course is a constant with respect to z_α). Note that IE outperforms KG only over a narrow range. Even more important, notice the severe degradation of IE as z_α moves away from its best setting.

This experiment demonstrates that there is a tremendous amount of information in a tunable parameter. Statements have been made in the literature that z_α can be safely

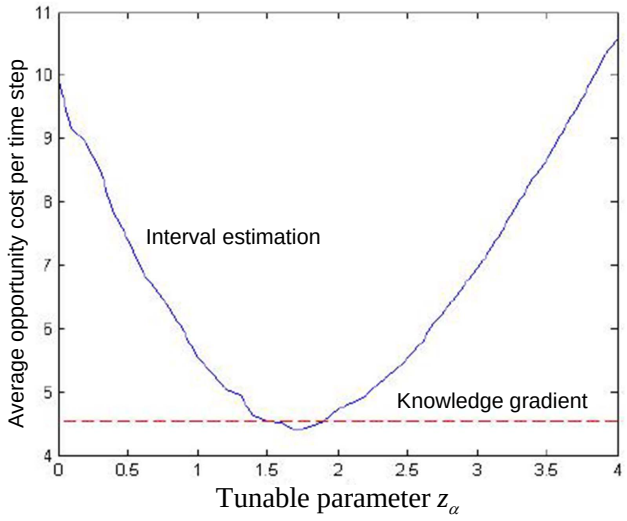


Figure 5.3 Expected opportunity cost for interval estimation as a function of z_α along with the knowledge gradient (from Ryzhov & Powell (2009*a*)).

chosen to be around 2 or 3, but problems have been found where the optimal value of z_α ranges anywhere from 0.5 to 5.0. Furthermore, the solution can be quite sensitive to the choice of z_α, suggesting that tuning has to be performed with care.

5.4.4 Non-normal models

Just as in ranking and selection, KG can also be extended to non-normal learning models, although we should take care to define the knowledge gradient in accordance with our reward structure. Suppose that we are working with the gamma-exponential model, where W_x follows an exponential distribution with parameter λ_x, and each λ_x has a gamma prior distribution with parameters a_x^0, b_x^0. Thus, our prior estimate of every reward is

$$\begin{aligned}\mathbb{E}\left(W_x\right) &= \mathbb{E}\left[\mathbb{E}\left(W_x \mid \lambda_x\right)\right] \\ &= \mathbb{E}\left[\frac{1}{\lambda_x}\right] \\ &= \frac{b_x^0}{a_x^0 - 1},\end{aligned}$$

where $b_x^0 > 0$ and $a_x^0 > 1$. Suppose that our objective function is to maximize the sum of the expected rewards,

$$\max_{\pi \in \Pi} \sum_{n=0}^{N} \gamma^n \frac{1}{\lambda_{x^n}}.$$

Then, the online KG decision rule for $\gamma = 1$ is given by

$$X^{KG,n} = \arg\max_x \frac{b_x^n}{a_x^n - 1} + (N - n)\, \nu_x^{KG,n}.$$

To write the knowledge gradient $\nu_x^{KG,n}$, let $C_x^n = \max_{x' \neq x} \frac{b_{x'}^n}{a_{x'}^n - 1}$ and define the baseline KG quantity

$$\tilde{\nu}_x^n = \frac{(b_x^n)^{a_x^n}}{(a_x^n C_x^n)^{a_x^n - 1}} \left(\frac{1}{a_x^n - 1} - \frac{1}{a_x^n} \right).$$

Then,

$$\nu_x^{KG,n} = \begin{cases} \tilde{\nu}_x^n & \text{if } x \neq \arg\max_{x'} \frac{b_{x'}^n}{a_{x'}^n - 1} \\ \tilde{\nu}_x^n - \left(\frac{b_x^n}{a_x^n - 1} - C_x^n \right) & \text{if } C_x^n \geq \frac{b_x^n}{a_x^n} \\ 0 & \text{otherwise.} \end{cases} \tag{5.26}$$

Once again, we have an easily computable KG algorithm, for a problem where Gittins approximations are not easily available. Note the presence of a penalty term in (5.26), much like in the expression for gamma-exponential KG in ranking and selection that was presented in Section 4.7.

5.5 VARIATIONS OF BANDIT PROBLEMS

One of the notable features of the bandit literature has been the introduction of a wide range of variations. The style of this community is to provide names of the form "XXX bandits" where XXX describes some variation, typically in the nature of the arms (choices), the exogenous information process, and/or the belief model. This style has produced a colorful literature full of creative adjectives, as illustrated in table 5.6. Below we provide more detailed descriptions on a few of the examples.

Restless bandits - The standard bandit model assumes that the truth μ_x for arm x remains the same over time. Restless bandits describe a model where the means are allowed to vary over time.

Arm acquiring bandits - These are problems where the number of alternatives grows over time.

Intermittent bandits - In this setting, not all the arms are available to be tested. A related variation is known as "sleeping bandits."

Continuous armed bandits - While the standard bandit problem involves discrete alternatives $x \in \mathcal{X} = \{x_1, \ldots, x_M\}$, some authors have considered the case where the choices are continuous.

Response-surface bandits - This is an instance of a bandit problem where the adjective is describing the belief model. While the most standard belief model is a lookup table, some authors have considered parametric models ("response surface"). The term linear bandit has also been used for this purpose.

Finite horizon bandits - While the classic problem assumes an infinite horizon, some authors have worked on finite-horizon problems.

Dueling or adversarial bandits - Here we have two agents where one is requesting information (choosing an arm to test), while another is providing information specifically to produce worse results for the first bandit.

Contextual bandits - This describes a setting where we are given exogenous information (call it S^c) that influences the underlying distributions. Instead of learning the best arm x^*, we want to learn the best arm $x^*(S^c)$ in the "context" of the new information S^c.

Despite the popularity of multiarmed bandit problems, there does not seem to be a formal definition of a bandit problem. The original setting of a problem with discrete alternatives where rewards are accumulated while learning is happening has been generalized to problems with continuous alternatives, and for problems where we are only interested in the performance after all experiments have been completed.

Bandit problems appear to be any problems which involve an exploration vs. exploitation tradeoff, which is a property of the policy (or search algorithm), rather than the problem. We note that exploration-exploitation tradeoffs occur with any problem where we maximize cumulative rewards (this applies to both finite and infinite horizon problems), as well as any finite horizon problems where we are optimizing the final reward. A key feature of any procedure that manages an exploration-exploitation tradeoff is the use of a belief model, which is the only way to learn.

5.6 EVALUATING POLICIES

In chapter 3, we introduced a series of heuristic policies such as Boltzmann exploration or interval estimation, but we also introduced a tunable form of upper confidence bounding, with more variations in this chapter. In section 4.3.3, we even introduced a tunable form of the knowledge gradient for problems where the value of information is nonconcave.

Any tunable policy can be tuned for either offline (terminal reward) or online (cumulative reward) settings. Let $X^\pi(S|\theta)$ be a policy where S is our belief state and θ is some tunable parameter (there could be more than one). If we are in an offline setting, we can use $X^\pi(S|\theta)$ to guide a sequence of experiments $x^0, W^1, x^1, W^2, \ldots, x^{N-1}, W^N)$ that produces a set of estimates $\bar{\mu}_x^N$ from which we can identify the best alternative (or design)

$$x^{\pi,N}(\theta) = \arg\max_x \bar{\mu}_x^N,$$

where we retain the explicit dependence on θ. We can then optimize θ by solving (approximately) the optimization problem

$$\max_\theta \mathbb{E}_\mu \mathbb{E}_{W^1,\ldots,W^N|\mu} \mu_{x^{\pi,N}}, \tag{5.27}$$

where we are assuming that we can use our known μ (known only to our simulator) to evaluate the performance of the policy. This is the same as our policy search problem we posed in equation (5.1) at the beginning of the chapter, except that now we are writing it explicitly in terms of the tunable parameter θ.

Bandit problem	Description
Multiarmed bandits	Basic problem with discrete alternatives, online (cumulative regret) learning, lookup table belief model with independent beliefs
Restless bandits	Truth evolves exogenously over time
Adversarial bandits	Distributions from which rewards are being sampled can be set arbitrarily by an adversary
Continuum-armed bandits	Arms are continuous
X-armed bandits	Arms are a general topological space
Contextual bandits	Exogenous state is revealed which affects the distribution of rewards
Dueling bandits	The agent gets a relative feedback of the arms as opposed to absolute feedback
Arm-acquiring bandits	New machines arrive over time
Intermittent bandits	Arms are not always available
Response surface bandits	Belief model is a response surface (typically a linear model)
Linear bandits	Belief is a linear model
Dependent bandits	A form of correlated beliefs
Finite horizon bandits	Finite-horizon form of the classical infinite horizon multi-armed bandit problem
Parametric bandits	Beliefs about arms are described by a parametric belief model
Nonparametric bandits	Bandits with nonparametric belief models
Graph-structured bandits	Feedback from neighbors on graph instead of single arm
Extreme bandits	Optimize the maximum of recieved rewards
Quantile-based bandits	The arms are evaluated in terms of a specified quantile
Preference-based bandits	Find the correct ordering of arms
Best-arm bandits	Identify the optimal arm with the largest confidence given a fixed budget

Table 5.6 A sample of the growing population of "bandit" problems.

We can apply the same idea to any online problem by simply changing the objective function to capture the cumulative reward, giving us

$$\max_{\theta} \mathbb{E}_{\mu}\mathbb{E}_{W^1,\ldots,W^N|\mu} \sum_{n=0}^{N-1} \mu_{X^\pi(S^n)}. \tag{5.28}$$

With this observation in hand, it is useful to take a closer look at the policies themselves to contrast how they might behave in offline and online learning settings. We begin by noting that most of our policies, including interval estimation, the UCB

policies, Gittins indices and the knowledge gradient for online learning, all have the basic structure

$$X^{\pi}(S) = \arg\max \left(\bar{\mu}_x^n + \theta U^B(S)\right), \tag{5.29}$$

where U^B is some term that is often referred to as the "uncertainty bonus." Examples of policies that have this structure include:

Interval estimation:
$X^{IE}(S|\theta) = \arg\max_x \left(\bar{\mu}_x^n + \theta\bar{\sigma}_x^n\right)$

Upper confidence bounding (UCB1 variation):
$X^{UCB1}(S|\theta) = \arg\max_x \left(\bar{\mu}_x^n + \theta\sqrt{\frac{\log n}{N_x^n}}\right)$

Upper confidence bounding (UCBE variation):
$X^{UCBE,n}(\theta) = \arg\max_x \left(\bar{\mu}_x^n + \theta\sqrt{\frac{1}{N_x^n}}\right)$. Upper confidence bounding (UCBE variation):
$X^{UCB}(S|\theta) = \arg\max_x \left(\bar{\mu}_x^n + \theta\sqrt{\frac{\log n}{N_x^n}}\right)$

Gittins indices:
$X^{Gittins}(S|\theta) = \arg\max_x \left(\bar{\mu}_x^n + \theta\sigma^W\right)$

Online knowledge gradient:
$X^{OLKG}(S|\theta) = \arg\max_x \left(\bar{\mu}_x^n + \theta\nu_x^{KG}\right)$

Each of these has been presented in its tunable form. Each strikes a balance between the estimated reward from trying experiment x, given by $\bar{\mu}_x^n$, and a measure of the uncertainty in this estimate. The parameter θ then strikes the balance between exploitation (which emphasizes $\bar{\mu}_x^n$), and exploration, captured by the uncertainty bonus.

We note, however, that these policies are all designed for online learning, since it is only in this setting that $\bar{\mu}_x^n$ is actually our best estimate of what we will receive by choosing $x^n = x$, with the uncertainty bonus encouraging exploration. If our budget is just one experiment, then the optimal $\theta = 0$.

By contrast, a value of information policy such as the knowledge gradient or expected improvement is specifically designed for offline learning, since it only captures the value of information. Recall that the offline knowledge gradient is given by

$$\nu_x^{KG,n} = \mathbb{E}\left\{V^{n+1}(S^{n+1}(x))|S^n\right\} - V^n(S^n).$$

Note that $\bar{\mu}_x^n$ is missing, as it should be, since in an offline setting, we do not care how well an experiment might perform. We only care about what we learn, that contributes to our ability to identify the best choices at the end.

Given this observation, how is it that these policies can be tuned for offline learning (and work reasonably well)? First, they all capture the fundamental tradeoff between exploitation (captured by $\bar{\mu}_x^n$ in the policy), and exploration (captured by some form of uncertainty bonus). However, these are heuristic policies, which is why they all have a tunable parameter. This also hints at the power of tuning.

5.7 BIBLIOGRAPHIC NOTES

Section 5.3 - Gittins index theory is due to Gittins & Jones (1974), Gittins (1979) and Gittins (1989). Berry & Fristedt (1985) also provides a rigorous analysis of bandit problems. This research has launched an entire field of research into the search for index policies for variations on the basic bandit problems. Glazebrook (1982) analyzes policies for variations of the basic bandit model. Glazebrook & Minty (2009) presents a generalized index for bandit problems with general constraints on information collection resources. Bertsimas & Nino-Mora (2000) show how an index policy can be computed using linear programming for a certain class of bandit problems. See the updated version of Gittins' 1989 book, Gittins et al. (2011), for a modern treatment of bandit problems and a much more thorough treatment of this extensive literature. The approximation of Gittins indices is due to Chick & Gans (2009), building on the diffusion approximation of Brezzi & Lai (2002).

Section 5.2 - Lai & Robbins (1985) and Lai (1987) provide the seminal research that shows that the number of times an upper confidence bound policy chooses a particular suboptimal machine is on the order of $\log N$, and that this is the best possible bound. Auer et al. (2002) derives finite-time regret bounds on the UCB1 and UCB1-Normal policies and reports on comparisons against variations of UCB policies and epsilon-greedy on some small problems (up to 10 arms). The UCB policy for exponential rewards comes from Agrawal (1995).

Section 5.4 - The online adaptation of the knowledge gradient is due to Ryzhov et al. (2011). Some additional experimental comparisons can be found in Ryzhov & Powell (2009*b*). Ryzhov & Powell (2011*c*) presents the KG policy for the gamma-exponential model. Rates of convergence for KG-type policies are still an open question, but Bull (2011) is an interesting first step in this direction.

PROBLEMS

5.1 You have three materials, A, B and C that you want to test for their ability to convert solar energy to electricity, and you wish to find which one produces the highest efficiency, which you have to do by learning from field implementations (in other words, online learning). Table 5.7 shows your initial beliefs (which we assume are independent) summarized as the mean and precision. Your prior belief is normal, and testing alternative x produces an observation W_x which is normally distributed with precision $\beta^W = 1$.

a) You follow some learning policy that has you first evaluating A, then B and finally C, obtaining the observations W_x^n shown in Table 5.7 (for example, $W_A^1 = 3$). Give the updated belief (mean and precision) for $\bar{\mu}_A^1$ given the observation $W_A^1 = 3$. Also compute the updated means only (not the precisions) for $\bar{\mu}_B^2$ and $\bar{\mu}_C^3$.

b) Give the objective function (algebraically) to find the best policy after N experiments if this is an off-line learning problem. Compute a sample realization of the objective function for this example.

Iteration	A	B	C
Prior (μ_x, β_x)	(5,.05)	(3,.02)	(4,.01)
1	3	-	-
2	-	2	-
3	-	-	6

Table 5.7 Three observations, for three alternatives, given a normally distributed belief, and assuming normally distributed observations.

c) Give the objective function (algebraically) to find the best policy if this is an on-line learning problem. Compute a sample realization of the objective function for this example.

5.2 Consider a classic multi-armed bandit problem with normally distributed rewards.

a) Is the multi-armed bandit problem an example of an on-line or off-line learning problem?

b) Let $R^n(x^n)$ be the random variable giving the reward from measuring bandit $x^n \in (1, 2, \ldots, M)$ in iteration n. Give the objective function we are trying to maximize (define any other parameters you may need).

c) Let $\Gamma(n)$ be the Gittins index when rewards are normally distributed with mean 0 and variance 1, and let $\nu_x(\mu_x, \sigma_x^2)$ be the Gittins index for a bandit where the mean reward is μ with variance σ^2. Write $\nu_x(\mu_x, \sigma_x^2)$ as a function of $\Gamma(n)$.

5.3 Consider a bandit problem with $\gamma < 1$ where we use the beta-Bernoulli learning model.

a) Suppose that, for a particular choice of α, β and r, we have $V(\alpha, \beta, r) \approx V(\alpha + 1, \beta, r) \approx V(\alpha, \beta + 1, r)$. Show that the Gittins recursion is solved by

$$V(\alpha, \beta, r) = \frac{1}{1-\gamma} \max\left(r, \frac{\alpha}{\alpha+\beta}\right).$$

b) In a spreadsheet, choose values for r and γ (these should be stored in two cells of the spreadsheet, so that we can vary them), and create a table that compute the values of $V(\alpha, \beta, r)$ for all $\alpha, \beta = 1, 2, ...$ with $\alpha + \beta < 200$. When $\alpha + \beta = 200$, use $V(\alpha, \beta, r) = \frac{1}{1-\gamma}\frac{\alpha}{\alpha+\beta}$ as a terminal condition for the recursion.

c) The spreadsheet from part b) can now be used to compute Gittins indices. The Gittins index r^* for a particular α and β with $\alpha + \beta < 200$ is the smallest value of r for which $\frac{r}{1-\gamma}$ is equal to the entry for $V(\alpha, \beta, r)$ in the table. Use trial and error to find r^* for $\alpha, \beta = 1, ..., 5$ with $\gamma = 0.9$. Report the values you find, and compare them to the exact values of the Gittins indices in Table 5.1.

5.4 Consider a bandit problem with $\gamma < 1$. Repeat the derivation from Section 5.4 to show that the KG decision rule is given by

$$X^{KG,n} = \arg\max_x \bar{\mu}_x^n + \gamma \frac{1-\gamma^{N-n}}{1-\gamma} \nu_x^{KG,n}$$

for finite N.

5.5 Consider a finite-horizon bandit problem with $\gamma = 1$ and a gamma-exponential learning model. Show that $\nu_x^{KG,n}$ is given by (5.26).

5.6 This exercise needs the spreadsheet:

`http://optimallearning.princeton.edu/exercises/FiveAlternative.xls`

available on the optimal learning web site. You are going to have to construct a learning policy to choose the best of five options, using the problems that are described in the attached spreadsheet. You are welcome to solve the problem directly in the accompanying spreadsheet. But this is an exercise that will be easier for some of you to solve using a programming environment such as MATLAB, Java or perhaps visual basic in Excel.

The spreadsheet illustrates the calculations. Each time you choose a path, the spreadsheet will show you the time for the path. It is up to you to update your estimate of the average travel time and the variance of the estimate. Use a Bayesian framework for this exercise. You can repeat the exercise different times on the same set of random realizations. If you wish to use a fresh set of random numbers, hit the "Refresh button." You can see the data on the "Data" tab. The data tab uses the data in columns A-F, which will not change until you hit the refresh button. The data in columns H-L use the Rand() function, and will change each time there is a recompute (which can be annoying). If you click on the cells in columns H-L, you will see how the random numbers are being generated. The true means are in row 2, and the numbers in row 3 control the spread. You should use a prior estimate of the standard deviation equal to 10 for all your analyses.

The problem set requires testing a number of exploration policies. For each policy, compute two objective functions (averaged over 100 random number seeds):

1) The online objective function, which is the discounted sum of your performance (for the chosen option) over all 100 experiments, with a discount factor of $\gamma = 0.80$. If $C^n(\omega)$ is the observed value of the measured option for the n^{th} experiment for random number seed ω, your objective function would be:

$$F^\pi = \frac{1}{100} \sum_{\omega=1}^{100} \sum_{n=0}^{100} \gamma^n C^n(\omega)$$

2) The final experiment:

$$G^\pi = \frac{1}{100} \sum_{\omega=1}^{100} C^{100}(\omega)$$

F^π is our online objective function, while G^π is our offline objective function. Also let

$$\begin{aligned} F &= \sum_{n=0}^{100} \gamma^n \bar{\mu}^* \\ G &= \bar{\mu}^* \end{aligned}$$

where $\bar{\mu}^*$ is the true mean for the best choice, if we knew the true means. Where necessary, you may assume that the standard deviation of an experiment is 10. You have to test the following policies:

1) Pure exploitation.
2) Boltzmann exploration, using scaling factor $\theta = 1$.
3) Epsilon-greedy exploration, where the exploration probability is given by $1/n$.
4) Interval estimation. Test the performance of $z_\alpha = 1.0, 2.0, 3.0$ and 4.0 and select the one that performs the best.
5) Gittins indices (use the numerical approximation of Gittins indices given in Section 5.3.3.

Do the following

a) In a graph, report F^π and G^π for each of the policies below. Also show F and G to provide a measure of how well we are doing.

b) Discuss your results. Compare the performance of each policy in terms of the two different objective functions.

5.7 Consider a multi-armed bandit problem with independent normal rewards. In this exercise, you will implement a few online policies.

a) How should the opportunity cost be expressed in the online problem?

b) In exercise 3.4 you implemented the interval estimation policy in an offline setting. Now take your code from before, and adapt it to online problems by changing the objective function appropriately. The parameters of the problem should be the same as before (use the same priors and assume $\gamma = 1$), but now you need to compute opportunity cost differently. How does the best value of the tunable parameter z_α change when the problem becomes online? After you tune z_α, report the confidence interval for the opportunity cost using 200 simulations.

c) Now implement the Gittins index approximation for the independent normal-normal model. You cannot solve the Gittins recursion – just use the approximation function $\tilde{b}$. Our problem has a finite horizon, so you can treat the parameter γ in the Gittins index calculation as another tunable parameter. What value of γ gives you the best results?

d) Now implement the online KG policy. Compare the confidence intervals for KG, Gittins, and interval estimation.

CHAPTER 6

ELEMENTS OF A LEARNING PROBLEM

By now we have covered some elementary classes of learning problems. Fortunately, these describe a very broad class of applications. But there are some important problem classes that we have not yet discussed, and which we cannot solve using the tools presented so far. However, before we launch into what can seem to be a list of scattered applications, it is useful to lay out a more comprehensive modeling framework that at least hints at how we might deal with the major classes of unsolved problems. This presentation will motivate the problems we consider in the remainder of the volume, but will also serve potentially as areas for further research.

The dimensions of any stochastic, dynamic problem can be organized along five core dimensions:

1) States variables - For learning problems, these capture our belief about unknown parameters.

2) Actions (or decisions or controls) - These are the experimental choices we can make.

3) Exogenous information - This is the new information we obtain as the result of an experiment.

4) The transition function - For learning problems, these are the equations that update our beliefs given the results of an experiment.

5) The objective function - This captures our performance metric, and is the basis for how we evaluate a learning policy.

This presentation will provide a general framework that allows us to approach virtually any sequential decision problem, and certainly any learning problem. This will lay the foundation for the more complex settings we are going to address later in this book.

6.1 THE STATES OF OUR SYSTEM

The state S_t (or S^n) of a system can be described as consisting of all the information needed from history to model the system from time t (or experiment n) onward. This means it has the information we need to make a decision, compute our performance metrics, and compute the transition to the next state. For example, the transition equations for updating beliefs

make a decision, compute the objective (contributions and rewards), and compute the transition to the next state. Elements of a state variable can include the following types of information:

- The physical state - These are variables that describe the locations, quantities and status of people, equipment and goods. In most applications, decisions directly impact the physical state.
- The information state - The information state captures other information needed to model the problem. In most applications, this information arrives exogenously to the system, such as weather, prices, customer demands or changes in technology.
- The belief state - This captures our distribution of belief about quantities that we do not know perfectly. For example, if we are using an independent normal-normal model, the belief state consists of a vector of means and a vector of variances (or precisions). For pure learning problems, the belief state is the only state variable, but later we consider more general problems.

We might designate the state as S_t if it is the information available before we make a decision x_t at time t after which we learn W_{t+1} between t and $t+1$. Or we might use S^n as the state after the nth experiment which is used to make decision x^n, which are the choices needed to run the $n+1st$ experiment. Either way, S_t or S^n is known as the *pre-decision state*.

There are times when it is useful to define the post-decision state S_t^x (or $S^{x,n}$), which is the state immediately after we make a decision, but before any new information arrives. The key difference is that S_t^x no longer includes any information needed to make the decision x_t (or x^n). Instead, S_t^x only carries the information that is needed at time $t+1$, which would always include our belief state.

Up to now, our problems (ranking and selection and the bandit problem) have been characterized by just a belief state, which is used only by the policy. To help with this distinction, we identify two classes of functions (or problems):

State-independent functions These are problems where the function we are optimizing does not depend on our state variable. For example, imagine that we are trying to optimize a newsvendor problem:

$$\max_x \mathbb{E}F(x,W) = \mathbb{E}[p\min(x,W) - cx].$$

Assume that we do not know the distribution of W, but we are allowed to observe actual sales after each iteration. With each iteration, we learn more about the function $\mu_x = \mathbb{E}F(x,W)$, but this information is used only to make a decision. After day t, we would write our (Bayesian) belief about the function $\mathbb{E}F(x,W)$ as

$$S_t = B_t = (\bar{\mu}_{tx}, \beta_{tx})_{x\in\mathcal{X}}.$$

State-dependent functions Now imagine that the prices change from day to day, and that on day t, we are first shown the price p_t before we have to make our decision where we wish to solve

$$\max_x \mathbb{E}\{F(x,W)|p_t\} = \mathbb{E}[p_t\min(x,W) - cx].$$

For this version, our state variable includes both p_t as well as our beliefs $B_t = (\bar{\mu}_{tx}, \beta_{tx})_{x\in\mathcal{X}}$ about the function $\mathbb{E}\{F(x,W)|p_t\}$, so we would write our state as

$$S_t = (p_t, B_t).$$

Some call this a *contextual learning problem* (in some communities this is known as a *contextual bandit problem*) since we have to learn the belief state B_t in the "context" of the price p_t. If p_t is independent of p_{t-1}, then the post-decision state would be

$$S_t^x = B_t,$$

since the belief state does not change until we have observed the outcome of the next experiment, and p_t is irrelevant at time $t+1$. However, we might have a price process that evolves according to

$$p_{t+1} = \theta_0 p_t + \theta_1 p_{t-1} + \theta_2 p_{t-2} + \varepsilon_{t+1}.$$

In this case, our pre-decision state would be

$$S_t = ((p_t, p_{t-1}, p_{t-2}), B_t).$$

The post-decision state would then be

$$S_t^x = ((p_t, p_{t-1}), B_t),$$

since we no longer need p_{t-2} when we advance to time $t+1$.

We could also introduce a physical state by assuming that leftover newspapers are held until tomorrow. Let R_t be the newspapers on hand at the beginning of day t, where

$$R_{t+1} = R_t + x_t - \min(x_t + R_t, W_{t+1}).$$

Returning to the setting where p_t does not depend on history, our state variable now includes both the physical state R_t, and the price p_t, as well as our belief state B_t about the function:

$$\max_x \mathbb{E}\{F(x,W)|p_t\} = \mathbb{E}\{p_t \min(x+R_t,W) - cx\}.$$

Thus, our (pre-decision) state would be

$$S_t = (R_t, p_t, B_t),$$

where B_t is our belief state as it was above.

State-independent functions represent pure learning problems, which is the focus of this book. State-dependent functions are inherently more difficult to solve than standard learning problems. For example, consider the easiest state-dependent problem, where we do not have a physical state R_t, and where p_t does not depend on history. In this setting, the price p_t represents the "context," and it means that we have to design a policy $X^\pi(B_t|p_t)$, whereas in the past we only had to design $X^\pi(B_t)$. Think of the contextual problem as having to design a learning policy $X^\pi(B_t|p_t)$ for each price p_t.

If we introduce a physical state R_t where R_{t+1} depends on x_t, then our policies need to capture the impact of a decision x_t on the future state. The types of learning policies we have seen up to now will not work well in this setting.

Problems with a physical state are quite complex, and generally beyond the scope of this book. We only revisit problems with a physical state at the end of the book.

Most of our attention focuses on problems with a pure belief state. There are three important classes of belief states:

Stationary distribution This is the primary focus of this volume, and describes problems with an unknown parameter characterized by a distribution, where we are trying to reduce the uncertainty by collecting information, typically from noisy experiments.

Nonstationary distribution In this setting, the unknown parameter is changing exogenously over time, which means that our distribution of belief about these parameters will exhibit increasing variance over time. This represents the problems known as restless bandits that were introduced in section 5.5.

Controllable distribution The most difficult problem class is where the uncertainty is controlled (or at least influenced) by decisions. For example, we may be trying to minimize the blood sugar of a patient through the choice of type of medication (which is a decision) that affects how the patient responds to different types of food (in an uncertain way). Alternatively, an athlete (or team) will improve with practice.

We have already seen problems with correlations in the belief model. This is where $Cov(\mu_x, \mu_{x'}) \neq 0$. We have only begun to explore the depth and richness of covariance structures for correlated beliefs. Correlated beliefs allow us to address problems with large (or infinite) numbers of potential alternatives with relatively small experimental budgets. Some examples of correlated beliefs include:

- Evaluations are made of a continuous (and possibly scalar) function. We might be sampling disease within a population, the response due to a particular drug dosage, or the demand response to the price of a product. The results of experiments are correlated inversely proportional to the distance between two experiments.
- Evaluations of multiattribute entities. We might be predicting the importance of a document (based on the attributes of the document) or the likelihood that someone will default on a loan (as a function of an individual's financial characteristics).
- Estimating the time on a path in a network. The observed travel time over one path will be correlated with other paths that share common links.
- Effectiveness of a drug compound. Different drug compounds will share common atomic subsequences which interact and determine the effectiveness of a drug. Drug compounds sharing common atoms (typically in specific locations) may exhibit correlations in their effectiveness.

The distinction between a physical (or resource) state and a belief (or knowledge) state has caused confusion in the dynamic programming community since the 1950s. It is common for people to equate "state" and "physical state," which is problematic when we only have a belief state. When it became clear that our belief about parameters could also evolve, Bellman & Kalaba (1959) used the term "hyperstate" to refer to the combination of physical state (an imperfect term) and belief state. In Chapter 15 we address the problem of learning in the presence of a physical state. Rather than use terms like hyperstate, we prefer to use terms like physical (or resource) state, information state and belief state.

6.2 TYPES OF DECISIONS

The complexity of a problem depends in large part on the nature of the decision we have to make in order to make an experiment. Major problem classes include:

- Binary decisions - We can continue to collect information, or stop; we can decide to show a document to an expert, or not; or we can switch to a new web design, or stay with our existing one (known as A/B testing).
- Discrete choice - Here, we have a set of discrete choices (not too large - dozens, hundreds, perhaps thousands), where at each point in time we have to make a decision to collect information about one of the choices. A discrete choice could be a person to do a job, a technology, a drug compound or a path through a network. So far, we have considered only discrete choice problems under the heading of ranking and selection or bandit problems.
- A scalar, continuous variable - We have to choose a quantity, price, location of a facility or concentration of a chemical that we need to optimize.
- A vector, where the elements can come in several flavors:
    a) Binary - 0's and 1's, which we would use to choose subsets or portfolios of discrete items.

b) Reals - We might have to set a multidimensional set of continuous parameters, such as testing a series of settings on a physical device such as an engine, or parameters governing the performance of a simulator.

c) Integers - Similarly, we may have a vector of discrete variables (but other than 0 or 1), such as testing an allocation of ambulances or locomotives, or a mix of different types of aircraft.

d) Multiattribute - We might choose a complex item (a document, a molecule, a person) characterized by a vector of categorical attributes.

We let x represent a generic "decision." We might have $x \in (0, 1)$, or $x = (1, 2, \ldots, M)$, or $x = (x_d)_{d \in \mathcal{D}}$ where $d \in \mathcal{D}$ is a type of decision ("fly to Chicago," "try a particular drug compound") where x_d can be binary, discrete or continuous. We let $\mathcal{X}$ be the set of feasible decisions. It is very important from a computational perspective to understand the nature of x, since there are problems where we assume that we can easily enumerate the elements of $\mathcal{X}$. In the list above, we open the door to problem classes where the size of $\mathcal{X}$ is infinite (if x is continuous) or much too large to enumerate.

A different dimension arises when we separate experimental decisions from implementation decisions. Some examples include:

- We may fund research (the experiment) to build a device (the implementation).
- We may observe the performance of links in a network (the experiment) to choose a path (implementation).
- We may wish to visit a company to talk to its management team (the experiment) before deciding to invest in the company (implementation).

For online learning problems, we learn as we go, so the only decisions are implementation decisions, but we can also learn from these. For offline learning problems, we run a series of experiments where x represent experimental design decisions. We have assumed up to now that after testing different $x \in \mathcal{X}$, that we choose one x as our final design, which means that $\mathcal{X}$ is also the set of implementation decisions. However, this is not always the case.

6.3 EXOGENOUS INFORMATION

Exogenous information clearly comes in a wide variety depending on the application. We briefly survey a few common types:

- Stationary, uncorrelated processes - This is the simplest problem, where observations come from a stationary distribution (with unknown mean, and potentially unknown variance), and where observations of different alternatives are uncorrelated.
- Nonstationary processes - We may make an observation of the level of disease in the population, but this can clearly change over time, but it changes in an exogenous way that we do not control.

- Correlated beliefs - We might wish to simulate multiple configurations using what are known as common random variables, which means that we use the same sample realizations to help control statistical error. It is important to separate correlated beliefs (where W_x^n is correlated with $W_{x'}^n$) and correlated beliefs (where $Cov(\mu_x, \mu_{x'}) > 0$).
- Learning processes - We might have a nonstationary process, but one that improves (or potentially deteriorates) as we sample it, rather than through some exogenous processes. For example, imagine observing how well a baseball player can hit; as we observe the player, his hitting may actually improve.
- State and/or action dependent processes - We generally view exogenous information as dependent on an unknown truth, but exogenous information may be a function of the state (this happens with learning processes described above) and possibly even the action (if we decide to sell stock or expensive equipment, this can depress the market price for these assets). When this happens, we will replace our conditional expectation $\mathbb{E}_{W^1,\ldots,W^N|\mu}\ldots$ with $\mathbb{E}^{\pi}_{W^1,\ldots,W^N|\mu}\ldots$, since the observations W^n depend on the policy $X^{\pi}(S^n)$.

All of these generalizations represent interesting extensions of our learning model.

6.4 TRANSITION FUNCTIONS

If there is a physical state R^n, we are going to assume we are given a function that describes how it evolves over time. We write it as

$$R^{n+1} = R^M(R^n, x^n, W^{n+1}).$$

The notation $R^M(\cdot)$ represents the "resource transition function." In traditional dynamic models (where the belief state is held constant), this would be the transition function, system model or simply "the model." For our purposes, we do not need to get into the details of this function, which can vary widely.

We assume that with each decision x^n, we learn something that allows us to update our belief state. We represent this generically using

$$B^{n+1} = B^M(B^n, x^n, W^{n+1}).$$

For example, $B^M(\cdot)$ could represent the Bayesian updating equations (3.2) and (3.3) in a ranking and selection problem. When we want to be really compact, we write the state variable as $S^n = (R^n, B^n)$ and represent its transition function using

$$S^{n+1} = S^M(S^n, x^n, W^{n+1}).$$

Elsewhere in this volume, we have been using the function $S^M(\cdot)$ to represent the updating of the belief state when there is no physical state because this notation is more familiar and there is no ambiguity. We suggest using the notation $B^M(\cdot)$ in situations where there is both a physical and a belief state, and when we want to refer specifically to the updating of the belief state.

Most of this volume focuses on problems without a physical state. Chapter 15 addresses learning in the presence of a physical state.

6.5 OBJECTIVE FUNCTIONS

There are several dimensions to the design of an objective function:

1) Are we focusing on the best design (terminal reward) or maximizing performance along the way (cumulative reward)? Terminal reward problems tend to arise in a laboratory or test setting (sometimes called offline learning) while cumulative reward problems arise when we have to learn in the field (also known as online learning).

2) Different types of experimental costs.

3) Performance metrics and policy evaluation.

6.5.1 Terminal vs. cumulative reward

Chapters 3 and 4 started with the types of learning problems that arise in a laboratory, computer simulator or a test environment, where we care only about the quality of the final design rather than how well we do while learning this final design. Since this objective typically arises in a testing environment, it is often referred to as offline learning.

Chapter 5 then shifted to learning in the field, a setting that is sometimes referred to as online learning. Here, we have to pay attention to how well we are doing while we are learning, which means we want to maximize the cumulative reward.

Below, we briefly sketch how offline (terminal reward) and online (cumulative reward) objectives might be structured.

Offline learning (terminal reward)

Up to now, we have represented our truth about design x using an unknown μ_x. We can use our learning policy $X^\pi(S)$ to generate decisions about what experiment to perform next, giving us the sequence

$$(S^0, x^0, W^1_{x^0}, S^1, x^1, W^2_{x^1}, \ldots, x^{N-1}, W^N_{x^{N-1}}, S^N)$$

where $x^n = X^\pi(S^n)$. From the sequence $W^1_{x^0}, W^2_{x^1}, \ldots, W^N_{x^{N-1}}$, we obtain estimates $\bar{\mu}^{\pi,N}_x$, from which we find our best design

$$x^{\pi,N} = \arg\max_{x \in \mathcal{X}} \bar{\mu}^{\pi,N}_x.$$

We can then evaluate the performance of our policy using

$$F^\pi = \mathbb{E}_\mu \mathbb{E}_{W^1,\ldots,W^N|\mu} \bar{\mu}^{\pi,N}_{x^{\pi,N}}.$$

In a simulator where we simulate μ_x from the prior $N(\mu^0_x, \beta^0_x)$, we can use our simulated truth to evaluate the policy with

$$F^\pi = \mathbb{E}_\mu \mathbb{E}_{W^1,\ldots,W^N|\mu} \mu_{x^{\pi,N}}.$$

We are going to later need to represent our unknown function as $F(x, W)$ where W is a random input, from which we can create a noisy observation $\hat{F} = F(x, W)$. We typically do not know the function $F(x, W)$, or we may not know the distribution of W, but either way, we do not know $\mathbb{E}F(x, W)$. If x is discrete, we can still let $\mu_x = \mathbb{E}F(x, W)$. More generally, we can let $\overline{F}^n(x)$ be an approximation of $\mathbb{E}F(x, W)$ after n experiments (where, for example, we have observed $\hat{F}^1, \ldots, \hat{F}^n$).

While there are different ways of approximating $\overline{F}^n(x)$, one that we use later is to assume a parametric form that we represent using $f(x|\theta)$. For example, we might use a linear model

$$f(x|\theta) = \theta_0 + \theta_1\phi_1(x) + \theta_2\phi_2(x),$$

where $(\phi_1(x), \phi_2(x))$ are features derived from x. For example, we may be choosing the best movie, where x specifies the movie, while $\phi_f(x)$ extracts information such as year made, genre, leading actors, and so on.

Using our parametric model, we are going to assume that we can view W as either an input to $F(x, W)$, or as a noisy observation of the parametric function itself as in

$$W = f(x|\theta) + \varepsilon. \tag{6.1}$$

In other words, we can treat a realization of W as a realization of the function.

With a parametric belief model, we assume we know the parametric form, but do not know the parameter vector θ. If we use a linear belief model (as we do in chapter 7), we show how to create a multivariate normal distribution of θ with point estimate $\bar{\theta}^n$, and covariance matrix $\Sigma^{\theta,n}$, where

$$\Sigma^{\theta,n}_{ij} = Cov(\theta_i, \theta_j).$$

In chapter 8 we are going to show how to do optimal learning use a *sampled belief model*, which we first introduced in section 2.4, where the true θ is represented

on nonlinear belief models, we are going to introduce a different way of representing the distribution of belief about the true θ called the sampled belief model.

Using our parametric model, if we use the sampled belief model we introduced in section 2.4 we would obtain a set of probabilities $p^N_k = \mathbb{P}[\theta = \theta_k]$. We could use these probabilities to find an estimate of θ using

$$\bar{\theta}^{\pi,N} = \sum_{k=1}^{K} p^N_k \theta_k.$$

We can then use this estimate to find the best experimental design using

$$x^{\pi,N} = \arg\max_{x\in\mathcal{X}} f(x|\bar{\theta}^{\pi,N}).$$

A better way (if we can compute it) is to compute the best design using

$$x^{\pi,N} = \arg\max_{x\in\mathcal{X}} \sum_{k=1}^{K} f(x|\bar{\theta}_k)p^N_k.$$

We then evaluate our learning policy π by evaluating the performance of the solution using

$$F^{\pi} = \mathbb{E}_{\theta}\mathbb{E}_{W^1,\ldots,W^N|\theta}\mathbb{E}_{\widehat{W}}F(x^{\pi,N},\widehat{W}). \tag{6.2}$$

We can break down the expectation in equation (6.2) as follows. The first expectation $\mathbb{E}_{\theta}$, which would only be used with a Bayesian belief model, is where we take the expectation over different possible values of θ based on some prior (note that W depends on θ as given in equation (6.1)). Then, given θ, the expectation $\mathbb{E}_{W^1,\ldots,W^N|\theta}$ represents the random experiments that we run while learning.

We then have to evaluate how well our design works, and we use $\widehat{W}$ to represent the uncertainty during the testing (while the sequence $W^1,\ldots,W^N$ is the observations used for training). For this reason, we use the final $\mathbb{E}_{\widehat{W}}$ to evaluate our solution $F(x^{\pi,N},\widehat{W})$ by averaging over the random realizations $\widehat{W}$ if we were to implement our final design $x^{\pi,N}$. Of course, we cannot compute these expectations in practice, so we have to turn to the simulation-based estimates that were described in section 3.4.

Online learning (cumulative reward)

In online learning, we are learning as we progress, so we are trying to maximize the cumulative rewards, as we did in section 5.6. Using the same notation as above for offline learning, the value of a policy would be given by

$$F^{\pi} = \mathbb{E}_{\mu}\mathbb{E}_{W^1,\ldots,W^N|\mu}\sum_{n=0}^{N-1}\mu_{X^{\pi}(S^n)}.$$

Now make the transition to using a function $F(x,W)$ to represent our performance that we wish to maximize.

$$F^{\pi} = \mathbb{E}_{\theta}\mathbb{E}_{W^1,\ldots,W^N|\theta}\sum_{n=0}^{N-1}F(X^{\pi}(S^n),W^{n+1}). \tag{6.3}$$

If we refer back to the offline counterpart of equation (6.3), we see that (6.2) has a final $\mathbb{E}_W$ for evaluating the final design $x^{\pi,N}$. In the online version in (6.3), we evaluate as we go.

If we compare our offline, terminal reward objective in (6.2) to our online, cumulative reward objective in (6.3), we see two differences. The most obvious, of course, is that in when evaluating the cumulative reward in (6.2) we have to sum the rewards as we progress rather than just the performance at the end. Second, when using the terminal reward in (6.2) we have to introduce a testing step, which is where we use the random variable $\widehat{W}$. We do not need this in online learning, since the observations $W^1,\ldots,W^N$ used for learning are the same that are used for evaluating experimental choices as we progress.

6.5.2 Experimental costs

There are several variations that distinguish problems in terms of experimental costs. These include

- Startup costs - There may be a certain amount of time or money required to start measuring an alternative, making the first observation much more expensive than subsequent observations. For example, perhaps we have to invite a baseball player to training camp, or we have to purchase a device to test it, or we have to create a molecule in a lab to determine its properties.
- Switchover costs - Startup costs address the first experiment of an alternative, but there may be switchover costs, where even after the first experiment, there may be a cost from making an observation of x to an observation of x'. For example, perhaps we are simulating a policy, but it takes time to switch from testing one set of parameters to another because we have to reload a software package or restart a model that has been stopped.
- Alternative specific costs - We often represent the budget for running experiments as the number of experiments, which also implies that each alternative "costs" the same. But this may not be true at all. For example, an experiment might involve drilling an exploration well (costs depend on geology) or testing a population for disease (cost might reflect distance to travel, or the type of diagnostic test).

6.5.3 Objectives

Below we provide an indication of some objectives we might use. Perhaps not surprisingly, there is a variety of possible objectives that we might use. This discussion simply hints at the diversity of perspectives.

Expected value

Up to now, we have been choosing a policy to maximize our performance. We can write this as

$$V^\pi = \mathbb{E}_\mu \mathbb{E}_{W^1,\ldots,W^N|\mu} \mu_{x^\pi} \tag{6.4}$$

$$= \mathbb{E}_\mu \mathbb{E}_{W^1,\ldots,W^N|\mu} \left\{ \max_{x\in\mathcal{X}} \bar{\mu}_x^{\pi,N} \right\}, \tag{6.5}$$

where $x^\pi = \arg\max \bar{\mu}_x^{\pi,N}$. We remind the reader that the expectation is over all beliefs μ and all experimental outcomes W. We note that in (6.4), we are using our belief μ_x to evaluate the result of our policy x^π. In (6.5), we are using our estimates $\bar{\mu}_x^{\pi,N}$. Both objectives produce the same policy, and in expectation they are the same, as we showed in Section 3.6. In both versions of the objective function, we are taking an expectation over truths, so we are not changing our fundamental model. However, in practice it will be statistically more reliable to use (6.4).

It helps to illustrate the approximation of the expectations in equation (6.5). Assume we have L possible values for the truth μ with possible values $\mu(\ell)$, $\ell = 1,\ldots,L$, where $\mu_x(\ell)$ is the value of alternative x when $\mu = \mu(\ell)$. Next assume that we have

K possible experimental paths, where we represent the kth possible experimental sequence as $W^1(k), \ldots, W^N(k)$, where $k = 1, \ldots, K$.

Next let $\bar{\mu}_x^{\pi,N}(k,\ell)$ be the estimate of μ_x when the truth $\mu = \mu(\ell)$ and we are following sample path $W^1(k), \ldots, W^N(k)$ while using experimental policy π. Averaged over all truths and experimental samples gives us estimates

$$\bar{\mu}_x^{\pi,N} = \frac{1}{L}\sum_{\ell=1}^{L}\frac{1}{K}\sum_{k=1}^{K}\bar{\mu}_x^{\pi,N}(k,\ell). \tag{6.6}$$

Our best design is then given by

$$x^{\pi,N} = \arg\max_x \bar{\mu}_x^{\pi,N}. \tag{6.7}$$

We can then evaluate the value of our policy using

$$\overline{V}^{\pi} = \frac{1}{L}\sum_{\ell=1}^{L}\mu_{x^{\pi,N}}(\ell). \tag{6.8}$$

Note that we are using estimated values $\bar{\mu}_x^{\pi,N}$ when we find the best design in equation (6.7), whereas we use the true value $\mu_x(\ell)$ when evaluate the value of the policy in equation (6.8).

Expected opportunity cost (or regret)

It is very common to minimize the *opportunity cost* (also known as *regret*) which measures how much worse we are doing than the optimal (whether we are minimizing or maximizing). This is written as

$$V^{OC,\pi} = \mathbb{E}_\mu \mathbb{E}_{W^1,\ldots,W^N|\mu}\left\{\max_x \mu_x - \mu_{x^\pi}\right\}. \tag{6.9}$$

This objective function is also known as "linear loss," because it represents the suboptimality of the alternative selected by policy π. The best possible loss value is zero, indicating that we have found the true best alternative.

In online problems, it sometimes makes sense to consider the average opportunity cost or regret per time step. Thus, if we have N experiments total, we can compute

$$\overline{V}^{OC,\pi} = \mathbb{E}_\mu \mathbb{E}_{W^1,\ldots,W^N|\mu}\frac{1}{N}\sum_{n=0}^{N-1}\left\{\max_x \mu_x - \mu_{x^n}\right\}, \tag{6.10}$$

where $x^n = X^\pi(S^n)$.

Risk

A largely overlooked aspect of experimentation is risk. We might be running a series of experiments in the hope that we will achieve a certain target, such as killing

enough cancer cells, finding the price that produces revenue over a particular target, or designing a material whose strength is over a required level. For such situations, we need to replace the expectation with a risk measure that we designate $\rho(Z)$ which operates on a random variable Z (the objective function) in a way that penalizes unacceptable outcomes. For example, imagine we have a threshold z_α. Our risk measure might be

$$\rho(Z) = \mathbb{E}Z + \eta\mathbb{E}\max\{0, Z - z_\alpha\}. \tag{6.11}$$

This version of $\rho(Z)$ is striking a balance between maximizing the expectation $\mathbb{E}Z$ and maximizing the expectation of Z when it exceeds a threshold z_α. Here, the parameter η puts the weight on outcomes $Z > z_\alpha$, with an implicit penalty on outcomes $Z < z_\alpha$. Evaluating policies using the risk measure $\rho(Z)$ using simulation is the same as with an expectation, with the single change of including the additional penalty term. This said, very little attention has been placed on handling risk in learning problems.

Robust optimization

In some settings, it makes more sense to focus on the worst case rather than an expectation. This is becoming known in the optimization community as *robust optimization*. We illustrate the calculation of a robust objective using the sampled approximation we introduced when computing the expected value of a policy above. There, we let $\mu_x(\ell)$ for $\ell = 1, \ldots, L$ be a set of truths for μ_x, and $W^n(k)$ be the nth realization of W when following sample path k, $k = 1, \ldots, K$. Let $\bar{\mu}^{\pi,N}$ calculated using equation (6.6), and let $x^{\pi,N}$ be the best design computed using equation (6.7).

Above, we computed the expected performance of a policy using

$$\overline{V}^{\pi} = \frac{1}{L}\sum_{\ell=1}^{L} \mu_{x^{\pi,N}}(\ell).$$

We might be interested in knowing how badly our policy might perform when the truth is some extreme value. Instead of taking an average, we might instead use the truth that produces the worst performance, which we can compute using

$$\overline{V}^{robust,\pi} = \min_{\ell} \mu_{x^{\pi,N}}(\ell). \tag{6.12}$$

where $x^{\pi,N}$ is computed using equation (6.7).

Equation (6.12) recognizes that some truths may be harder to discover than others. We may also want to consider that some experimental sequences may also produce worse designs. Let

$\bar{\mu}_x^{\pi,N}(k,\ell)$ = Estimate of μ_x when $\mu = \mu(\ell)$ and we observe experimental sequence $W^1(k), \ldots, W^N(k)$.

Now let the resulting best design be computed using

$$x^{\pi,N}(k) = \arg\max_x \bar{\mu}_x^{\pi,N}(k,\ell). \tag{6.13}$$

Finally, we can compute the worst possible performance over truths and sample paths using

$$\overline{V}^{robust2,\pi} = \min_{\ell} \min_{k} \mu_{x^{\pi,N}(k)}(\ell). \tag{6.14}$$

We can now use $\overline{V}^{robust,\pi}$ or $\overline{V}^{robust2,\pi}$ as the value of the policy when comparing the performance of policies. While it is nice to find policies that work best on average, there may be considerable interest in policies that mitigate the worst possible outcomes.

Yet a further generalization is to use the concept of *quantile optimization*. If Q_α is the α quantile of a random variable, we may wish to solve

$$V_\alpha^\pi = Q_\alpha \mu_{x^\pi}. \tag{6.15}$$

Quantile optimization is related to robust optimization, because we might choose to maximize the 10^{th} quantile. Quantile optimization is also useful in the presence of heavy-tailed random variables.

Probability of correct selection

A different perspective is to focus on the probability that we have selected the best out of a set $\mathcal{X}$ alternatives. In this setting, it is typically the case that the number of alternatives is not too large, say 10 or 20, and certainly not 100,000. Assume that

$$x^* = \arg\max_{x \in \mathcal{X}} \mu_x$$

is the best decision (for simplicity, we are going to ignore the presence of ties). As before our we choose as the best design

$$x^{\pi,N} = \arg\max_{x \in \mathcal{X}} \bar{\mu}_x^{\pi,N}.$$

We have made the correct selection if $x^{\pi,N} = x^*$, but even the best policy cannot guarantee that we will make the best selection every time. Let $1_{\{\mathcal{E}\}} = 1$ if the event $\mathcal{E}$ is true, 0 otherwise. We write the probability of correct selection as

$$\begin{aligned} P^{CS,\pi} &= \text{probability we choose the best alternative} \\ &= \mathbb{E}^\pi 1_{\{x^{\pi,N}=x^*\}}, \end{aligned}$$

where the underlying probability distribution depends on our learning policy π. The probability is computed using the appropriate distribution, depending on whether we are using Bayesian or frequentist perspectives. This may be written in the language of loss functions. We would define the loss function as

$$L^{CS,\pi} = 1_{\{x^{\pi,N} \neq x^*\}}.$$

Although we use $L^{CS,\pi}$ to be consistent with our other notation, this is more commonly represented as L_{0-1} for "0-1 loss."

Note that we write this in terms of the negative outcome so that we wish to minimize the loss, which means that we have not found the best selection. In this case, we would write the probability of correct selection as

$$P^{CS,\pi} = 1 - \mathbb{E}^\pi L^{CS,\pi}.$$

Indifference zone selection

A variant of the goal of choosing the best is to maximize the likelihood that we make a choice that is almost as good as the best. Assume we are equally happy with any outcome within δ of the best. This is referred to as the *indifference zone*. Let $V^{n,\pi}$ be the value of our solution after n experiments. We require $\mathbb{P}^\pi\{\mu_{d^*} = \bar{\mu}^*|\mu\} > 1-\alpha$ for all μ where $\mu_{[1]} - \mu_{[2]} > \delta$, and where $\mu_{[1]}$ and $\mu_{[2]}$ represent, respectively, the best and second best choices.

We might like to maximize the likelihood that we fall within the indifference zone, which we can express using

$$P^{IZ,\pi} = \mathbb{P}^\pi(V^{n,\pi} > \bar{\mu}^* - \delta).$$

As before, the probability has to be computed with the appropriate Bayesian or frequentist distribution.

Least squared error

A different form of loss function arises when we want to fit a function to a set of data. In this setting, we think of "x" as a set of independent variables which we choose directly or indirectly. For example, we may be able to choose x directly when fitting a linear regression of the form

$$\begin{aligned} Y(x) &= \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_I x_I + \epsilon \\ &= \theta x + \epsilon. \end{aligned}$$

where Y is the observed response and ϵ is the random error explaining differences between the linear model and the responses. We choose it indirectly when our regression is in the form of basis functions, as in

$$Y(x) = \sum_{f\in\mathcal{F}} \theta_f \phi_f(x) + \epsilon.$$

Classical linear regression assumes that we are given a set of observations which we use to fit a model by choosing θ. Let

$$Y^{n+1} = \theta x^n + \epsilon^{n+1}.$$

where θ is the true set of parameters. Our indexing reflects our requirement that x^n be chosen before we observe ϵ^{n+1}. Our measure of performance is given by

$$F(Y^{(N+1)}, x^{(N)}|\bar{\theta}) = \sum_{n=1}^{N} (Y^{n+1} - \bar{\theta}x^n)^2.$$

which is the sample sum of squares given experiments $x^{(N)} = (x^0, \ldots, x^N)$ and observations $Y^{(N+1)} = (Y^1, \ldots, Y^{N+1})$. Ordinary least squares regression fits a model by finding

$$\bar{\theta}^{N+1} = \arg\min_{\bar{\theta}} F(Y^{(N+1)}, x^{(N)}|\bar{\theta}).$$

Let $F^*(Y^{(N+1)}, x^{(N)}) = F(Y^{(N+1)}, x^{(N)}|\bar{\theta}^{N+1})$ be the optimal solution given $Y^{(N+1)}$ and $x^{(N)}$. Sequential estimation starts with $\bar{\theta}^n$, then measures x^n, and finally observes Y^{n+1} from which we compute $\bar{\theta}^{n+1}$. This can be done easily using recursive least mean squares, given by

$$\bar{\theta}^{n+1} = \bar{\theta}^n - H^n x^n (\bar{\theta}^n x^n - Y^{n+1})$$

where H^n is an $I \times I$ scaling matrix that is computed recursively (we cover this in Section 7.2.2).

Our focus is on choosing the experiment x^n. Classical experimental design assumes that we choose $(x^n)_{n=0}^N$ first and then fit the model. This is sometimes referred to as batch design because the entire sample is chosen first. This is equivalent to solving

$$\min_{x^{(N)}} \mathbb{E} F^*(Y^{(N+1)}, x^{(N)})$$

where the expectation is over the random variables in $Y^{(N+1)}$.

We focus on sequential design, where we choose x^n given our state S^n, which includes $\bar{\theta}^n$ and the information we need to update $\bar{\theta}^n$. In a sequential learning problem, we have to use some basis for determining how well we have done. In our optimization problems, we want to maximize our expected contribution. This optimization problem determines the values of x that are most interesting. In the area of adaptive estimation, we have to specify the values of x that are most likely to be interesting to us, which we designate by a density $h(x)$ which has to be specified. In an off-line learning environment, we want to choose $x^1, \ldots, x^n, \ldots, x^N$ according to a policy π to solve

$$\min_{\pi} \mathbb{E} \int_x \left(Y(x) - \bar{\theta}^{\pi} x\right)^2 h(x) dx$$

where $Y(x)$ is the random variable we observe given x, and where $\bar{\theta}^{\pi}$ is the value of $\bar{\theta}$ produced when we select x^n according to π, and when we estimate $\bar{\theta}$ optimally.

This formulation requires that we specify the domain that interests us most through the density $h(x)$. An illustration of the density function arises when we are trying to sample nuclear material over a border or in a region. For such cases, $h(x)$ might be the uniform density over the region in question. When we solve on-line and off-line optimization problems, we do not have to specify $h(x)$ explicitly. The optimization problem, e.g. equation (5.3), determines the region within $\mathcal{X}$ that is of greatest interest.

Entropy minimization

Entropy is a measure of uncertainty that can be used for numeric and nonnumeric data. Imagine that we are trying to estimate a parameter μ that we know with uncertainty. If our distribution of belief about μ is continuous with density $f(u)$, a measure of the uncertainty with which we know μ is given by the entropy of $f(u)$, given by

$$H(\mu) = -\int_u f_u \log(f_u) du.$$

The logarithm is typically taken with base 2, but for our purposes, the natural log is fine. The entropy is largest when the density is closest to the uniform distribution. If the entropy is zero, then we know μ perfectly. Thus, we can try to take experiments that reduce the entropy of the distribution that describes our knowledge about a parameter.

6.6 DESIGNING POLICIES

Up to now, we have introduced a number of different policies to illustrate different methods for learning. Our list at this stage includes

- Decision trees (section 1.6) - Decision trees allow us to optimize decisions and information over some horizon.
- Dynamic programming (section 3.2.2) - We showed that we could set up a learning problem as a dynamic program using Bellman's optimality equation to characterize an optimal policy (which is typically impossible to solve).
- Pure exploration (section 3.2.3) - Here we just choose actions at random.
- Excitation policies (section 3.2.3) - For problems with continuous controls, we add a noise term as in

  $$X^{\pi}(S_t|\theta) = \theta_0 + \theta_1 S_t + \theta_2 S_t^2 + \varepsilon.$$

  The noise ε serves as an *excitation* term to encourage exploration.
- Epsilon-greedy (section 3.2.3) - Randomly explore (choose an action at random) or exploit (choose the action that appears to be best).
- Boltzmann exploration (section 3.2.3) - Choose an action x with a probability proportional to the exponential of an estimated value of the action.
- Interval estimation (section 3.2.3) - Choose the action with the highest index which is given by the α-percentile of the distribution of belief about the value of the actcion.
- Upper confidence bounding (sections 3.2.3 and 5.2) - Choose the action with the highest index given by expected reward plus a term that captures how often an action has been tested (there are a number of variations of this).
- Gittins indices (section 5.3) - Choose an action based on the Gittins index, which scales the standard deviation of the measurement noise. This produces an optimal policy for certain problems.
- Value of information policies (chapter 4) - This includes the knowledge gradient and expected improvement.

At this point we are not going to make a distinction between policies for offline learning (where we maximize the final reward) or online learning (where we maximize the cumulative reward). Rather, we are going to take the position that we have a family

of policies to consider which have to be evaluated using whatever objective function we choose.

It turns out that this rather daunting list of policies can be organized into two core strategies, each of which can then be further divided into two classes, creating four (meta) classes of policies.

The two core strategies are:

Policy search These are policies that are characterized by a set of parameters θ that have to be tuned either using a simulator (offline) or in the field (online).

Lookahead policies This group tries to build good policies by approximating the impact of a decision now on the future.

6.6.1 Policy search

Policy search involves searching over a parameterized set of policies to find the one that works the best, typically in a simulated setting but possibly in an online, real-world setting. These come in two flavors:

Policy function approximations (PFAs) These are analytical functions that map states directly to actions. Some examples are

- We have a policy to apply a force x based on the speed of a robot and the slope of the surface it is moving over. If S_{t1} is the speed and S_{t2} is the slope, we might choose are speed using the policy

$$X^\pi(S_t|\theta) = \theta_0 + \theta_1 S_{t1} + \theta_2 S_{t1}^2 + \theta_3 S_{t2} + \theta_4 S_{t2}^2.$$

- We wish to use a battery to draw energy from the grid when prices are low, and then sell back to the grid when they are high. We can write the policy as

$$X^\pi(S_t|\theta) = \begin{cases} -1 & \text{If } p_t \leq \theta^{min} \\ 0 & \text{If } \theta^{min} < p_t < \theta^{max} \\ 1 & \text{If } p_t \geq \theta^{max} \end{cases}$$

  where $\theta = (\theta^{min}, \theta^{max})$ are tunable parameters.

Parametric cost function approximations (CFAs) In this class, we are maximizing or minimizing a parametric function, which is typically a parametrically modified cost function. The simplest example of this is interval estimation, where the policy is

$$X^{IE}(S^n|\theta) = \arg\max_x \left(C(S^n, x) + \theta\bar{\sigma}_x^n\right).$$

The distinguishing feature of a CFA policy is the presence of an $\arg\max_x$ or $\arg\min_x$ of a cost that does not model decisions and information in the future.

6.6.2 Lookahead policies

Lookahead policies are based on finding the best decision that maximizes the expected rewards over some horizon. We can divide these into two broad groups:

Value function approximations (VFAs) - In this class we design functions that approximate the value of being in a state resulting from an action made now.

Direct lookaheads (DLAs) - This approach explicitly optimizes decisions now and into the future over some horizon, to help make a better decision now. For learning problems, it is useful to further divide this class into two subclasses:

- Single-period lookahead - We make a decision at iteration n that considers the decision we will make at iteration $n+1$.
- Multi-period lookahead - We optimize decisions for H time periods into the future.

We first saw a policy using a value function in section 3.2.2 where we showed we could write

$$V^n(S^n) = \max_x \big(C(S,x) + \mathbb{E}_W V^{n+1}(S^{n+1})\big). \tag{6.16}$$

where $S^{n+1} = S^M(S^n, x, W^{n+1})$, and where S^n is our belief state after n experiments. There are settings where our belief state is discrete which allows us to actually compute $V^n(S^n)$, but this is rare. Typically S^n is multidimensional and, typically, continuous, making it impossible to compute $V^n(S^n)$ exactly. An alternative approach is to try to approximate the value function, but as of this writing, this has not emerged as a useful approach for pure learning problems.

By contrast, policies based on one-step lookahead models (and in some cases multi-step) have actually proven to be quite popular.

6.6.3 Classifying policies

Every single policy can be classified as one of our four classes: PFAs, CFAs, VFAs, and DLAs. Below we list all the policies we have seen so far using this grouping:

PFAs Policy function approximations include

- Pure exploration, where we choose an action at random (unrelated to its value).
- Lookup tables which specify "when in discrete state s take discrete action x." This includes any rule-based system.
- Linear models, often called "affine policies" with the form:

$$X^\pi(S^n|\theta) = \sum_{f\in\mathcal{F}} \theta_f \phi_f(S^n).$$

- Excitation policies, where we add a noise term to a continuous controller (such as the affine policy above)

- Neural networks. These are popular in the control of engineering systems, but have not been used for learning problems.

CFAs Parametric cost function approximations include

- Pure exploitation, where we choose

$$X^\pi(S^n) = \arg\max_x \bar{\mu}_x^n.$$

- Epsilon-greedy, where we choose an action at random with probability ϵ and use a pure exploitation policy with probability $1-\epsilon$.
- Bayes greedy. Imagine we have a nonlinear function $f(x|\theta)$ where θ is an unknown parameter that might be multivariate normal $\mathcal{N}(\theta^n, \Sigma^n)$. A basic pure exploitation policy would be

$$X^\pi(S^n) = \arg\max_x f(x|\mathbb{E}^n\theta) = \arg\max_x f(x|\theta^n).$$

  This is a basic greedy (or myopic) policy, but what we really want to do is to maximize the expectation of the function (rather than the function evaluated at the expectation). This would be written

$$X^\pi(S^n) = \arg\max_x \mathbb{E}\{f(x|\theta)|S^n\}$$

  where S^n is the state that the true $\theta \sim \mathcal{N}(\theta^n, \Sigma^n)$. Using the function of the expectation is fine if $f(x|\theta)$ is linear in θ, but not if it is nonlinear. If we maximize the expectation of the function, then this is called a Bayes greedy policy.
- Interval estimation:

$$X^{IE}(S^n|\theta^{IE}) = \arg\max_x \left(\bar{\mu}_x^n + \theta^{IE}\sigma_x^n\right).$$

- Upper confidence bounding, for example

$$X_x^{UCB1,n}(S^n|\theta^{UCB1}) = \arg\max_x \left(\bar{\mu}_x^n + \theta^{UCB1}\sqrt{\frac{2\log n}{N_x^n}}\right).$$

- Boltzmann exploration, where we choose an action x with probability

$$p_x^n(\theta) = \frac{e^{\theta\bar{\mu}_x^n}}{\sum_{x'\in\mathcal{X}} e^{\theta\bar{\mu}_{x'}^n}}.$$

  The Boltzmann policy is often referred to as a "soft-max" policy does not have an explicit $\arg\max_x$, but is performing this operation probabilistically.

VFAs This would include our optimal policy (equation (6.16)), where the policy depends on a value function, as in

$$X^{VFA}(S^n) = \arg\max_x \left(\bar{\mu}_x^n + \mathbb{E}_W V^{n+1}(S^{n+1})\right).$$

This is how Gittins indices were computed in section 5.3.

DFAs Direct lookaheads - These come in single-step and multi-step variations:

- Single step - These are very popular in optimal learning applications, and include:
  - Value of information policies such as the knowledge gradient or expected improvement.

$$X_x^{KG,n} = \arg\max_x \left(\mathbb{E}\left\{ \max_{x' \in \mathcal{X}} \bar{\mu}_{x'}^{n+1}(x) | S^n \right\} - V^n(S^n) \right).$$

    The maximization over x' is actually an optimization over possible decisions x^{n+1}, after the outcome of the experiment $x^n = x$ has been completed.
  - Thompson sampling:

$$X^{TS}(S^n) = \arg\max_x \hat{\mu}_x^{n+1}.$$

    Thompson sampling can be thought of a simulation of what might happen in the $n+1st$ experiment based on the time n distribution of belief.
- Multistep lookaheads:
  - Decision trees, which we first introduced in section 1.6.
  - The KG(*) policy, where we make a decision based on repeating an experiment n_x times (this is a restricted lookahead).
  - Extensions of KG(*) based on tuning the lookahead. We introduced this in seciton 4.3.3, where the policy is given by

$$X^{KG}(\theta^{KG}) = \arg\max_x \nu_x^{KG,n}(\theta^{KG})$$

    Here, θ^{KG} is the parameter governing how many times we plan on repeating each experiment.

As of this writing, CFAs (such as upper confidence bounding) and one-step DLAs are by far the most popular. Randomized exploitation policies (a form of PFA) are popular with online applications, where it is tempting to do what appears to be best, but with some recognition that we need to introduce exploration.

Almost all of these policies can be tuned, or converted to a tunable version. Value of information policies are generally used without a tunable parameter by exploiting a Bayesian prior.

6.7 THE ROLE OF PRIORS IN POLICY EVALUATION

The process of evaluating policies is subtle, because it can bias the identification of good policies toward problems with specific properties that may not be clearly identified. Perhaps one of the most difficult issues arises in the interaction between how problems are generated (that is, the "truth") and the relationship between the truth and any prior information that may be used.

Truth from prior

From the Bayesian point of view, arguably the most natural way to evaluate a learning policy is to use the method described in Chapter 3, where we:

1) Generate a truth $\mu(\psi)$ from a prior.
2) Sample observations $W(\omega)$ from the truth $\mu(\psi)$.
3) Evaluate how well we discover the truth using one of the objectives listed above.
4) Repeat many times for different truths and different sample observations, to evaluate how well we discover the truth on average, and how reliably.

This strategy represents a very controlled experiment, which eliminates potential biases which a particular policy might be able to exploit. However, this approach may bias the evaluation process toward policies that work well with good priors.

Truth from an alternative prior

A way of evaluating the robustness of a policy is to generate truths from distributions other than the prior. For example, our prior may be normally distributed, but we might generate truths from a uniform or exponential distribution. Such experiments should focus on the ability of a policy to work with truths that come from distributions with potentially heavy tails, but the expected truth should match the expectation of the priors, which is to say that the prior should be unbiased.

Truth from a fixed prior

In the global optimization literature, where we collect noisy observations in an attempt to discover the maximum of an unknown continuous function, it is common to evaluate policies using a few "test functions." In other words, the truth is set to a fixed value that, for some reason, is believed to be a particularly interesting or difficult test case. For example, the Branin function

$$f(x_1, x_2) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos(x_1) + 10$$

is a commonly used test function in two dimensions. Test functions may be highly non-linear with multiple local maxima, and so we may view them as posing an especial challenge for an optimization algorithm. If we are able to successfully find the point (x_1^*, x_2^*) that maximizes this difficult function, we may reasonably believe that our policy will be able to learn other functions.

This is a frequentist approach to policy evaluation, where we consider a few fixed test cases rather than sampling truths from a distribution. However, the strength of this approach is that it presents a very clear and well-defined test suite of problems. In Chapter 14, we mention some of these test functions, such as the Branin function, the Ackley function, and others.

Truth from a biased prior

In any particular learning situation, the sample realization of a specific truth is going to be higher or lower than the mean of any prior distribution. However, a subtle issue arises if the expected truth is higher or lower than the expectation of the prior, an issue that we touched on in Section 4.9. In that section, we discussed the problem with the prior is consistently higher or lower than the truth. But this discussion did not recognize that it may be possible to identify and correct biases. Even more important to the discussion here is the fact that if there is a bias, then certain types of policies will tend to exploit this bias. For example, we might use epsilon-greedy or interval estimation, each of which has a tunable parameter. In the presence of a bias, we only have to tune the parameters to emphasize more exploration, and we are likely to get a better policy.

Expectations versus risks

Most of our presentation focuses on the expected performance of a learning policy. The probability of correct selection and indifference zone criterion can be viewed as risk-based measures, since they focus on finding alternatives that are within a specified tolerance with a specified probability.

For many information collection problems, risk is a major issue. Finding a workable drug that will help to extend lives with a high probability may be much more attractive than finding the best drug, but with a significant risk of not finding anything. The workable drug may not be ideal, but it may be good enough to gain federal approval which can then be marketed.

In real implementations, we are not allowed to run many sample realizations and take an average. We will use a policy on a single sample path. If we are not successful in finding, for example, a workable drug, we may be left asking whether this was just bad luck, or did we have a bad policy. We would like to minimize the possibility that it was bad luck.

Regret bounds

In Section 5.2, we described a policy known as upper confidence bounding. For example, for the case of normally distributed priors and rewards, the UCB policy was given by

$$\nu_x^{UCB1-Normal,n} = \bar{\mu}_x^{\pi,N} + 4\sigma_W\sqrt{\frac{\log n}{N_x^n}}.$$

where N_x^n is the number of times we have observed alternative x after n trials. UCB policies have attracted considerable attention in the community that works on multi-armed bandit problems after it was found that it was possible to bound the number of times that incorrect arms would be tested by $C \log N$, for an appropriately chosen constant C. In fact, it was shown that this was the best possible bound, which is often interpreted to mean that this may be the best possible policy.

In Section 5.4, comparisons were made between a tuned UCB policy and the knowledge gradient policy for problems with normally distributed rewards. Although the knowledge gradient policy outperformed UCB, it is hard to know if this reflects a subtle bias in how the experiments were run. As of this writing, there is insufficient empirical evidence to form any judgments regarding the value of regret bounds in terms of predicting the empirical performance of a UCB policy.

6.8 POLICY SEARCH AND THE MOLTE TESTING ENVIRONMENT

We first addressed the problem of evaluating policies in some depth in section 3.3 in the context of offline (terminal reward) settings. When we are using a simulator, we can use a policy π to find estimates $\bar{\mu}_x^{\pi,N}$ and then find the best choice using

$$x^{\pi,N} = \arg\max_x \bar{\mu}_x^N,$$

We then evaluate our policy using the known truth (which we are simulating), which we write as

$$F^{\pi,N} = \mathbb{E}_\mu \mathbb{E}_{W^1,\ldots,W^N|\mu}\, \mu_{x^{\pi,N}}.$$

We often use the opportunity cost, or regret, where we compare how well we do against the best that could be done (which is possible in our simulated environment). This would be written as

$$R^{\pi,N} = \mathbb{E}_\mu \mathbb{E}_{W^1,\ldots,W^N|\mu}\big(\max_x \mu - \mu_{x^{\pi,N}}\big).$$

Then, in chapter 5, we noted that we could evaluate policies using an online (cumulative reward) which we can write as

$$F^{\pi,N} = \mathbb{E}_\mu \mathbb{E}_{W^1,\ldots,W^N|\mu} \sum_{n=0}^{N-1} \mu_{X^\pi(S^n)}.$$

or in terms of its opportunity cost (or regret)

$$F^{\pi,N} = \mathbb{E}_\mu \mathbb{E}_{W^1,\ldots,W^N|\mu} \sum_{n=0}^{N-1} \big(\max_x \mu_x - \mu_{X^\pi(S^n)}\big).$$

Policy search typically involves searching across different classes of policies (UCB, interval estimation, Gittins indices, knowledge gradient), and then tuning parameters within a class (if there are tunable parameters).

We developed a public domain, Matlab-based system called MOLTE (Modular, Optimal Learning Testing Environment) for comparing a variety of policies on a library of test problems. Comparisons can be made on either an offline (terminal reward) or online (cumulative reward) basis. Each policy, and each problem, is encoded in its own .m file.

The front end of MOLTE is a spreadsheet, shown in figure 6.1, where you can specify which problems you want to use for testing, and which policies you want to apply

Problem class	Prior	Measurement Budget	Belief Model	Offline/ Online	Number of Policies				
PayloadD	MLE	0.2	independent	Offline	4	kriging	EXPL	IE(1.7)	Thompson Sampling
Branin	MLE	10	correlated	Online	4	OLkgcb	UCBEcb(*)	IE(2)	BayesUCB
Bubeck4	uninformative	5	independent	Online	4	OLKG	UCB	SR	UCBV
GPR	Default	0.3	correlated	Offline	4	kriging	kgcb	IE(*)	EXPT

Figure 6.1 Snapshot of the spreadsheet front-end of MOLTE. Each line lists a test problem, along with choices on how to construct a prior (maximum likelihood estimation, pre-specified, uninformative), the measurement budget (as a fraction of the number of alternatives), belief model (independent or correlated), and whether it is offline vs. online evaluation. Then there is a list of policies, where the first policy is used as the reference policy against which the other policies are compared. Tunable parameters can be specified, or a "*" indicates the package should do the tuning.

to that problem. Each line of the spreadsheet refers to a different test problem in the library, which is captured by a .m file with the name of the problem (such as "Branin.m"). The user then has to specify

- Prior - This is how to construct a prior. Some choices are
  - MLE - Maximum likelihood estimation -
  - Default - Here we assume that that the prior is encoded within the specification of the problem.
  - Uninformative - An uninformative prior means that we have no information at all about the function (think of infinite variance in the prior).
- Measurement budget - this is specified as a fraction of the number of alternatives.
- Belief model - Independent or correlated beliefs (the system can only handle lookup table belief models).
- Offline vs. online - Offline means we evaluate a policy based on the terminal reward (after the experimental budget has been exhausted). Online means evaluating a policy based on the total rewards earned during the evaluation (the classic bandit setting).

A partial list of the test problems is given in table 6.1.

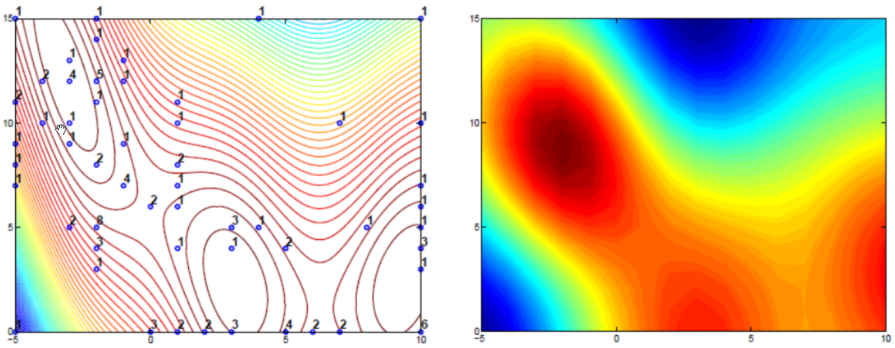
Then, for each problem, the user may list any number of pre-coded learning policies, shown in table 6.2. Many (but not all) of these policies have a tunable parameter. When these exist, the user may specify (for interval estimation) "IE(2.2)" if the parameter should be set to 2.2, or "IE(*)" to request that MOLTE do automated tuning. When automated tuning is requested, MOLTE first searches over the range 10^{-5} to 10^{5}, finds the right order of magnitude, and then refines the search.

MOLTE outputs a number of tabular and graphical reports to summarize the performance of the different policies. Figure 6.2 shows a graphical picture of the sampling pattern for a two-dimensional problem for some policy; these plots provide insights into the nature of the different behaviors. Figure 6.3 provides a frequency distribution of the difference between each policy and the reference policy.

MOLTE also produces tabular performance statistics, including both the opportunity cost relative to the reference policy, and the probability a policy outperforms the

Bubeck1 Bubeck7
Asymmetric Unimodular Functions
Rosenbrock function
Pinter's function
Goldstein function
Griewank function
Braninâ^s function
Axis parallel hyper-ellipsoid function
Rastriginâ^s function
Ackleyâ^s function
Six-hump camel back function
Easom function
Gaussian process regression functions
Payload delivery problem
Nanoparticle design

Table 6.1 A list of some of the test problems in MOLTE.



(a) Sampling distribution (b) Function heatmap

Figure 6.2 Example of the sampling distribution from MOLTE, and the resulting function approximation shown as a heatmap.

reference policy over many simulations. An example is shown in figure 6.3. MOLTE outputs the Latex code for the body of the future.

MOLTE makes it easy to add new problems and new policies by just following the style of other problems and policies. The software and a users manual is available at

Interval Estimation (IE)
Kriging
UCB (independent beliefs)
UCBcb (correlated beliefs)
UCBNormal
UCB-E (independent beliefs)
UCBEcb (correlated beliefs)
UCB-V (independent beliefs)
UCBVcb (correlated beliefs)
Bayes-UCB
KL-UCB
Knowledge gradient policy for offline learning
Knowledge gradient for online learning
Successive rejects
Thompson sampling
Pure exploration
Pure exploitation

Table 6.2 A list of policies that are available in MOLTE.

`http://castlelab.princeton.edu/software/`.

6.9 DISCUSSION

The purpose of this chapter was to provide a more comprehensive perspective of learning problems by covering all the major dimensions of any sequential learning problem. The four classes of policies provides an equally comprehensive roadmap for designing policies that cuts across the different communities that have contributed to solving this important problem class.

One of the challenges of optimal learning is that important contributions have been made in specific subcommunities with a style that reflects both the characteristics of specific problem classes as well as the culture of different research communities. For example, the bandit community, which has attracted the attention of computer science, tends to emphasize online problems with relatively small action spaces, no switchover costs, a Bayesian belief model and a desire to derive provable regret bounds. The simulation optimization community, which tends to focus on finding the best of a small set of designs using discrete event simulation, also deals with

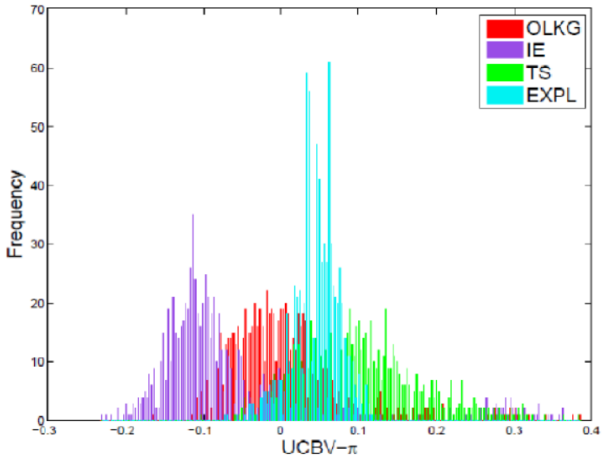


Figure 6.3 Frequency distribution of difference in performance between each policy and the reference policy.

Problem Class	IE		UCBE		UCBV		Kriging		TS		EXPL	
	OC	Prob.	OC	Prob.	OC	Prob.	OC	Prob.	OC	Prob.	OC	Prob.
Goldstein	-0.061	0.81	-0.097	0.92	-0.003	0.45	-0.031	0.73	0.100	0.09	0.041	0.16
AUF_HNoise	0.058	0.40	0.022	0.43	0.037	0.54	0.031	0.39	0.073	0.22	0.047	0.48
AUF_MNoise	0.043	0.29	0.027	0.42	0.343	0.21	0.023	0.28	0.173	0.21	-0.057	0.52
AUF_LNoise	-0.043	0.73	-0.013	0.64	0.053	0.51	0.005	0.53	0.038	0.20	0.003	0.62
Branin	-0.027	0.76	0.025	0.24	0.026	0.26	0.004	0.54	0.041	0.07	0.123	0.00
Ackley	0.007	0.42	0.04	0.41	0.106	0.20	0.037	0.42	0.100	0.23	0.344	0.00
HyperEllipsoid	-0.059	0.73	0.064	0.12	0.08	0.07	0.146	0.22	0.011	0.38	0.243	0.03
Pinter	-0.028	0.56	-0.003	0.51	0.029	0.42	-0.055	0.65	0.122	0.19	0.177	0.04
Rastrigin	-0.082	0.70	-0.03	0.56	0.162	0.04	-0.026	0.57	0.136	0.08	0.203	0.01

Table 6.3 Example of table produced by MOLTE. MOLTE outputs the latex code for the body of the table; headings have to be provided. OC refers to opportunity cost, and Prob. is the probability the policy outperforms the reference policy specified in the spreadsheet.

small choice sets but uses offline learning of a model with significant switching costs, a frequentist belief model, and a desire to show asymptotic optimality.

6.10 BIBLIOGRAPHIC NOTES

Sections 6.1-6.5 - This representation of learning problems is new, but is based on the modeling framework for dynamic programs presented in Powell (2011). The notion of the "hyperstate" to solve the optimal learning problem was put forth by Bellman & Kalaba (1959); further efforts in this direction were undertaken by Cozzolino et al. (1965), Martin (1967) and Satia & Lave (1973).

Section 6.5 - The different objective functions have been proposed by different authors in different communities. Expected opportunity cost lends itself better to the Bayesian

approach (Chick & Inoue 2001), but can also be analyzed in a frequentist setting (Chick 2003, Chick & Wu 2005). A review of early research on indifference zone procedures is given in Bechhofer et al. (1968), with a review of somewhat more recent papers given by Bechhofer et al. (1995).

Section 6.7 - The truth-from-prior approach is used in Bayesian problems; see e.g. Ryzhov et al. (2011). Vermorel & Mohri (2005) and Chhabra & Das (2011) also make use of other evaluation strategies such sampling from an alternative prior or from an empirical dataset. Lai & Robbins (1985) provides the original paper on regret bounds for upper confidence bound policies. Regret bounds continue to be popular, usually in connection with upper confidence bound methods; see, for example, Agrawal (1995), Auer et al. (2002), Auer et al. (2008), Bartlett et al. (2008), Kleinberg et al. (2010), or Srinivas et al. (2010).

PROBLEMS

In each of the exercises below, you are given a situation that involves learning. For each situation, you need to describe the five fundamental dimensions of a learning model, including:

- Carefully define the state variable, giving variable names to each component. Clearly distinguish the belief state (what you think about unknown parameters whose beliefs are evolving over time) and any physical state variables.
- Define the experimental decision (how you are collecting information) and the implementation decision (what you are doing with the information). Note that these may be the same.
- Define the exogenous information. What are you observing? What is the source of the information? Is the information possibly changing your belief about a parameter?
- Describe the equations that make up the transition function. Distinguish between the equations used to update your belief state from those that update any physical or informational state variables that may be present.
- Define your objective function. Here is where you are going to distinguish between offline and online learning. Be sure to differentiate costs of measuring from implementation costs.

Note that in our modeling, we are ignoring the dimension of designing policies. This is addressed in all the remaining chapters of this volume. You may feel that you need information not specified in the problem description, so you should just highlight any missing elements, or make up elements to round out your model.

6.1 An entrepreneur would like to market the use of portable solar panels for recharging cell phones in Africa. The idea is to purchase a number of these solar panels and then to let individuals try to start businesses in different regions of Africa. Each region will face its own unique characteristics in terms of need, competition for alternative sources of energy and the ability of the local population to pay. For example, the market responds

badly to decisions to raise prices. Focus on the problem faced by a single individual who has to figure out a pricing strategy as quickly as possible.

6.2 A pharmaceutical company is faced with the problem of performing a series of market research studies to determine the best pricing for a drug. The market can be expected to respond to recent pricing behavior, and the performance of the drug. The company would like to strike a balance between maximizing revenue and minimizing costs related to the market research. Market research studies may be run in local regions while the drug is still being marketed nationally.

6.3 The Centers for Disease Control wants to collect information so that it can best understand the scope of an outbreak of a virulent new virus in the northeast of the United States. On a limited budget, the CDC would like to manage a single team of technicians who will set up a tent to test people as they walk by. The tent will typically be set up for 1-3 days before the team moves to another location.

6.4 An analyst is using an expensive computer simulation to model the dynamics of wind and its effect on generating electricity. It is important to understand the impact of very low periods of wind, but obtaining these observations can require simulating years, which can take weeks of time on the computer. The analyst is using the simulator to design the location of wind farms and investments in the power grid. She might run shorter simulations to do quick evaluations to eliminate poor designs, but much longer simulations are needed to obtain accurate estimates of the likelihood that a particular design will be susceptible to blackouts.

CHAPTER 7

LINEAR BELIEF MODELS

In the ranking and selection problem and the multiarmed bandit problem, we assumed we had a finite set of alternatives $x \in \mathcal{X} = \{1, 2, \ldots, M\}$, with a belief μ_x about each of the finite alternatives. Known as a lookup table representation, this model is very flexible in that it does not require any assumed structure among the alternatives. However, such models become very clumsy when the number of alternatives is large.

In this chapter, we make the transition from a lookup table belief model to one that uses a parametric model that is linear in the parameters. We assume that an experiment x can be written as the vector $x = (x_1, x_2, \ldots, x_K)$ where x_k may be discrete or continuous. Given x, we observe a value y where we assume a linear relationship of the form

$$f(x|\theta) = \sum_{f \in \mathcal{F}} \theta_f \phi_f(x),$$

where $\mathcal{F}$ is a set of features, x is our input data, and $(\phi_f(x))_{f \in \mathcal{F}}$ are features that we draw from the data in x. For example, x could be a movie, while $\phi_1(x)$ could be the genre, $\phi_2(x)$ could be the year the movie was made, and $\phi_3(x)$ could be the number of minutes of high action in the movie. After deciding to display movie x, we then observe the response y which might be the number of times the movie is viewed by online users when it is displayed on their login page.

Instead of observing a sample observation $\hat{\mu}_x^n$ of the unknown truth μ_x, we are going to observe $\hat{y}^n$ of the function $f(x|\theta)$. As before, if we decide to run experiment

Optimal Learning. By Warren B. Powell and Ilya O. Ryzhov

$x = x^n$, we then observe $\hat{y}^{n+1}$, staying with our standard convention of using the iteration counter to tell us which experiments the variable is allowed to see.

We are going to need to manipulate our linear model. For this reason, we are going to adopt the standard notation of statistics and let $x_1, \ldots, x_K$ be the features (instead of $\phi_f(x)$). This means that we are going to equate the experiment "x" with the features of the experiment $(x_1, \ldots, x_K)$.

$$\hat{y}^{n+1} = \theta_0 + \theta_1 x_1^n + \theta_2 x_2^n + \ldots + \theta_K x_K^n + \varepsilon^{n+1}$$

where ε^{n+1} is typically assumed to be a normally distributed error term with mean 0 and variance σ^2.

The most common motivation for using a parametric model is that the number of alternatives M may be extremely large (say, 100,000 or more), or even infinite (if x is continuous). However, there are applications where we know something about the structure of the function that we can capture with a parametric model. For example, we may be trying to estimate the maximum of a concave function. A quadratic approximation may capture this quite nicely, whereas a discretized approximation may easily lose the natural concavity of the function.

In classical regression model applications, we are given a series of observations $x^1, x^2, \ldots, x^n$, where for each set of explanatory variables x^m there is an observation $\hat{y}^{m+1}$. (In traditional batch statistics, we would write $\hat{y}^m$ as the observation corresponding to experiment x^m, but in our sequential setting, it makes more sense to choose x^m and then observe $\hat{y}^{m+1}$.) This data is then used to find a parameter vector θ that minimizes the mean squared error between the predicted model and the actual observations. We often do not have any choice in how the experiments $x^1, x^2, \ldots, x^n$ are chosen.

The problem of determining how to choose a set of experiments is a popular topic in the literature on statistical design of experiments. This literature, however, typically focuses on problems where a design (that is, a sequence of experiments) is chosen before any observations are made (for more on this topic, see Chapter 2). This strategy reflects the nature of the objective function that is used in this work. If the goal is to minimize variance, we can exploit the property that the variance of an estimate is a function of the experiments x^n and not the observations $\hat{y}^n$.

In this chapter, we show how the knowledge gradient (for offline or online learning) can be extended to problems where the belief is captured by a model that is linear in the parameters. This result will allow us to tackle problems with thousands of alternatives, capturing correlations in the beliefs between these alternatives but without having to store a covariance matrix with thousands of rows and columns. The complexity of most of the steps in our adaptation of the knowledge gradient will be determined by the number of parameters rather than the number of alternatives.

7.1 APPLICATIONS

It is useful to start with some real applications to provide context to the discussion that follows.

7.1.1 Maximizing ad clicks

Imagine that you are an Internet company. You might be selling a set of products, or perhaps you are Google selling ad space. Either way, you may have a set of ads (and possibly thousands of ads) that you could post on the Internet, and you want to find the ad that generates the most ad clicks (the number of times that someone clicks on an ad, indicating interest).

How do you choose which ads to post? Let D_i be the data describing the ith ad. D_i might include all the text, along with descriptors of any graphics. The hard part of this project requires identifying important *features*, which are often represented using a device called *basis functions*. We let $\phi_f(D)$ be a particular basis function in a set $f \in \mathcal{F}$. Examples of basis functions might include

$$\begin{aligned}
\phi_1(D) &= \text{The number of words in the ad,} \\
\phi_2(D) &= \text{The number of times the word "iphone," "itouch" or "ipad" appears,} \\
\phi_3(D) &= \text{1 if the ad involves wireless communication, 0 otherwise,} \\
\phi_4(D) &= \text{1 if the ad involves food, 0 otherwise,} \\
\phi_5(D) &= \text{1 if the ad has color graphics, 0 otherwise,} \\
\phi_6(D) &= \text{1 if the ad claims to save money, 0 otherwise,} \\
\phi_7(D) &= \text{1 if the ad involves travel or vacations, 0 otherwise.}
\end{aligned}$$

We first quickly realize that a "basis function" is the same as an independent variable in our linear regression model (different words for the same thing). Also, we can capture both individual features as well as combinations of features. For example, some people might be attracted to ads that involve food, while others enjoy travel, but what about ads that do both? We can reflect the combined contribution by adding the marginal effect of each, but we can also introduce a basis function that capture whether an ad covers food in exotic locations.

Using the notation of basis functions, we would write our linear model as

$$Y = \theta_0 + \sum_{f \in \mathcal{F}} \theta_f \phi_f(D) + \varepsilon.$$

We can make this a bit more compact by introducing the basis function $\phi_0(D) = 1$. Now let $\phi(D)$ be a column vector of basis functions, and θ be a column vector of the coefficients. We can then write

$$Y = \bar{\theta}^T \phi + \varepsilon.$$

There are numerous variations of this basic problem which involve identifying websites (or documents) that achieve a particular purpose. For example:

- Researching information about a company that might predict a change in the stock price - The observable information might be an actual change in stock price (which can be automated) or it might involve showing the website to a domain expert who

can assess its importance. Since the domain expert is a limited resource, we have to be careful in our selection of websites to be evaluated.

- Finding documents that provide information on terrorist activity - Similar to the previous item, we can scan millions of websites, but we have to choose which ones we should show to a domain expert for evaluation. This allows us to build up a dataset that can be used to calibrate the importance of different features.

7.1.2 Dynamic pricing

Now consider the same Internet company from the point of view of pricing. Perhaps we are selling textbooks, DVDs, or music downloads. In any case, the Internet offers us a great deal of freedom in setting prices for our product. We can choose a price, wait for a period of time and observe the resulting revenue, then adjust the price in response. Of course, this is an example of online learning, because our objective is to maximize total revenue.

To keep the modeling simple, we are going to assume that demand is a linear function of price which can be written

$$D(p) = \theta_0 + \theta_1 p,$$

where presumably $\theta_1 < 0$ to create a downward sloping demand function. We do not know θ_0 and θ_1, but we assume that we can charge a price p and then make a noisy observation of the demand $D(p) = \theta_0 + \theta_1 p + \varepsilon$. When we charge a price p, the revenue we earn is given by

$$R(p) = pD(p).$$

The pricing problem also arises outside the e-commerce setting. For example, it could apply to a small business that has a lot of room to change its pricing decisions, but is not prominent enough on the market to be able to change the underlying revenue curve. The business then has to adapt to an existing, unknown true curve. An interesting example of such an enterprise is a recharging station for mobile phones in a developing country. For example, cell phone use in rural Africa is currently experiencing prodigious growth, because it is much easier to build a single cell phone tower in a rural area than to stretch a land line out to every home. Mobile phones can serve as versatile and affordable tools even for many who are not connected to the electric power grid. An entrepreneur with a small power generator, such as a car battery or a solar panel, can serve many users; the question is how the service should be priced.

7.1.3 Housing loans

The Small Business Administration (SBA) plays the role of granting loans to people whose homes have been destroyed by hurricanes and other natural events. As with any bank, the SBA has to identify people who are good loan prospects. By granting a loan, the SBA is able to observe which people eventually replay the loan. This

information can then be used in a regression model to improve future predictions of whether someone might default on a loan. Explanatory variables might include

- The term (length) of the loan.
- The credit score of the applicant.
- Number of months employed.
- Income.
- Did the applicant own the home that was damaged or destroyed?
- Amount of the loan relative to the size of the loss from the storm.
- Amount of collateral.

Normally the SBA would grant loans to people whose probability of repayment is over some amount. However, it is possible for the SBA to grant some loans to applicants whose probability of repayment might be lower simply for the information that might improve their predictive ability for loan defaults.

7.1.4 Optimizing dose response

It is often the case that people respond differently to a particular dosage of a medication, forcing physicians to experiment with different dosages. It is natural to try to predict a patient's response (lowering blood sugar, controlling high blood pressure, raising the red blood cell count) using a statistical model. However, sometimes it is necessary to experiment with different dosages to help improve the model (which may involve a term unique to an individual patient).

We might envision a statistical model that predicts response as a function of a patient's body mass index, gender, age, ethnicity and other elements of a patient's history. However, it is often the case that we want to fit a nonlinear function that relates patient response to a nonlinear function which might look like

$$P_i = \frac{e^{U(x_i|\theta)}}{1 + e^{U(x_i|\theta)}},$$

where $U(x|\theta)$ is a linear function of independent variables with the general form

$$U(x|\theta) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots.$$

In this model, P_i gives the proportion of people in a group i with a specific set of attributes $x_i = (x_{i1}, x_{i2}, \ldots)$ who respond to a particular dosage. This is an example of a parametric model that is *nonlinear* in the parameter vector θ. Belief models that are nonlinear in the parameters cause problems with methods such as the knowledge gradient because we lose conjugacy (see Chapter **??** for an in depth analysis of this particular belief model). However, we can overcome this problem by introducing the transformation

$$\bar{P}_i = \ln\left(\frac{P_i}{1 - P_i}\right).$$

Using this transformation, we obtain the linear regression

$$\bar{P}_i = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots.$$

Now we have the response relationship expressed as a linear regression. We can use this model given estimates of θ to optimize a dosage strategy. This strategy, however, depends on estimates of the parameter vector θ. We may wish to attempt dosages simply to learn more about this relationship.

7.2 A BRIEF REVIEW OF LINEAR REGRESSION

Assume we have n experiments of a vector $x = (x_1, \dots, x_K)$. If x^m is the mth experiment, using the indexing convention we have used throughout this volume, we let $\hat{y}^{m+1}$ be the observation corresponding to x^m. Recall that our indexing system reflects the property that x^m depends on the observations $\hat{y}^1, \dots, \hat{y}^m$, and we observe $\hat{y}^{m+1}$ *after* we have chosen x^m. Let

$$x^n = \begin{pmatrix} x_1^n \\ x_2^n \\ \vdots \\ x_K^n \end{pmatrix}$$

be a K-dimensional column vector of observations. Often we will let $x_1 = 1$ to represent a constant term. Letting θ be the column vector of parameters, we can write our model as

$$\hat{y} = \bar{\theta}^T x + \varepsilon,$$

where we assume that the errors $(\varepsilon^1, \dots, \varepsilon^n)$ are independent and identically distributed. Since $\bar{\theta}^n$ is our estimate of θ after n observations, our best estimate of y is given by

$$\hat{y}^n = (\bar{\theta}^n)^T x^n.$$

7.2.1 The normal equations

We wish to find a parameter vector θ that solves

$$\min_{\theta} \sum_{m=0}^{n-1} \left(\hat{y}^{m+1} - \sum_{k=1}^{K} \theta_k x_k^m \right)^2. \tag{7.1}$$

Let $\bar{\theta}^n$ be the optimal solution for this problem. We can solve this problem very simply. Let X^n be the n by K matrix

$$X^n = \begin{pmatrix} x_1^1 & x_2^1 & & x_K^1 \\ x_1^2 & x_2^2 & & x_K^2 \\ \vdots & \vdots & \cdots & \vdots \\ x_1^n & x_2^n & & x_K^n \end{pmatrix}.$$

The vector of observations of the dependent variable is given by

$$\hat{Y}^n = \begin{pmatrix} \hat{y}^1 \\ \hat{y}^2 \\ \vdots \\ \hat{y}^n \end{pmatrix}.$$

The optimal parameter vector $\bar{\theta}^n$ (after n observations) is then given by

$$\bar{\theta}^n = [(X^n)^T X^n]^{-1}(X^n)^T \hat{Y}^n. \tag{7.2}$$

From the normal equations, we can compute the covariance matrix for $\bar{\theta}^n$ using a simple matrix identity. If u and w are scalar random variables where $u = Aw$, then we know that $Var(u) = A^2 Var(w)$. If u and v are vectors, and A is a suitably dimensioned matrix, then we can write $Var(u)$ (the covariance matrix for the vector u) as

$$Cov(u) = ACov(w)A^T$$

where $Cov(w)$ is the covariance matrix of w. Recall that for matrices A and B, $AB^T = (BA^T)^T$. Also keep in mind that $[(X^n)^T X^n]^{-1}$ is symmetric. Applying this identity to (12.2), where $A = [(X^n)^T X^n]^{-1}(X^n)^T$, we obtain

$$\begin{aligned} Var(\bar{\theta}^n) &= [(X^n)^T X^n]^{-1}(X^n)^T Cov(\hat{Y}^n)\left([(X^n)^T X^n]^{-1}(X^n)^T\right)^T \\ &= [(X^n)^T X^n]^{-1}(X^n)^T Cov(\hat{Y}^n)(X^n)[(X^n)^T X^n]^{-1}. \end{aligned}$$

Since the elements of $\hat{Y}^n$ are independent, $Cov(\hat{Y}^n) = \sigma_\epsilon^2 I$ where I is the identity matrix and σ_ϵ^2 is the variance of our experimental error . This allows us to write

$$\begin{aligned} \Sigma^{\theta,n} &= [(X^n)^T X^n]^{-1}(X^n)^T X^n [(X^n)^T X^n]^{-1}\sigma_\epsilon^2 \\ &= [(X^n)^T X^n]^{-1}\sigma_\epsilon^2. \end{aligned}$$

It is important to realize that the matrix X^n is n by K, so computing $\Sigma^{\theta,n}$ is not too difficult.

7.2.2 Recursive least squares

There is a shortcut that we can use to do this recursively. The updating equation for $\bar{\theta}^n$ can be computed using

$$\bar{\theta}^n = \bar{\theta}^{n-1} + \frac{1}{\gamma^n} B^{n-1} x^n \varepsilon^n, \tag{7.3}$$

where ε^n is the error given by

$$\varepsilon^n = \hat{y}^n - \bar{\theta}^{n-1} x^{n-1}. \tag{7.4}$$

The matrix $B^n = [(X^n)^T X^n]^{-1}$. This can be updated recursively without computing an explicit inverse using

$$B^n = B^{n-1} - \frac{1}{\gamma^n}(B^{n-1} x^n (x^n)^T B^{n-1}). \tag{7.5}$$

The scalar γ^n is computed using

$$\gamma^n = 1 + (x^n)^T B^{n-1} x^n. \tag{7.6}$$

Note that if we multiply (7.5) through by σ_ϵ^2 we obtain

$$\Sigma^{\theta,n} = \Sigma^{\theta,n-1} - \frac{1}{\gamma^n}(\Sigma^{\theta,n-1} x^n (x^n)^T \Sigma^{\theta,n-1}),$$

where we scale γ^n by σ_ϵ^2, giving us

$$\gamma^n = \sigma_\epsilon^2 + (x^n)^T \Sigma^{\theta,n-1} x^n.$$

Note that we had to multiply each B^{n-1} in the second term on the right of (7.5) by σ_ϵ^2 so we also divided the second term by σ_ϵ^2, which we did by scaling γ^n.

Thus, we have compact updating equations dimensioned only by the number of parameters, rather than the number of alternatives.

The recursive updating formulas, aside from allowing us to avoid an expensive matrix inversion, allows us to handle an issue that we have not yet considered. There are problems where the observations are nonstationary, which means they are coming from a process that is changing over time. In such settings, we may not want to give all the observations equal weight. We can do this by replacing the objective function (12.1) with

$$\min_\theta \sum_{m=0}^{n-1} \lambda^{n-m} \left(\hat{y}^{m+1} - \sum_{k=1}^{K} \theta_k x_k^m \right)^2. \tag{7.7}$$

Here, λ is a discount factor that we use to discount older observations. If we use this objective function, our recursive updating equations change only slightly to

$$\gamma^n = \lambda + (x^n)^T B^{n-1} x^n, \tag{7.8}$$

instead of (7.7), while we replace (7.5) with

$$B^n = \frac{1}{\lambda}\left(B^{n-1} - \frac{1}{\gamma^n}(B^{n-1} x^n (x^n)^T B^{n-1}) \right).$$

Setting $\lambda = 1$ gives us the original updating equations. Using smaller values of λ reduces the weight on older observations, but also increases the variance of the estimates.

7.2.3 A Bayesian interpretation

Classical linear regression is, of course, a frequentist method, but we can put linear regression into our standard Bayesian vocabulary. Let μ^i be the true value of alternative i which, in our Bayesian setting, is a random variable given by

$$\mu^i = \theta x^i.$$

Just as μ is a random variable (our truth), so is θ, which is the truth about the effect of each feature. We learned that we get tremendous benefits from exploiting the covariance between two alternatives. The covariance between μ^i and μ^j is given by

$$\begin{aligned} Cov(\mu^i, \mu^j) &= Cov\left(\sum_k \theta_k X_k^i, \sum_k \theta_k X_k^j\right) \\ &= \sum_{k,k'} X_k^i X_{k'}^j Cov(\theta_k, \theta_{k'}) = \sum_{k,k'} X_k^i X_{k'}^j \Sigma^\theta_{k,k'} \\ &= e_i^T X \Sigma^\theta X^T e_j, \end{aligned}$$

where $\Sigma^\theta_{k,k'} = Cov(\theta_k, \theta_{k'})$ is the covariance between the regression coefficients θ_k and $\theta_{k'}$. If Σ is our original covariance matrix between our beliefs of different alternatives, Σ and Σ^θ are related by

$$\Sigma = X\Sigma^\theta X^T.$$

This means that we can write our vector of truths about the value of each alternative as

$$\mu \sim N(\bar{\theta}^T X, X\Sigma^\theta X^T).$$

Here, X is a matrix with a row and column for *each* alternative, which means that it has M rows (one for every alternative which may be an extremely large number) and K columns (one for every feature, which is generally not too large). We then let

$$\Sigma^n = X\Sigma^{\theta,n} X^T$$

be the covariance matrix among all M alternatives (hence Σ^n is $M \times M$, while $\Sigma^{\theta,n}$ is $K \times K$). This means that we are getting an estimate of the covariance between every pair of alternatives, including those which we have not tested yet, from the much more compact matrix Σ^θ. Our goal is to avoid having to explicitly compute and store the complete matrix Σ^n.

The important result is that we can compute a covariance matrix among all *possible* alternatives, using only the covariance matrix $\Sigma^{\theta,n}$ among the parameters, and the matrix X of attributes of each possible alternative. Updating $\Sigma^{\theta,n}$ from data is relatively easy using our recursive formulas. Computing Σ^n, given the potentially large number of rows, is still quite manageable given that it involves fairly simple calculations.

7.2.4 Generating a prior

To perform learning in our Bayesian setting, we have to generate a truth, and then apply a learning algorithm (such as the knowledge gradient) to try to discover the truth. When we had a discrete set of alternatives with normally distributed beliefs, we could simply sample from the normal distribution we used as a prior.

There are several ways to generate a prior for these more complex settings. One, of course, is to collect a small sample of observations from an initial training set

$x^0, \ldots, x^{L-1}$, producing observations $\hat{y}^1, \ldots, \hat{y}^L$. If L is larger than the number of parameters to be estimated, we can create an initial estimate of the parameter vector $\bar{\theta}^0$ using the normal equations in equation (12.2). We can then use this initial estimate to obtain an estimate of the variance σ^2 (if this is not known already). In the context of Bayesian learning, this becomes the same empirical Bayes strategy that we have already discussed.

Using an estimate of σ_ϵ^2 obtained from this model, we can estimate an initial covariance matrix for θ from

$$\Sigma^{\theta,0} = [(X^L)^T X^L]^{-1}\sigma_\epsilon^2 \tag{7.9}$$

where X^L is our initial set of observations from the first L observations.

We cannot use equation (7.9) if we have fewer than L observations, and we need at least $L+1$ observations if we are going to obtain a valid estimate of σ_ϵ^2. However, we can still get initial estimates of θ and Σ^θ with fewer than L observations, as long as we have some sort of prior. If we have some starting point, we can use the recursive equations (7.3)-(7.7) for an initial set of training points which may be chosen at random.

A useful strategy for generating an initial set of estimates is to write the regression equation in the form

$$y = \theta_0 + \theta_1(x_1 - \bar{x}_1) + \theta_2(x_2 - \bar{x}_2) + \ldots + \varepsilon,$$

where $\bar{x}_i$ is viewed as an average or typical value of the independent variable x_i. In this model, θ_0 should be an average (or typical) value of the observations y, while θ_i captures the marginal impact of x_i on y. For example, consider our pricing model above where the demand $D(p)$ is given by

$$D(p) \approx \theta_0 + \theta_1(p - \bar{p}).$$

Here, we have written the independent variable as the deviation from what we might call a typical price give by $\bar{p}$. In this model, θ_0 would be a prior estimate of the typical demand, while θ_1 captures the effect of changes in the price on demand.

Estimating $\Sigma^{\theta,0}$ is somewhat tricker. An estimate of the diagonal terms can be found by trying to estimate a confidence interval for each coefficient θ_i. So, we might think the right coefficient for our demand curve is $\theta_1 \approx -0.07$, and we might then be willing to say that the right coefficient is probably in the range (-0.03,-0.10). From this, we could infer a standard deviation of, say, 0.015, and use this to create the diagonal elements of $\Sigma^{\theta,0}$.

Estimating the off-diagonal elements of $\Sigma^{\theta,0}$ is inherently more complicated. A first order approximation is to just set them to zero, but this can be dangerous. Consider the set of potential demand functions shown in Figure 7.1. We may be uncertain about the precise line that describes the demand-price tradeoff, but we may feel that they all go through a certain region that might reflect the current demand and current price. The implication is that smaller values of θ_1 correspond to smaller values of θ_0, so our prior on these coefficients would not be independent.

Once we have an initial estimate of a mean vector $\bar{\theta}^0$ and covariance matrix Σ^0, we can generate a truth μ from a normal distribution using the techniques described in Section 2.7.

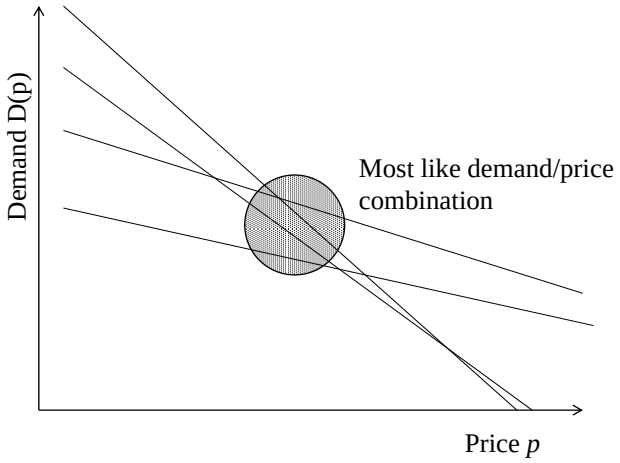


Figure 7.1 Illustration of range of potential demand functions.

7.3 THE KNOWLEDGE GRADIENT FOR A LINEAR MODEL

Now we are ready to derive the knowledge gradient using our linear belief model. We begin with the calculations, which are very similar to the knowledge gradient for correlated beliefs, except that now the correlations are between the parameters. We then contrast the behavior of the knowledge gradient when using a linear belief model versus when we use a lookup table model.

7.3.1 Calculations

Recall that the calculation of the knowledge gradient requires computing the function $h(a, b(j))$ where a is a vector with element $a_i = \bar{\theta}_i^n$ giving the estimate of the value of the ith alternative, and $b(j)$ is a vector with element $b_i(j) = \tilde{\sigma}_i^n(j)$, which is the conditional variance of the change in $\bar{\theta}^{n+1}$ from measuring alternative j. The function $h(a, b)$ is given by

$$h(a, b(j)) = \sum_{i=1}^{M-1} (b_{i+1}(j) - b_i(j)) f(-|c_i(j)|) \tag{7.10}$$

where

$$c_i(j) = \frac{a_i - a_{i+1}}{b_{i+1}(j) - b_i(j)}.$$

Refer to Section 4.5 for the details of computing the knowledge gradient for correlated beliefs.

Keeping in mind that M may be a very large number, we have to use care in how this is computed. We can use $\bar{\theta}_i^n = \bar{\theta}^n x^i$ to compute the expected value, which does not depend on the alternative j that we are measuring.

The harder part is computing $\tilde{\sigma}^n(j)$, which is a vector giving the change in the variance of each alternative i assuming that we have chosen to measure alternative j. Recall that

$$\tilde{\Sigma}^n(j) = \frac{\Sigma^n e_j (e_j)^T}{\sqrt{\Sigma^n_{jj} + \sigma^2_\epsilon}},$$

which gives us the change in the covariance matrix from measuring an alternative j. The matrix $\tilde{\Sigma}^n(j)$ is $M \times M$, which is quite large, but we only need the jth row, which we compute using

$$\tilde{\sigma}^n(j) = \frac{\Sigma^n e_j}{\sqrt{\Sigma^n_{jj} + \sigma^2_\epsilon}}. \tag{7.11}$$

Since $\Sigma^n = X\Sigma^{\bar{\theta}^n}(X)^T$, let X_j be the jth row of X. Then

$$\tilde{\sigma}^n(j) = X_j \Sigma^{\bar{\theta}^n} X^T.$$

We still have to multiply the $K \times K$-dimensional matrix $\Sigma^{\bar{\theta}^n}$ times the $K \times M$-dimensional matrix X^T, after which we have to compute equation (7.10) to find the knowledge gradient for each alternative. Even for problems with tens of thousands of alternatives, this can be executed in a few seconds, since K is much smaller than M.

The real value of the linear belief model is that it allows us to compute the knowledge gradient for much larger sets of alternatives, while still capturing the relationship between our beliefs at different values of x. If we were using our lookup table model with correlated beliefs, it means that we have to manipulate an $M \times M$ matrix, which becomes quite clumsy when M is more than 1,000. With our linear belief model, we only have to manage the parameter covariance matrix Σ^θ, which is much smaller.

In the next section, we take a look into how the parametric model changes the behavior of the learning process.

7.3.2 Behavior

The section above shows how to calculate the knowledge gradient, but does not address the question of how it changes its behavior. Consider, for example, the behavior of a smooth function estimated using a fine-grained lookup table representation as depicted in figure 7.2. The figure to the left shows both the true and estimated function after three experiments. The figure on the right shows the knowledge gradient, which is lowest at each of the three points where experiments have been run. These drops are typical since an experiment reduces the uncertainty about the function at that point. The knowledge gradient can actually rise if the function is sufficiently large.

Now consider what happens when we use a parametric model. Assume that the x is a scalar (such as the price of a product or dosage of a drug) and that the true response is quadratic in x, giving us the belief model

$$f(x|\theta) = \theta_0 + \theta_1 x + \theta_2 x^2. \tag{7.12}$$

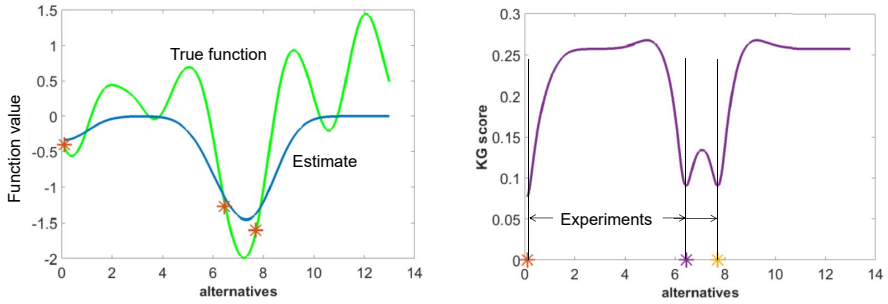


Figure 7.2 Lookup table representation of true function and current estimate after three experiments (left), and the knowledge gradient after three experiments (right) (from **?**).

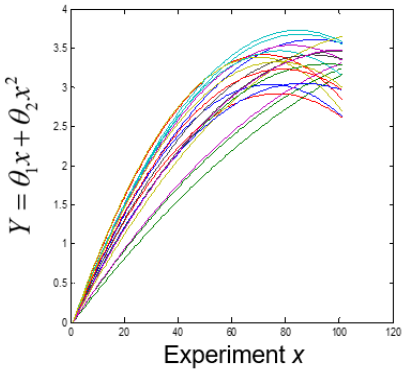


Figure 7.3 A family of possible values quadratic functions assuming $\theta_0 = 0$. (from **?**).

To demonstrate a point, we are going to assume that $\theta_0 =$, but that θ_1 and θ_2 are random (specifically, multivariate normal). Figure 7.4 depicts a sample of these functions, all going through the origin. Our problem is to figure out which experiments to run to learn our function as quickly as possible.

Figure 7.4 shows a plot of the knowledge gradient computed using the equations given above, after 0, 1, 2, and 3 experiments. Initially, the knowledge gradient increases left to right, encouraging the largest value of x so that we experiment with the function where there is the greatest variability. After that, the knowledge gradient becomes bimodal, with the maximum switching between a point at around 1/3 of the maximum, and the maximum. This basic shape and pattern does not change even after a large number of experiments.

The behavior of the knowledge gradient shown in figure 7.4 is quite different than what we saw with a lookup table. It also no longer has the behavior that the knowledge gradient shrinks wherever we run an experiment. In short, working with a parametric belief model produces a completely different behavior.

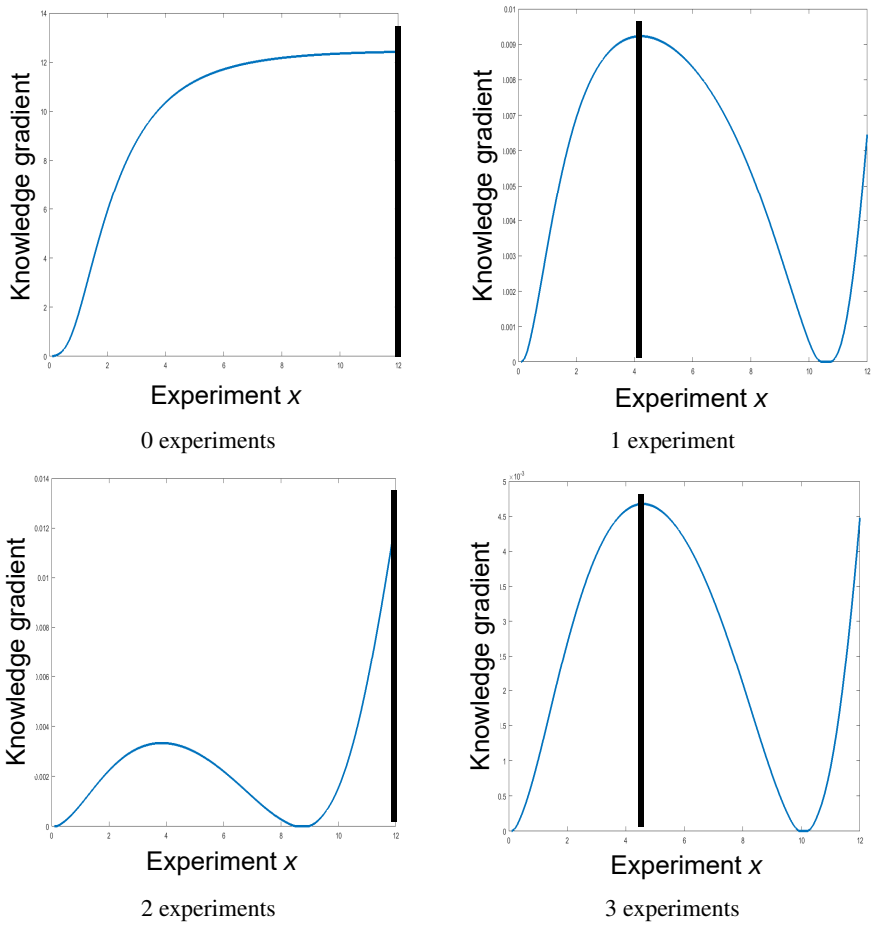
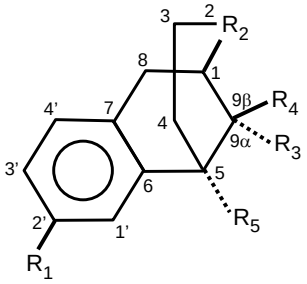


Figure 7.4 The knowledge gradient for the quadratic belief model given in figure **??** after 1, 2, 3 and 4 experiments (top left to bottom right) (from **?**).

7.4 APPLICATION TO DRUG DISCOVERY

A nice area of application of this logic is the design of molecules to accomplish some purpose, such as storing energy or curing cancer. We start with a base molecule such as that shown in Figure 7.5, where there are five locations $(R_1, R_2, \ldots, R_5)$. At these locations, we can add small molecular components called *substituents*, such as those listed to the right of the molecule in the figure. While we may not be able to put every substituent at every location (a chemist would understand which combinations make sense), the number of permutations and combinations can be quite large. For example, this base molecule and associated set of substituents produced 87,000 possible combinations.

The challenge with testing these molecules is that they are fairly difficult to create. It is necessary to design actual chemical processes to produce these compounds. A knowledgeable chemist can create a number of potential molecular compounds on a



Potential substituents:

F	OCH_3
OH	CH
CH_3	NO
$OCOCH_3$	CI
OCOCH	OCOCH

Figure 7.5 Illustration of base molecule with five substituent locations (from Katz & Ionescu 1977).

white board, but once we decide which compound to test next, the actual testing is a project that can take a day to several days. This is the reason why it is important to sequence the experiments carefully.

The problem of designing molecules arises in many settings. The work in this section was motivated by a problem of finding compounds to cure a form of cancer, but the numerical work was done using data taken from the literature which was motivated by a variety of applications. All we require is that we are trying to choose a compound that maximizes (or perhaps minimizes) some metric, such as killing cancer cells.

In practice, the knowledge gradient can be used to produce a ranked list of potential molecules to be tested next. An attraction of this strategy is that it allows a scientist to use expert knowledge to capture dimensions other than the performance of a molecule in terms of a specific metric such as killing cancer cells. For example, a molecule might be toxic, hard to manufacture or insoluble in water.

To put this problem in the context of our model with linear beliefs, let $i = 1, \ldots, I$ be the set of sites (five in our example) and let $j = 1, \ldots, J$ be the set of potential substituents (10 in our example). Let

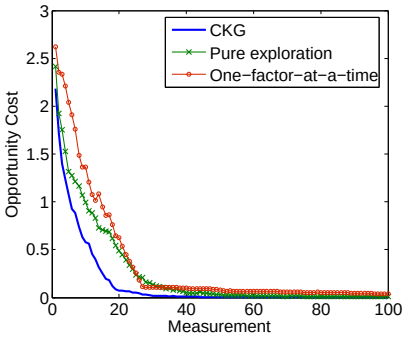
$$X_{ij} = \begin{cases} 1 & \text{If we put the } j\text{th substituent at the } i\text{th site,} \\ 0 & \text{Otherwise.} \end{cases}$$

Now let Y be an observation of the performance of a particular molecule, which might be the percentage of cancer cells that are killed. We might model the performance of a molecule using

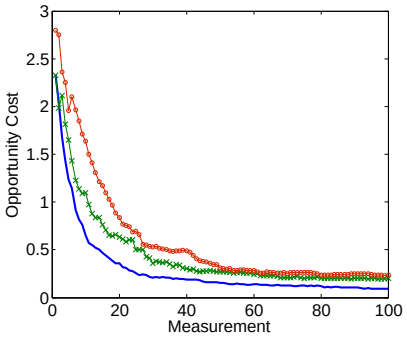
$$Y = \theta_0 + \sum_{i=1}^{I} \sum_{j=1}^{J} \theta_{ij} X_{ij}. \tag{7.13}$$

With this simple model, if we have five sites and 10 substituents, we have 51 parameters to estimate (including the constant intercept θ_0).

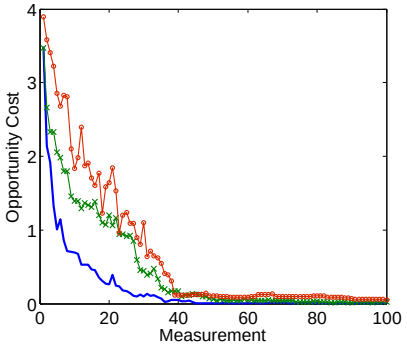
To determine the effectiveness of the knowledge gradient, it is necessary to create a truth (following our standard practice) and then try to discover the truth using different



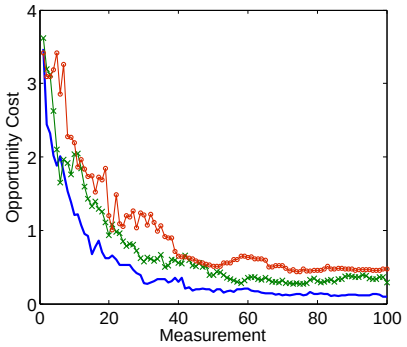
7.6a: Average opportunity cost over 100 runs using a data set of 2640 compounds and a noise standard deviation of 0.1.



7.6b: Average opportunity cost over 100 runs using a data set of 2640 compounds and a noise standard deviation of 0.5.



7.6c: Average opportunity cost over 10 runs using a data set of 87120 compounds and a noise standard deviation of 0.1.



7.6d: Average opportunity cost over 10 runs using a data set of 87120 compounds and a noise standard deviation of 0.5.

Figure 7.6 Comparison of the knowledge gradient to pure exploration and a standard experimental design on a medium and large molecule with different levels of noise (from Negoescu et al. 2010).

methods. The truth was created by using data from an initial set of experiments that was then used to perform an initial fit of the regression model in equation (7.13). This model was then assumed to give us an initial parameter vector $\bar{\theta}^0$ that we used as our prior, and an initial covariance matrix $\Sigma^{\theta,0}$. We made our standard assumption that the true parameter vector $\theta \sim N(\bar{\theta}^0, \Sigma^{\theta,0})$ follows a normal distribution. We would generate a sample realization of a truth $\theta(\omega)$ by sampling from this distribution.

Once we have a truth (which means we now have a "true" parameter vector θ), we run our experiments by choosing a molecule to test (using some policy), and then sampling its performance using equation (7.13). It is important to recognize that this method of testing a learning policy assumes that the linear *model* is accurate, even if we do not know the true parameter vector. But, there is nothing stopping us from generating observations from some other model, and then using linear regression as an approximation.

The knowledge gradient was compared to a pure exploration policy (choosing molecules at random) and a simple experimental design policy which tests one factor (substituent) at a time, cycling through all the substituents. Each of these strategies are used in an off-line setting (since this is laboratory experimentation) to collect data

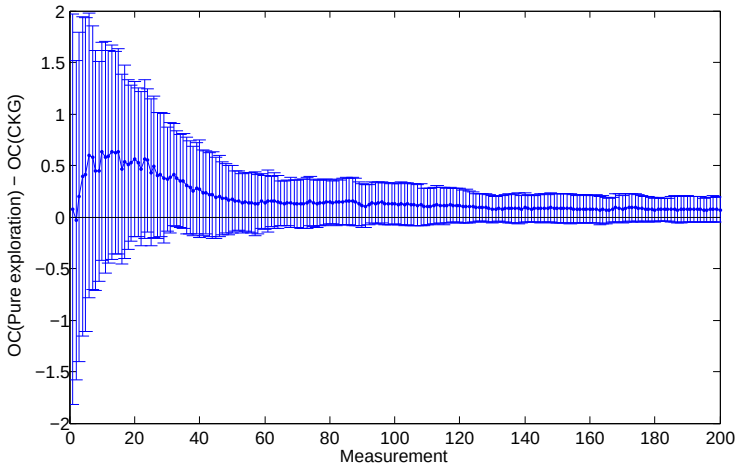


Figure 7.7 Mean and standard deviation of difference in opportunity cost between pure exploration and KGCB using 75 sample paths of 10000 compounds each and a noise standard deviation of 0.38 (from Negoescu et al. 2010).

to fit our regression model. We then use this regression model to decide which of a much larger set of molecules is best.

Figure 7.6 shows the expected opportunity cost for the knowledge gradient, pure exploration and the one-at-a-time factor design on molecules with 2,640 and 87,120 variations, and with experimental noise of 0.1 (lower than that observed from actual data) and 0.5 (higher than that observed from actual data). These experiments suggest that the knowledge gradient consistently outperforms the competition, with a fairly significant improvement if we wanted to stop at around 20 experiments. Note that we were able to find a near-optimal molecule within approximately 30 experiments.

Care has to be used when evaluating the performance of a learning policy based on averages. Figure 7.7 shows the actual spread in the results from 75 sample paths, comparing knowledge gradient against pure exploration (positive means that KG is outperforming pure exploration). After 20 experiments, the knowledge gradient policy is almost always outperforming pure exploration. The difference is much more mixed during the first 20 observations, which should be expected. When there is a lot of uncertainty and not enough information to make sensible choices, random exploration can work just as well as the knowledge gradient on specific sample paths. There is no evidence that the knowledge gradient ever underperforms pure exploration; but it is certainly possible that pure exploration can outperform knowledge gradient in the early iterations. Interestingly, the knowledge gradient significantly outperforms pure exploration *on average* in the early iterations, but the spread is quite high.

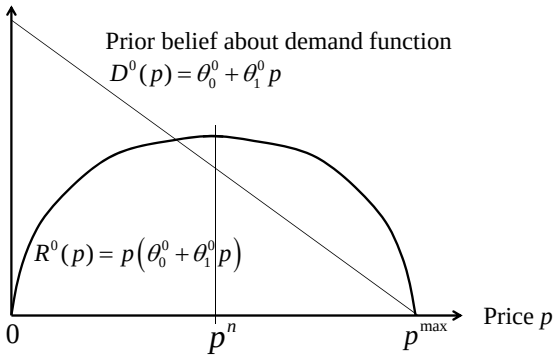


Figure 7.8 Prior demand function and revenue curve.

7.5 APPLICATION TO DYNAMIC PRICING

Let us briefly return to the dynamic pricing application in Section 7.1.2. We are interested in the pricing decision of the African entrepreneur running a cell phone recharging station. This is a small but profitable operation. For example, the entrepreneur can purchase a car battery, charge it in town, and then travel to a village to charge phones for a fee that can be anywhere between $\$0.10$ and $\$0.50$. An alternative with a higher startup cost may be to purchase a small solar system (e.g. a set of solar panels) and then sell the generated power. With a good pricing strategy, the recharging station may see 50–70 customers every day. Assuming that the entrepreneur works five days a week, the resulting revenue can be as high as $\$50$–$\$100$ per week.

This problem provides the basic parameters for our example. Every week, the price per use of the recharging station is chosen from the range $[0.10, 0.50]$. The weekly revenue $R(p)$ can be as high as $\$50$. In this example, we use a simple quadratic belief model,

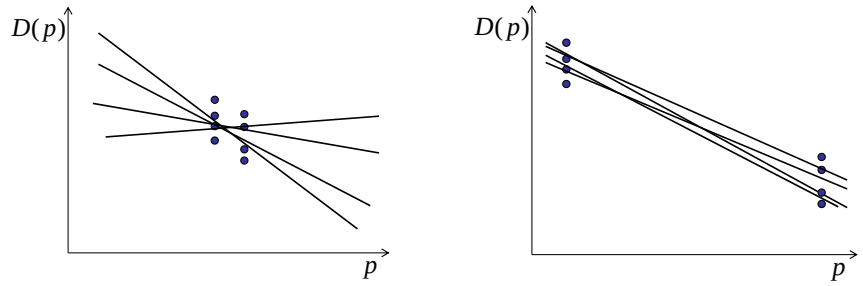
$$R(p) = p\big(\theta_0 + \theta_1 p\big).$$

where we capture the fact that $R(0) = 0$. Our belief about the revenue for the nth week can be written as $R^n(p) = p(\bar{\theta}_0^n + \bar{\theta}_1^n p)$. We would typically start with a prior where $\bar{\theta}_1^0 < 0$, because revenue is likely to be concave (we do not expect it to keep increasing with price forever).

The pricing problem is inherently online: every time we set a price, we collect a revenue. As always, we need to balance the need to maximize revenue with the need to fit a good curve. In this case, we would use the knowledge gradient for online learning, given by

$$X^{KG,n}(s^n) = \arg\max_p R^n(p) + (N-n)\,\nu_p^{KG,n},$$

where ν_p^{KG} is the offline knowledge gradient corresponding to the value of observing demand at price p.



(a) (b)

Figure 7.9 Estimating the demand function using (a) observations near the middle and (b) observations near the endpoints.

Let's consider how different learning policies should behave. Figure 7.8 depicts a possible prior on the demand function and resulting revenue curve. Let p^{max} be the price that drives the demand to 0, which means that we should limit our search to prices in the interval $[0, p^{max}]$. Also let p^n be the price that maximizes revenue given our current belief after n experiments, which is to say

$$p^n = \arg\max R^n(p).$$

If we were to stop after n experiments, p^n is the price that we would charge. Otherwise, we would characterize the decision to charge p^n as a pure exploitation policy.

What would we do if we wanted to focus on estimating the demand function? In Figure 7.9(a), we see that if we focus our energies on observations near the middle, we may obtain a wide range of possible functions as a result of our experimental noise. By contrast, if we focus our observations near the endpoints as in 7.9(b), we obtain much more reliable estimates of the demand function. This behavior is well known in the statistics community.

Measuring near the middle offers the potential for maximizing revenue, but we end up learning almost nothing from the observations. By contrast, if we measure near the endpoints, we learn a lot but earn almost nothing. Ideally, we would like to make observations that strike a balance between these two extremes, a property that might be called "sampling the shoulders." This, in fact, is exactly what the knowledge gradient does, in a way that adapts to the number of observations remaining in our budget.

Figure 7.10 illustrates the behavior of a pure exploitation policy (first row), the knowledge gradient policy for offline learning (second row) and the knowledge gradient policy for online learning (third row). These policies were tested for datasets with a moderate level of observation noise (the left column) and a relatively high level of observation noise (right column). The line with solid dots is the initial prior, while the line with open circles is the true demand. The remaining lines represent the estimates of the demand functions for each of the first 20 observations obtained under each policy. The darker lines represent the early iterations, while the lighter lines (which are more clustered) represent the later iterations.

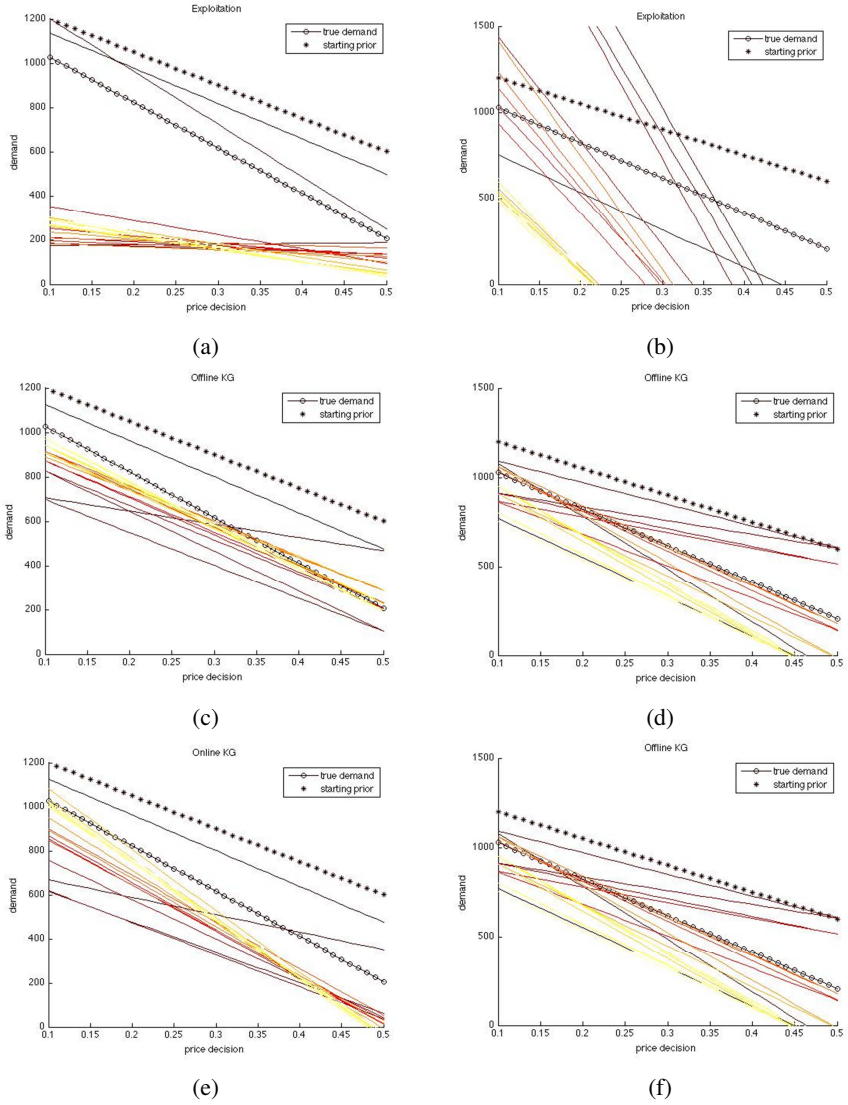


Figure 7.10 The behavior of pure exploitation (first row), offline learning (second row) and online learning (third row) for observations with a moderate level of noise (left column) and a high level of noise (right column).

The results show that using a pure exploitation policy does a terrible job identifying the correct demand curve. By contrast, using an offline policy produces the best results, producing an estimate of the demand curve that is quite close to the truth. The online learning policy strikes a balance between the two, producing an estimate of the demand curve that is clearly not as good as what is obtained using an offline learning policy, but much better than the pure exploitation policy.

The first row of Figure 7.11 shows the actual pricing decisions produced by each policy. The offline knowledge gradient behaves exactly as we would expect given

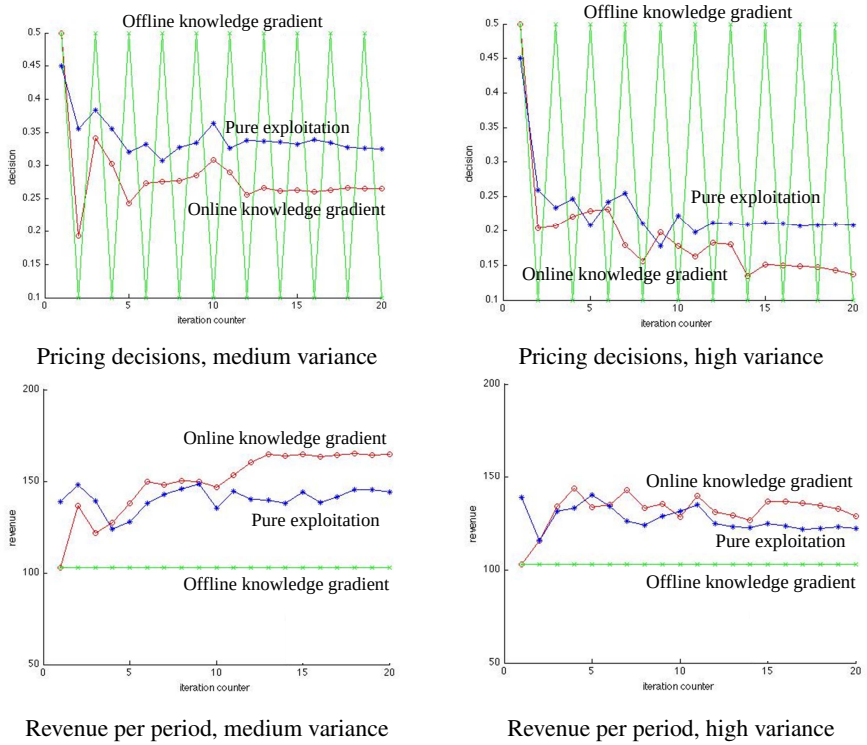


Figure 7.11 The pricing decisions selected by each policy (first row), and revenue earned per period (second row) for moderate and high variance in the experimental noise.

the illustration in Figure 7.9(b). The policy alternates between the two endpoints. This behavior is consistent for both the medium and high variance cases. For the medium variance case, the online knowledge gradient does more exploration in the early iterations before settling in to a single point compared to the pure exploitation policy. Note that the online knowledge gradient approaches a pure exploitation policy toward the later iterations. The only reason that it is measuring different points is that it is converging on a different estimate of the demand curve than pure exploitation, as a result of the exploration in the early iterations.

The second row of Figure 7.10 shows the revenue earned per period. Note that the offline knowledge gradient earns a fixed revenue greater than zero because the endpoints were defined slightly interior to the interval $(0, p^{max})$. The more interesting comparison is between the pure exploitation policy and the online knowledge gradient. For both the medium and high variance cases, the pure exploitation policy produces higher revenues initially, but eventually loses out to the online knowledge gradient policy. Note that the relative improvement of the online knowledge gradient over pure exploitation is largest for the moderate noise case. Again, we think this is to be expected. As the experimental noise increases, it is harder for any policy to learn the correct function. By contrast, we would expect the difference between the two policies to diminish as the experimental noise decreases, because the value of

information from learning will be minimal, which again pushes the online knowledge gradient policy toward the pure exploitation policy.

7.6 BIBLIOGRAPHIC NOTES

Section 7.2 - This is classic material that can be found in many statistical textbooks such as Hastie et al. (2005), which can be downloaded from

`https://web.stanford.edu/~hastie/Papers/ESLII.pdf`

Section 7.3 - 7.4 - This material is based on Negoescu et al. (2011). The drug discovery example is taken from Katz & Ionescu (1977).

Section 7.5 - The experimental results presented here come from work by two undergraduate students at Princeton University. The numerical work was performed by Xiaoyang Long as part of her research for the Program in Applied and Computational Mathematics. The application of pricing cell phone charges was studied by Megan Wong as part of her senior thesis.

PROBLEMS

The exercises below require the knowledge gradient algorithm where the belief model is linear in the parameters. This can be downloaded from

`http://optimallearning.princeton.edu/exercises/KGCBLinReg.m`

An example implementation of the algorithm is given in

`http://optimallearning.princeton.edu/exercises/KGCBLinRegEx.m`

7.1 You are going to replicate the experiments in Section 7.5 where we try to learn the demand as a function of price, while simultaneously trying to maximize revenue. We start by assuming that the demand as a function of price is given by

$$D(p) = \theta_0 + \theta_1 p.$$

Our revenue is given by $R(p) = pD(p)$, and prices are assumed to range between .1 and .5. Assume the true value of $\theta = (\theta_0, \theta_1) = (1233, -2055)$. Normally we would sample this truth from a normal distribution, but we are going to assume a single truth to illustrate how well we discover this truth.

Now assume we start with a high prior $\bar{\theta}^0 = (1350, -1500)$, with a starting covariance matrix of

$$\Sigma^{\theta,0} = \begin{pmatrix} 62500 & 0 \\ 0 & 2{,}500{,}000 \end{pmatrix}$$

a) Set up and run the knowledge gradient with a linear belief model for $N = 20$ iterations assuming a low experimental noise of $\sigma_\epsilon^2 = 62,500$, limiting your search of prices to the range $[.1, .5]$. Do this for both online and offline learning (recall that Section 5.4 describes how to compute the online knowledge gradient from the offline

version). Plot the prices chosen and the updated estimate of the demand curve after each iteration.

b) Now sample prices using a pure exploitation policy which maximizes the revenue at each iteration, and compare your results to those obtained using the online and offline knowledge gradient.

c) Repeat (a) and (b) using $\sigma_\epsilon^2 = 625,000$.

d) Repeat (a) and (b) using $\sigma_\epsilon^2 = 6,250,000$.

e) Compare the performance of the three algorithmic strategies under the different noise levels.

7.2 Repeat exercise 7.1, but this time start with a low prior of $\bar{\theta}^0 = (1350, -4500)$.

7.3 Repeat exercise 7.1, but now assume that the truth is normally distributed around the prior with mean $\bar{\theta}^0 = (\bar{\theta}_0^0, \bar{\theta}_1^0) = (1233, -2055)$ and variance

$$\Sigma^{\theta,0} = \begin{pmatrix} 62500 & 0 \\ 0 & 2{,}500{,}000 \end{pmatrix}$$

CHAPTER 8

NONLINEAR BELIEF MODELS

While linear models will always remain an important tool for approximating functions (remember we mean that they are linear in the parameters), there are some problems where we cannot avoid models that are nonlinear in the parameters. Some examples include:

■ **EXAMPLE 8.1**

We wish to estimate the probability that a customer will purchase an online product with characterized by price p (which is a decision variable) and other features of the product which we denote by $a_1, \ldots, a_K$. We can then describe our product using

$$x = (p, a_1, \ldots, a_K).$$

We think of x as a decision, although the only controllable parameter is p. We may feel that we can model the probability of a sale using a logistic model of the form

$$p(x) = \frac{e^{U(x|\theta)}}{1 + e^{U(x|\theta)}},$$

where $U(x|\theta)$ is a linear function of independent variables with the general form

$$U(x|\theta) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots.$$

We start with an example of a relatively simple nonlinear problem that arises in the context of designing bidding policies. We then transition to more general strategies that can be applied to a wide range of problems.

8.1 AN OPTIMAL BIDDING PROBLEM

Imagine that industrial customers come to you requesting price quotes on contracts to provide a product of some type. This might be laptops for a large company, a component for a product that the customer is building (such as disk drives for the laptops), or materials used in a manufacturing process. You have to quote a price, recognizing that you have to match or beat competing prices. If you win the contract, you might be left wondering if you could have asked for a little more. If you lose the contact, you are kicking yourself if you feel that a slightly lower price would have helped you win the contract.

Figure 8.1 illustrates the economics of different prices. There is a breakeven price p^b, below which you will lose money on the contract. Then there is a price point $\bar{p}_c$ for each customer c. If you quote a price above this number, you will lose the contract. The problem, of course, is that you do not know $\bar{p}_c$. As the figure illustrates, increasing the price above p^b produces dramatic increases in profits, especially for competitive, commodity products.

We are going to assume that we have several opportunities to quote a price and observe a buy/not-buy decision from the customer. We are not allowed to observe $\bar{p}_c$ directly, even after the fact. After each iteration of quoting a price and observing a decision of whether or not to accept the bid, we have an opportunity to use what we have learned before deciding on our next bid. We know that we have to bid prices above p^b, and we know what prices have been accepted in the past. If we discover that we can get higher prices, the impact on our profits can be significant. Our challenge is trying to learn $\bar{p}_c$ while balancing the profits we realize against the information gained while trying higher prices (and potentially losing some contracts).

We can also consider a version of this problem from the customer's point of view. Suppose that there are now multiple sellers offering a single product. A customer makes a bid for the product at a price of his or her choosing. The bid is successful if there is a seller willing to accept it, but the customer does not get to observe the sellers' willingness to sell before making the bid. In this case, higher bids are more likely to be successful, but the customer can save money by finding the optimal price. Bidding too low and failing to secure an offer carries an opportunity cost (perhaps we are forced to wait a period of time before bidding again). We might call this the "Priceline problem," because of its similarities to the business model of the well-known online travel agency.

Overall, we can see a clear potential for optimal learning in the bidding problem. In fact, variations of this problem have attracted a great deal of attention in the revenue management community, precisely from an optimal learning perspective. However, it is less clear precisely how learning should be applied. The problem has a number

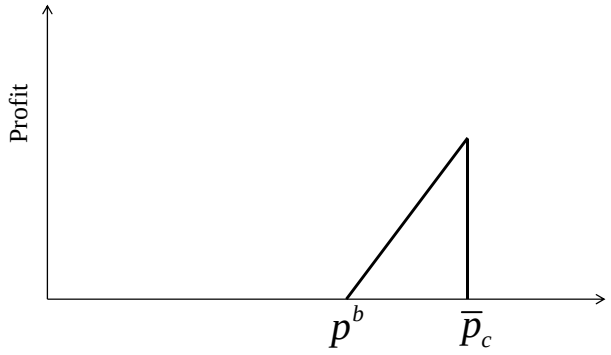


Figure 8.1 Illustration of the value of quoting a price p that is greater than the break even price p^b but lower than the cuttoff price $\bar{p}_c$ for customer c.

of features that bring us outside the scope of the clean, fundamental models presented in Chapters 3 and 5. For example, in the bidding problem, we are not able to observe an unbiased sample of the truth. That is, in the industrial setting, the company does not get to observe the exact amount that the client was willing to pay for the contract, only whether or not the quoted price was accepted. As we will see in this chapter, this creates an additional challenge for our analysis. We are no longer able to make use of an elegant conjugate prior, as in Chapter 2.

New challenges call for new methods. In contrast with our earlier focus on knowledge gradient methods, we look at a simpler greedy policy for the bidding problem. However, this policy is still in line with our general approach to optimal learning. We will use a Bayesian model to represent our uncertainty about customer demand. Our policy incorporates this uncertainty into the decision-making. As a result, we will tend to quote higher prices to the customers than we would without the learning dimension. How much higher will depend on the amount of uncertainty we have. By doing so, we will take on more risk with the first few contracts, but the information we collect will help us to make money in the long run. The bidding problem shows that optimal learning provides value in a slightly messier setting that goes beyond the standard models we have discussed up to this point.

8.1.1 Modeling customer demand

Suppose that we are working with a sequence of bids. The customers' behavior is modeled using the concept of *valuation* or *willingness to pay*. The nth customer values a product or service at W^n. We make the sale as long as this valuation is at least as much as our quoted price p, that is, $W^n \geq p$. Our revenue, in this case, is the price p. If $W^n < p$, our revenue is zero. Thus, our *expected revenue*, for a fixed price p, is given by

$$R(p) = p \cdot P(W^n \geq p). \tag{8.1}$$

In most applications, we will work with the *expected profit*

$$P(p) = (p - c) \cdot P(W^n \geq p), \tag{8.2}$$

where c is the cost of the product to the seller.

In real life, many customers may not have an exact number for the most they are willing to pay. In any case, we will never observe this quantity, only whether it is larger or smaller than p. However, the idea of the valuation gives us a way to think about the bidding problem formally and put a number on the probability that a price will be accepted. Initially, let us assume that the valuations $W^0, W^1, ...$ are independent and identically distributed. Thus, while different customers can have different valuations, all customers come from the same population. This may be a reasonable assumption if a type of industrial contract serves a particular market (such as semiconductor manufacturers). Later, in Section 8.1.3, we begin to push the boundaries of this assumption.

8.1.2 Some valuation models

Nearly any standard distribution might be used to model customer valuation. We list several distributions that have been used in the literature on bidding. We will make use of some of these models to illustrate some of the issues inherent in the problem, although our main focus will be the logistic model of Section 8.1.3. Many of these same distributions also appeared in Chapter 2 as sampling models.

Uniform valuation The simplest model assumes that the valuation follows a uniform distribution, $W^n \sim U[a,b]$. It follows that the probability of making the sale is

$$P(W^n \geq p) = \begin{cases} 1 & p < a \\ \frac{b}{b-a} - \frac{p}{b-a} & p \in [a,b] \\ 0 & p > b \end{cases},$$

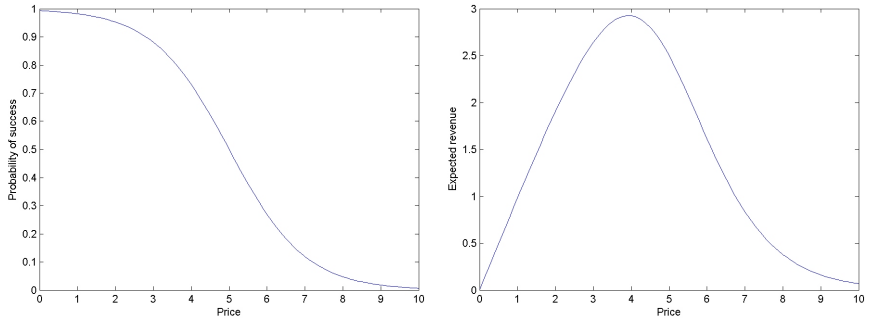
which is a linear function of p. This is known as *linear demand*. If we assume that $a = 0$, the only parameter in this model is the maximum valuation b. If this parameter is unknown, a natural choice for a distribution of belief would be a Pareto prior (see Section 2.6.3).

The uniform valuation is clean, but involves several strong assumptions. We are assuming that there is a fair amount of variation; customers are equally likely to have low or high valuations. Additionally, we assume that there is a cutoff point. A high enough price is guaranteed to lose the sale.

Exponential valuation An exponential valuation assumes that $W^n \sim Exp(\lambda)$. The demand curve then has the simple form $P(W^n \geq p) = e^{-\lambda p}$. As in the linear model, higher prices are less likely to be successful. However, now any price will have a non-zero probability of being successful. We are assuming that there will be a small proportion of customers willing to pay very large prices.

Lognormal valuation In the lognormal model, we assume that

$$P(W^n \geq p) = 1 - \Phi\left(\frac{\log p - \mu}{\sigma}\right),$$



(a) (b)

Figure 8.2 Example of (a) probability of making a sale and (b) expected revenue under a logistic model with $\mu_1 = 5$, $\mu_2 = 1$.

where μ and σ are the parameters of the demand curve. This model implies that $\log W^n \sim \mathcal{N}\left(\mu, \sigma^2\right)$. If we had a way to observe the exact values W^n, we could put a normal prior on μ and treat $\log W^n$ as a normal observation, enabling the use of the normal-normal learning model.

8.1.3 The logit model

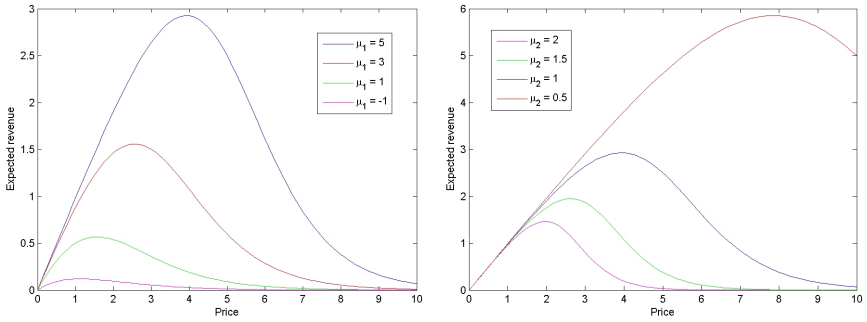
The logit model is a particularly attractive method of approximating the probability that a customer will accept a bid. The logic model expresses the probability of making a sale as

$$P\left(W^n \geq p\right) = \frac{1}{1 + e^{-(\mu_1 - \mu_2 p)}}. \tag{8.3}$$

When $\mu_2 > 0$, plotting (8.3) as a function of p yields a logistic curve, a well-known mathematical model for predicting demand, population growth, technology adoption, and other cyclical phenomena. Figure 8.2(a) gives an example for a particular choice of μ_1 and μ_2. We see a classic S-curve, flipped around so that higher prices lead to lower success probabilities. Balancing the decreasing success probability with the increasing potential revenue, the expected revenue function (8.1) has a maximum at approximately $p^* \approx 3.9$.

The parameters μ_1 and μ_2 determine the customer's reaction to different price offers. We can view μ_1 as the *market share* of the seller. Even if the seller were to give away the product for free, the probability of success would only be $\frac{1}{1+e^{-\mu_1}}$, that is, some customers would still prefer to buy from a competitor. Essentially, these customers have a negative valuation of our product, and there is no way we could convince them to buy it. The higher the value of μ_1, the higher the market share and the closer $P\left(W^n \geq 0\right)$ is to 1.

The second parameter μ_2 can be viewed as the *price sensitivity* of the customers. Large values of μ_2 tend to make the curve in Figure 8.3 steeper, causing the probability of success to decrease faster as the price goes up. We typically require that $\mu_2 > 0$, to preserve the S-curve shape of the success probability. Negative values of μ_2 would



(a) (b)

Figure 8.3 Sensitivity of the revenue curve to changes in (a) market share and (b) price parameter.

imply that higher prices are more likely to be successful, which does not make sense for our problem.

One important advantage of the logistic model is that it allows us to move beyond the assumption that the customers come from the same population. We could, for example, allow the demand curve to depend on attributes of the customer, as given by

$$P\left(W^n \geq p\right) = \frac{1}{1 + e^{-(\mu^T x^n - p\mu_{P+1})}}. \tag{8.4}$$

Here, $x^n = [x_1^n, x_2^n, ..., x_P^n]^T$ is a vector representing P attributes of the customer. For example, the attributes could reflect the location of the customer and its size. The vector μ contains the parameters assigned to the attributes. We also have a single price sensitivity parameter $\mu_{P+1} > 0$. This model is an example of the *logistic regression* technique in statistics, which fits a set of parameters to a success probability. Throughout this chapter, we mostly focus on the simple two-parameter model of (8.3), but it is important to remember that our analysis can easily be extended to the multi-parameter case.

Optimal learning comes into play when we, as the seller, do not know the exact values of the parameters μ_1, μ_2. We may have some prior estimates of these parameters, based on past sales figures. However, what makes this problem especially difficult is that even small changes to the parameters will greatly change the expected revenue function. Figure 8.3(a) shows the expected revenue curve for different values of μ_1, with μ_2 fixed at 1. Figure 8.3(b) shows the same curve for different values of μ_2, with μ_1 fixed at 5.

Higher values of μ_1 move the optimal price to the right, but they also expand the magnitude of the entire revenue curve. Increasing μ_1 from 1 to 3 moves the optimal price roughly from 1.5 to 2.5, but triples the expected revenue collected in the process. Smaller values of μ_2 (that is, closer to zero) have the same two effects, but move the optimal price more than they increase optimal revenue. When we allow both parameters to change at the same time, as in Figure 8.4, even small changes will allow for a wide range of possible optimal prices and revenues.

8.1.4 Bayesian modeling for dynamic pricing

We will use a Bayesian model to represent our uncertainty about the parameters. In this setting, it becomes especially important to construct our prior in such a way as to realistically cover our range of uncertainty about the optimal price. Furthermore, the nature of the observations in this problem creates additional challenges. We are not able to observe the customer valuations W^n. Rather, we only observe whether or not $W^n \geq p$. The clean, conjugate Bayesian models introduced in Chapter 2 cannot be directly applied, and we need to do additional work to be able to use them.

8.1.4.1 A conjugate prior for choosing between two demand curves

Before we delve into the intricacies of non-conjugate Bayesian learning, let us first begin with a simple, stylized model that allows for a conjugate prior. In dynamic pricing, our observations are binary: either the customer accepts the offer (that is, $W^n \geq p$), or not. One way to obtain a conjugate prior is if the truth is binary, as well. Suppose that there are only two possible demand curves. Each curve has known, fixed parameters, but we do not know which curve is the right one for describing customer valuations. Figure 8.5 gives an example where we are trying to choose between a uniform valuation on the interval $[3, 7]$, and an exponential valuation with parameter 0.2.

This model is somewhat stylized, but may be useful in some cases. It may be that we, as the seller, have a large amount of historical data on sales figures, enough to fit any particular type of demand curve. We could conduct a statistical analysis to fit a uniform valuation model, or an exponential model, or perhaps a logistic model. For each model, we would fit a different set of parameters. However, we are not sure which type of demand curve is most appropriate. A logistic model may be fundamentally better-suited to the data than a uniform model. In that case, the binary-truth problem may help us to distinguish between two competing types of demand models.

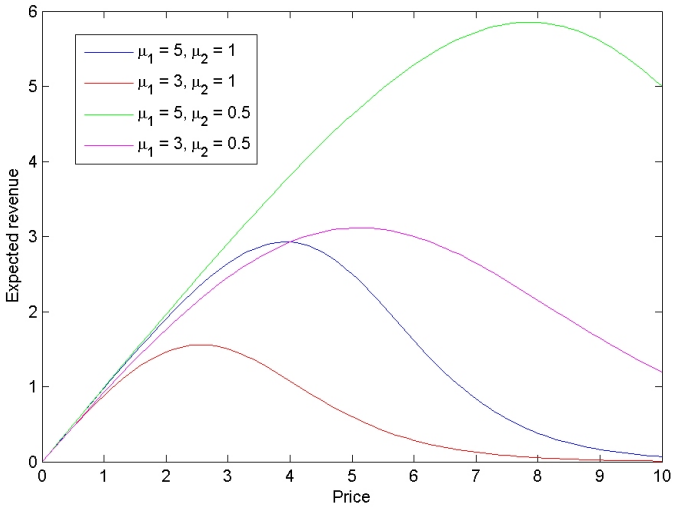


Figure 8.4 Sensitivity of the revenue curve to changes in both parameters.

Let f_1 and f_2 be the two demand curves under consideration. That is, $f_i(p) = P(W^n \geq p)$ under two different models $i = 1, 2$. Let A be the event that f_1 is the correct demand curve; then, A^c is the event that f_2 is correct. We begin with a prior probability $q^0 = P^0(A)$. For a fixed price p, the revenue function is given by

$$R(p) = p\left[q^0 f_1(p) + \left(1 - q^0\right) f_2(p)\right]. \tag{8.5}$$

We then make a pricing decision p^0 and observe either $W^1 \geq p^0$ or $W^1 < p^0$. First, let us consider the case where $W^1 \geq p^0$, that is, we make the sale with price p^0. Using Bayes' rule, we can derive

$$P\left(A \mid W^1 \geq p^0\right) = \frac{P\left(W^1 \geq p^0 \mid A\right) P(A)}{P\left(W^1 \geq p^0 \mid A\right) P(A) + P\left(W^1 \geq p^0 \mid A^c\right) P\left(A^c\right)}. \tag{8.6}$$

Observe that

$$\begin{aligned} P\left(W^1 \geq p^0 \mid A\right) &= f_1(p), \\ P\left(W^1 \geq p^0 \mid A^c\right) &= f_2(p), \\ P(A) &= q^0. \end{aligned}$$

We can let $q^1 = P\left(A \mid W^1 \geq p^0\right)$ denote our posterior probability of the event A. Then, (8.6) becomes

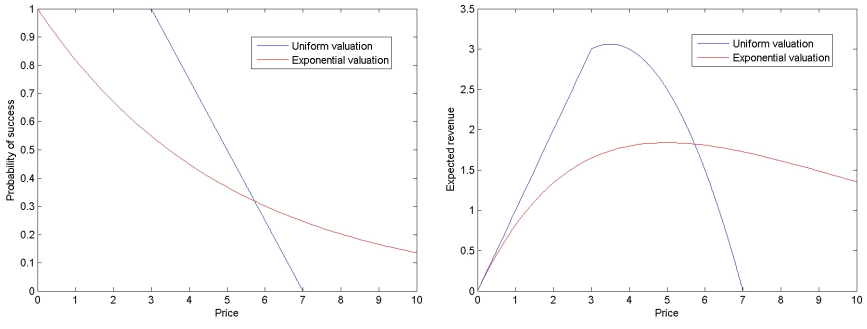
$$q^1 = \frac{q^0 f_1(p)}{q^0 f_1(p) + (1 - q^0) f_2(p)}.$$

Repeating the same analysis for the event that $W^1 < p^0$ produces the updating equation

$$q^1 = \frac{q^0 (1 - f_1(p))}{q^0 (1 - f_1(p)) + (1 - q^0)(1 - f_2(p))}.$$

Let $X^n = I_{\{W^n \geq p^n\}}$. That is, $X^n = 1$ if our price p^n is successful, and $X^n = 0$ otherwise. Then, we obtain a clean updating formula

$$q^{n+1} = \frac{q^n f_1(p^n)^{X^{n+1}} (1 - f_1(p^n))^{1 - X^{n+1}}}{q^n f_1(p^n)^{X^{n+1}} (1 - f_1(p^n))^{1 - X^{n+1}} + (1 - q^n) f_2(p^n)^{X^{n+1}} (1 - f_2(p^n))^{1 - X^{n+1}}}. \tag{8.7}$$



(a) (b)

Figure 8.5 Examples of (a) demand curves and (b) revenue curves for a pricing problem with two truths.

This may be the simplest possible conjugate model. We put a simple discrete (actually binary) distribution of belief on the probability that the truth is given by a particular demand curve. When we make a binary observation, the posterior distribution is also discrete.

We might then apply a variety of techniques to make our pricing decision. A common approach in the literature is to use a simple myopic policy,

$$p^n = \arg\max_p \mathbb{E}R\left(p\right) = \arg\max_p p\left[q^0 f_1\left(p\right) + \left(1-q^0\right) f_2\left(p\right)\right].$$

We could also apply some of the ideas from Section 4.7.1, where the observation was also binary, to this problem. There is one important issue to keep in mind, however. Observe that Figure 8.5(a) has a point where the two demand curves intersect. That is, there is a price $\bar{p}$ for which $f_1\left(\bar{p}\right) = f_2\left(\bar{p}\right) = \bar{f}$. Substituting this price into (8.7) gives us

$$\begin{aligned} q^{n+1} &= \frac{q^n \bar{f}^{X^{n+1}}\left(1-\bar{f}\right)^{1-X^{n+1}}}{q^n \bar{f}^{X^{n+1}}\left(1-\bar{f}\right)^{1-X^{n+1}} + \left(1-q^n\right)\bar{f}^{X^{n+1}}\left(1-\bar{f}\right)^{1-X^{n+1}}} \\ &= \frac{q^n \bar{f}^{X^{n+1}}\left(1-\bar{f}\right)^{1-X^{n+1}}}{\bar{f}^{X^{n+1}}\left(1-\bar{f}\right)^{1-X^{n+1}}} \\ &= q^n. \end{aligned}$$

If our policy chooses the price $\bar{p}$, our beliefs about $P\left(A\right)$ will remain unchanged. Furthermore, since the policy determines what to do based on our beliefs, it will continue to choose the same price $\bar{p}$ thereafter. We would thus get stuck with the same beliefs forever. For that reason, the price $\bar{p}$ is known as the *non-informative price* or the *confounding price*.

Fortunately, since we have both demand curves specified exactly (we just don't know which is the right one), we can calculate the confounding price before we set out to solve the problem. The literature suggests a simple fix for the problem of confounding. We simply fix another price $\tilde{p} \neq \bar{p}$ beforehand. If our policy tells us to choose the price $\bar{p}$, we choose $\tilde{p}$ instead. Otherwise, we follow the policy.

8.1.4.2 Moment matching for non-conjugate problems* Although the simple model of Section 8.1.4.1 has its uses, we are really interested in the case where we do not know the parameters of the demand curve. In any case, before we settle on two demand curves to choose from, we first need to find good parameters for those curves. So, we might focus on one particular type of curve, and then try to learn the right parameters for that type. This will lead us to the problem of non-conjugacy.

Let us illustrate this issue using a simple example. Suppose that the customers' valuations are drawn from a uniform distribution on the interval $[0, B]$, where B is unknown. Recalling Chapter 2, a natural choice of prior distribution on B is the Pareto distribution. If $B \sim Pareto\left(\alpha^0, b^0\right)$, and if we were able to observe the exact valuation $W^1 \sim U\left[0, B\right]$, we could apply the conjugate update

$$\begin{aligned} \alpha^1 &= \alpha^0 + 1 \\ b^1 &= \max\left(b^0, W^1\right). \end{aligned}$$

Unfortunately, we are never able to see the exact valuation. However, we can still derive a posterior distribution of belief, given the incomplete information that we do observe. For simplicity, let us assume that we set a price $p^0 < b^0$. We will first consider the case where $W^1 < p^0$, that is, we lost the sale. We apply Bayes' rule and write

$$g\left(u \,|\, W^1 < p^0\right) = \frac{P\left(W^1 < p^0 \,|\, B = u\right) g\left(u\right)}{P\left(W^1 < p^0\right)}. \tag{8.8}$$

Conditionally given $B = u$, the valuation W^1 has a uniform distribution on $[0, u]$. Thus,

$$P\left(W^1 < p^0 \,|\, B = u\right) = \begin{cases} \frac{p^0}{u} & p^0 < u \\ 1 & p^0 \geq u. \end{cases}$$

Next, the likelihood that $B = u$ is given by the Pareto density,

$$g\left(u\right) = \frac{\alpha^0 \left(b^0\right)^{\alpha^0}}{u^{\alpha^0+1}} \qquad \text{for } u > b^0.$$

Thus, in our calculations, we are implicitly assuming that $u > b^0$, since $B > b^0$ by the definition of a Pareto distribution. Since we are also assuming $p^0 < b^0$, it follows that $p^0 < u$ for all possible u. The numerator of the right-hand side of (8.8) thus becomes

$$P\left(W^1 < p^0 \,|\, B = u\right) g\left(u\right) = \frac{\alpha^0 p^0 \left(b^0\right)^{\alpha^0}}{u^{\alpha^0+2}}. \tag{8.9}$$

Integrating this expression from b^0 to infinity, we obtain the denominator of the right-hand side of (8.8),

$$P\left(W^1 < p^0\right) = \int_{b^0}^{\infty} \frac{\alpha^0 p^0 \left(b^0\right)^{\alpha^0}}{u^{\alpha^0+2}} du = \frac{\alpha^0}{\alpha^0+1} \frac{p^0}{b^0}. \tag{8.10}$$

Dividing (8.9) by (8.10) yields

$$g\left(u \,|\, W^1 < p^0\right) = \frac{\left(\alpha^0+1\right)\left(b^0\right)^{\alpha^0+1}}{u^{\alpha^0+2}}.$$

This is a Pareto density with parameters $\alpha^1 = \alpha^0 + 1$ and $b^1 = b^0$. That is, if we lose the sale, conjugacy is maintained. The conjugate updating equations hint at this. From (8.8), we see that $b^1 = \max\left(b^0, W^1\right)$, if we could observe the exact valuation W^1. However, we are assuming that $p^0 < b^0$. Thus, if we observe $W^1 < p^0$, it follows automatically that $\max\left(b^0, W^1\right) = b^0$. We do not really need to know the exact value of W^1 in this case. For any value of W^1 less than b^0, we will not change the scale parameter of our distribution of belief.

In the other case, however, we run into trouble. Suppose that $W^1 \geq p^0$, with $p^0 < b^0$ as before. Then,

$$P\left(W^1 \geq p^0 \,|\, B = u\right) = \begin{cases} 1 - \frac{p^0}{u} & p^0 < u \\ 0 & p^0 \geq u \end{cases}$$

and

$$P\left(W^1 \geq p^0 \,|\, B = u\right) g\left(u\right) = \frac{\alpha^0 \left(b^0\right)^{\alpha^0}}{u^{\alpha^0+1}} - \frac{\alpha^0 p^0 \left(b^0\right)^{\alpha^0}}{u^{\alpha^0+2}} \qquad u > b^0.$$

Integrating this quantity from b^0 to infinity yields

$$P\left(W^1 \geq p^0\right) = 1 - \frac{\alpha^0}{\alpha^0+1}\frac{p^0}{b^0},$$

and the posterior density turns out to be

$$g\left(u \,|\, W^1 < p^0\right) = \frac{\frac{\alpha^0\left(b^0\right)^{\alpha^0}}{u^{\alpha^0+1}} - \frac{\alpha^0 p^0\left(b^0\right)^{\alpha^0}}{u^{\alpha^0+2}}}{1 - \frac{\alpha^0}{\alpha^0+1}\frac{p^0}{b^0}} \qquad u > b^0. \tag{8.11}$$

What are we to do with this strange expression? It does not correspond to any standard density. (The difference of two Pareto densities is *not* the density of the difference of two Pareto random variables!) It is certainly not a Pareto density.

We get around this problem by using a technique called *moment-matching*. Define a random variable U that has the density given by (8.11). As long as we have the density, we can compute the first two moments of U as follows. First,

$$\begin{aligned}
\mathbb{E}U &= \int_{b^0}^{\infty} u \frac{\frac{\alpha^0\left(b^0\right)^{\alpha^0}}{u^{\alpha^0+1}} - \frac{\alpha^0 p^0\left(b^0\right)^{\alpha^0}}{u^{\alpha^0+2}}}{1 - \frac{\alpha^0}{\alpha^0+1}\frac{p^0}{b^0}} \\
&= \frac{\frac{\alpha^0 b^0}{\alpha^0-1} - p^0}{1 - \frac{\alpha^0}{\alpha^0+1}\frac{p^0}{b^0}}.
\end{aligned} \tag{8.12}$$

Similarly,

$$\begin{aligned}
\mathbb{E}\left(U^2\right) &= \int_{b^0}^{\infty} u^2 \frac{\frac{\alpha^0\left(b^0\right)^{\alpha^0}}{u^{\alpha^0+1}} - \frac{\alpha^0 p^0\left(b^0\right)^{\alpha^0}}{u^{\alpha^0+2}}}{1 - \frac{\alpha^0}{\alpha^0+1}\frac{p^0}{b^0}} \\
&= \frac{\frac{\alpha^0\left(b^0\right)^2}{\alpha^0-2} - \frac{\alpha^0 b^0 p^0}{\alpha^0-1}}{1 - \frac{\alpha^0}{\alpha^0+1}\frac{p^0}{b^0}}.
\end{aligned} \tag{8.13}$$

Let Y be a random variable following a Pareto distribution with parameters α^1 and b^1. The first and second moments of Y are given by

$$\mathbb{E}Y = \frac{\alpha^1 b^1}{\alpha^1 - 1}, \qquad \mathbb{E}\left(Y^2\right) = \frac{\alpha^1\left(b^1\right)^2}{\alpha^1 - 2}.$$

Moment matching works by setting the moments of Y equal to the moments of U, and solving for α^1 and b^1 in terms of α^0 and b^0. Essentially, we are forcing the posterior distribution to be Pareto, and choosing α^1 and b^1 to make this Pareto distribution resemble the actual posterior distribution, at least up to the first two moments. To do this, we have to solve two sets of nonlinear equations

$$\begin{aligned}
\frac{\alpha^1 b^1}{\alpha^1 - 1} &= \frac{\frac{\alpha^0 b^0}{\alpha^0-1} - p^0}{1 - \frac{\alpha^0}{\alpha^0+1}\frac{p^0}{b^0}}, \\
\frac{\alpha^1\left(b^1\right)^2}{\alpha^1 - 2} &= \frac{\frac{\alpha^0\left(b^0\right)^2}{\alpha^0-2} - \frac{\alpha^0 b^0 p^0}{\alpha^0-1}}{1 - \frac{\alpha^0}{\alpha^0+1}\frac{p^0}{b^0}}.
\end{aligned}$$

The solution gives us the non-conjugate updating equations,

$$\begin{aligned}\alpha^1 &= 1+\sqrt{\frac{\mathbb{E}\left(U^2\right)}{\mathbb{E}\left(U^2\right)-\left(\mathbb{E}U\right)^2}},\\ b^1 &= \frac{\alpha^1-2}{\alpha^1-1}\frac{\mathbb{E}\left(U^2\right)}{\mathbb{E}U},\end{aligned}$$

where $\mathbb{E}U$ and $\mathbb{E}\left(U^2\right)$ are given in (8.12) and (8.13) in terms of α^0 and b^0. Notice that, for compactness, the updating equation for b^1 is written in terms of α^1. To use these equations, we should first compute α^1, then use that value to find b^1.

Keep in mind that we have not computed updating equations for the case where $p^0 \geq b^0$. In this case, neither success nor failure will give us a Pareto posterior, and we have to repeat the above analysis. Choosing a different demand curve and prior distribution would also require a new round of moment-matching, and most likely new sets of nonlinear equations to solve. Moment-matching is no easy task. However, it is the simplest way to obtain clean updating equations in a problem where the observations are incomplete, that is, we can only obtain imperfect information about the customer valuations.

8.1.5 An approximation for the logit model

The issue of non-conjugacy is also present in the logit model. There is no natural choice of prior for the logistic regression parameters μ_1, μ_2 in (8.3). That is, there is no clean prior distribution that is conjugate with observations sampled from the logistic distribution. The simplest choice of prior, from the decision-maker's point of view, is a multivariate normal distribution on (μ_1, μ_2). Especially if we extend our model to incorporate customer attributes, as in (8.4), a multivariate normal prior would allow us to include correlations in our beliefs about the parameters of different attributes. We followed this same approach in Chapter 7 when we fit a linear regression model to our observations. As we saw earlier, the linear regression model admits an elegant conjugate normal-normal learning model.

Unfortunately, there is no direct analog in the case of logistic regression, because our observations are now binary, of the form

$$X^n = \begin{cases} 1 & \text{If } W^n \geq p^n, \\ 0 & \text{Otherwise.} \end{cases}$$

Recall that $X^n = 1$ if we make the sale, and $X^n = 0$ if we do not. If we start with a normal prior on the logistic parameters, then see an observation of this form, the posterior distribution will not be normal. However, there is an approximation that gives us a set of recursive updating equations. Like we did in (8.1.4.2) with moment matching, we can use this approximation to force the posterior to be normal. If $\mu = (\mu_1, \mu_2)$ has a multivariate normal distribution with mean vector $\bar{\theta}^n = \left(\bar{\theta}_1^n, \bar{\theta}_2^n\right)$

and covariance matrix Σ^n, these approximate recursive updating equations are

$$\Sigma^{n+1} = \left((\Sigma^n)^{-1} + 2\lambda(\xi^n)(x^n)(x^n)^T\right)^{-1}, \tag{8.14}$$

$$\bar{\theta}^{n+1} = \Sigma^{n+1}\left((\Sigma^n)^{-1}\bar{\theta}^n + \left(X^{n+1} - \frac{1}{2}\right)x\right). \tag{8.15}$$

The vector $x^n = (1, -p^n)^T$ corresponds to the explanatory variables in (8.4). The function λ is given by

$$\lambda(\xi) = \frac{\tanh(\xi/2)}{4\xi}.$$

The value ξ^n is an artificial parameter used in the approximation of the experimental precision. This parameter is also updated recursively, using the equation

$$\xi^{n+1} = \sqrt{(x^n)^T \Sigma^{n+1} x^n + (x^n)^T \bar{\theta}^{n+1}}.$$

We can apply the same technique we used for correlated normal beliefs back in Chapter 2 to get a cleaner form for the updating equations

$$\bar{\theta}^{n+1} = \bar{\theta}^n + \frac{\frac{X^{n+1}-\frac{1}{2}}{2\lambda(\xi^n)} - (x^n)^T\bar{\theta}^n}{\frac{1}{2\lambda(\xi^n)} + (x^n)^T\Sigma^n x^n}\Sigma^n x^n, \tag{8.16}$$

$$\Sigma^{n+1} = \Sigma^n - \frac{\Sigma^n x^n (x^n)^T \Sigma^n}{\frac{1}{2\lambda(\xi^n)} + (x^n)^T\Sigma^n x^n}. \tag{8.17}$$

These equations closely resemble the recursive updating rules we used for linear models in Section 7.2.2. Writing the update in this way gives us some insight into the way the approximation works. The quantity $\frac{1}{2\lambda(\xi)}$ is used in two ways. First, in the denominator of the fractional terms in (8.16) and (8.17), it serves as a stand-in for the variance of the observation. In the correlated normal model considered in Section 2.3.2, the same role is played by the experimental noise. Second, in the numerator of the fractional term in (8.16), the same quantity is used to convert the binary observation $X^{n+1} - \frac{1}{2}$ into a continuous quantity. We can rewrite (8.3) as

$$\mu_1 - \mu_2 p^n = \log\left(\frac{P(W^{n+1} \geq p^n)}{1 - P(W^{n+1} \geq p^n)}\right). \tag{8.18}$$

The right-hand side of (8.18) is called the *log-odds* of making a sale. The quantity $(x^n)^T\bar{\theta}^n$ can be viewed as our prediction of the log-odds. Thus, the continuous quantity $\frac{X^{n+1}-\frac{1}{2}}{2\lambda(\xi^n)}$ can be interpreted as an approximate observation of the log-odds.

We are forcing the problem into the framework of Chapter 7, where our regression model is used to predict continuous observations. To do this, we are artificially creating a set of continuous responses from our binary observations. If $X^{n+1} = 1$, that is, we make the sale, the continuous response is positive, and we conclude that the log-odds are more likely to be greater than zero, and adjust our prior if it is under-estimating them. If we lose the sale and $X^{n+1} = 0$, the continuous response is negative, leading us to believe that the log-odds are more likely to be negative.

One advantage of this approach is that it can easily handle customer attributes, as in (8.4). We simply place a multivariate prior on the vector $\mu = (\mu_1, ..., \mu_{P+1})$ on the parameters of the customer attributes, as well as on the price sensitivity. We then make a pricing decision p^n and apply (8.16) and (8.17) using $x^n = (x_1^n, ..., x_P^n, -p^n)$.

It is important to understand that our posterior distributions in this model are not really normal, just as our posterior distributions in Section 8.1.4.2 were not really Pareto. The updating equations in (8.16) and (8.17) can only serve as an approximation of the way we learn in this problem. The approximation may not always be accurate. We do not necessarily need to use this approach. For example, we can simply collect our observations $X^1, X^2, ..., X^{n+1}$ and fit a logistic regression model to obtain estimates of μ_1 and μ_2. This approach may give us better fits for some particular values of μ, but it is frequentist in nature. It does not incorporate any idea of our prior uncertainty about the parameters, as represented by the prior covariance matrix Σ. Thus, on average across many different truths, we may do better with the Bayesian approximation.

There is no perfect model for learning in this setting, making it difficult to create a sophisticated learning policy. For example, if we try to construct a look-ahead policy such as knowledge gradient, we are forced to rely on an approximation for the predictive distribution of the future beliefs, and another approximation for the optimal implementation decision under those future beliefs. Even so, we can still improve our decision-making by considering some concepts of optimal learning, such as the idea of our uncertainty about the problem parameters.

8.1.6 Bidding strategies

Recall that the revenue function for the logit demand model, under the parameters μ_1 and μ_2, is given by

$$R(p; \mu_1, \mu_2) = \frac{p}{1 + e^{-(\mu_1 - \mu_2 p)}}.$$

Suppose that we have some estimates $\bar{\theta}_1^n$ and $\bar{\theta}_2^n$ of the logistic parameters in (8.3). These estimates may be obtained using the approximate Bayesian model from Section 8.1.5, or they may come from a frequentist statistical procedure. Either way, once we have these estimates, a simple and intuitive course of action would be to simply assume that these are the true values, and make a pricing decision by solving

$$p^n = \arg\max_p R\left(p; \bar{\theta}_1^n, \bar{\theta}_2^n\right) = \arg\max_p \frac{p}{1 + e^{-\left(\bar{\theta}_1^n - \bar{\theta}_2^n p\right)}}. \tag{8.19}$$

For simplicity, let us assume that the set of possible prices is finite. Then, it is very easy to compute (8.19). We need only plug our estimated values into the revenue function and solve. This is known as a *point-estimate* policy, or a *certainty-equivalent* policy. We are making the decision that would be optimal if the true values were exactly equal to our estimates.

8.1.6.1 An idea from multi-armed bandits The main insight of optimal learning, however, is that the true values are *not* exactly equal to our estimates. If

we assume that they are, we may under-perform. Consider a very simple bandit problem with a single arm. The one-period reward of the arm follows an exponential distribution with unknown parameter λ. We use a gamma-exponential learning model from Section 2.6.1, and assume that $\lambda \sim Gamma\,(a, b)$.

The average one-period reward is $\frac{1}{\lambda}$, a very simple function of λ. Our estimate of λ is $\mathbb{E}\lambda = \frac{a}{b}$. If we assume that the true value is exactly equal to our estimate, then our resulting estimate of the average one-period reward becomes $\frac{1}{\mathbb{E}\lambda} = \frac{b}{a}$. This reasoning seems straightforward, but it ignores the uncertainty in our beliefs about λ.

Instead of merely plugging in our estimate of λ into the average reward, let us view that reward as a function of a random variable λ. We can then take an expected value of that reward over our distribution of belief, arriving at

$$\mathbb{E}\left(\frac{1}{\lambda}\right) = \frac{b}{a-1}, \tag{8.20}$$

a different estimate of the average reward. This approach accounts for the fact that λ is unknown. Not only do we have an estimate of this parameter, we also have some amount of uncertainty about that estimate. That uncertainty is encoded in the gamma distribution that we use to represent our beliefs.

Suppose now that we have M independent arms. The one-period reward obtained by playing arm x is exponential with parameter λ_x, and we assume that $\lambda_x \sim Gamma\,(a_x, b_x)$. Suppose also that our goal is simply to use a greedy, pure-exploitation strategy for pulling arms. On average over many truth values, the policy that plays $\arg\max_x \frac{b_x}{a_x - 1}$ will do better than the policy that plays $\arg\max_x \frac{b_x}{a_x}$. We do better by incorporating the uncertainty in our beliefs into our decision-making.

8.1.6.2 Bayes-greedy bidding

We apply the same idea to the bidding problem. Rather than using (8.19) to make decisions, we take an expectation of the revenue function over our distribution of belief,

$$p^n = \arg\max_p \mathbb{E}R\,(p; \mu_1, \mu_2) = \arg\max_p \mathbb{E}\frac{p}{1 + e^{-(\mu_1 - \mu_2 p)}}. \tag{8.21}$$

Of course, this requires us to have a distribution of belief in the first place. Thus, this policy only really makes sense if we use the Bayesian model from Section 8.1.5 to learn about the unknown parameters. Under this model, we can assume that, at time n, $\mu \sim \mathcal{N}\left(\bar{\theta}^n, \Sigma^n\right)$. After we make a decision p^n and observe X^{n+1}, we use (8.16) and (8.17) to change our beliefs. We refer to this policy as a *Bayes-greedy* policy, to distinguish it from the point-estimate greedy policy.

A well-known property of multivariate normal distributions is that a linear function of a multivariate normal vector is also normal. That is, if $\mu \sim \mathcal{N}\,(\theta, \Sigma)$ and c is a vector, then $c^T x \sim \mathcal{N}\left(c^T\theta, c^T\Sigma c\right)$. In our case, $c = (1, -p)^T$ and

$$\begin{aligned} c^T\theta &= \theta_1 - \theta_2 p, \\ c^T\Sigma c &= \Sigma_{11} - 2p\Sigma_{12} + p^2\Sigma_{22}. \end{aligned}$$

Here we are using the fact that $\Sigma_{12} = \Sigma_{21}$ due to the symmetry of the covariance matrix. Consequently, we can rewrite (8.21) as

$$p^n = \arg\max_p p \cdot \mathbb{E}\left(\frac{1}{1 + e^{-Y}}\right), \tag{8.22}$$

where $Y \sim \mathcal{N}\left(\theta_1 - \theta_2 p, \Sigma_{11} - 2p\Sigma_{12} + p^2\Sigma_{22}\right)$. Note that Y is a one-dimensional normal random variable. The right-hand side of (8.22) now seems straightforward: we need to compute an expectation over a normal density. We can view $\mathbb{E}\left(\frac{1}{1+e^{-Y}}\right)$ as the Bayesian probability of success.

Unfortunately, this expectation is impossible to compute analytically by integration. We have to resort to yet another approximation. One approach is to generate a large number of Monte Carlo samples of Y from a normal distribution with the appropriate mean and variance. We can then take a sample average

$$\mathbb{E}\left(\frac{1}{1+e^{-Y}}\right) \approx \frac{1}{K}\sum_{k=1}^{K}\frac{1}{1+e^{-Y(\omega_k)}}, \tag{8.23}$$

where $Y(\omega_k)$ is the kth sample realization. Figure 8.6(a) plots the right-hand side of (8.23), as a function of the mean and standard deviation of Y. For very large values of K, we will get more accurate estimates. However, generating enough samples may be fairly expensive computationally.

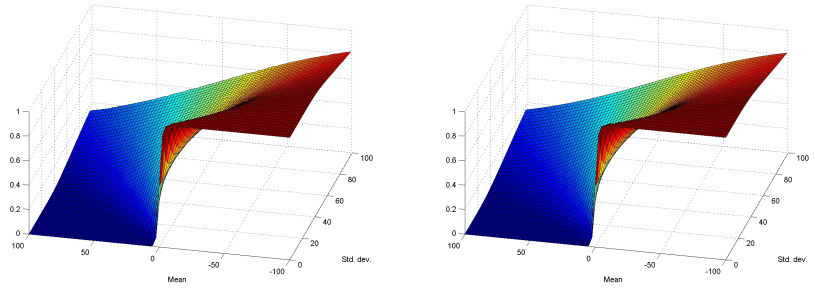
From this figure, we can make an interesting observation. If we fix a value of the standard deviation, and view the left-hand side of (8.23) as a function of the mean of Y, this function also appears to be a logistic curve. In fact, this is borne out by observing that, if $Var(Y) = 0$, then

$$\mathbb{E}\left(\frac{1}{1+e^{-Y}}\right) = \frac{1}{1+e^{-\mathbb{E}Y}},$$

which is itself a logistic function in $\mathbb{E}Y$. If we fix a larger value of the standard deviation, the expectation continues to look like a logistic function, only with a gentler slope.

This insight has led to a clever closed-form approximation for the difficult expectation. We can write

$$\mathbb{E}\left(\frac{1}{1+e^{-Y}}\right) \approx \frac{1}{1+e^{-\frac{\mathbb{E}Y}{\gamma}}}, \tag{8.24}$$



(a) (b)

Figure 8.6 Bayesian success probability, as a function of the mean and standard deviation of Y using (a) Monte Carlo simulation, and (b) closed-form approximation.

where

$$\gamma = \sqrt{1 + \frac{\pi^2}{8} Var\left(Y\right)}.$$

Figure 8.6(b) plots this approximation for different values of $\mathbb{E}Y$ and $Var\left(Y\right)$. The result does not exactly match the values we got from Monte Carlo sampling (there are errors on the order of 0.01), but one can easily see that the two surfaces are quite close.

Converting this result back into the language of our bidding problem, our Bayes-greedy policy makes pricing decisions according to the rule

$$p^n = \arg\max_p \frac{p}{1 + e^{-\frac{\bar{\theta}_1^n - \bar{\theta}_2^n p}{\gamma^n(p)}}}, \tag{8.25}$$

where

$$\gamma^n\left(p\right) = \sqrt{1 + \frac{\pi^2}{8}\left(\Sigma_{11}^n - 2p\Sigma_{12}^n + p^2\Sigma_{22}^n\right)}.$$

Interestingly, the calculation in (8.25) is very similar to what we use for the point-estimate policy. The only difference is that we divide the point estimate $\bar{\theta}_1^n - \bar{\theta}_2^n p$ by an additional factor $\gamma^n\left(p\right)$ that depends on the uncertainty in our beliefs (the covariance matrix), as well as the price decision p.

8.1.7 Numerical illustrations

We will examine the performance of Bayes-greedy pricing in an example problem. Suppose that we are running an online bookstore and selling copies of a certain textbook (perhaps this one!) over the Internet. Suppose, furthermore, that the cost of buying the textbook from the publisher wholesale is $40 per copy. We are thus interested in maximizing the profit function (8.2) with $c = 40$. We will never set a price below $40, and it is also highly unlikely that anyone will buy the book for more than, say, $110.

Figure 8.7 shows a realistic starting prior ($\bar{\theta}_1^0 = 15$, $\bar{\theta}_2^0 = 0.2$) for a logistic demand curve in this setting. Notice that the prior logistic curve is a bit narrower than the

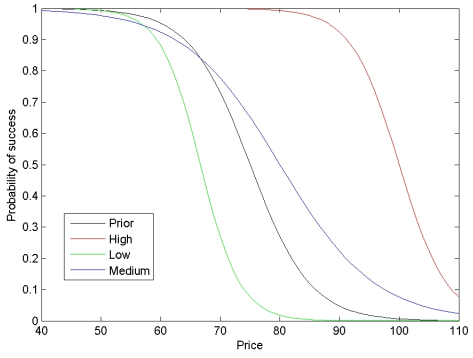
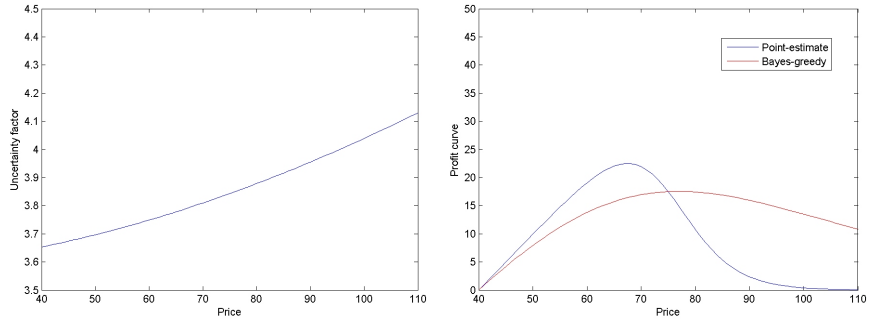


Figure 8.7 Comparison of our starting prior distribution with several possible truth values in a logistic valuation model.



(a) (b)

Figure 8.8 Graphs of (a) the uncertainty factor $\gamma\,(p)$ and (b) the modified profit curve as a function of price.

proposed range of \$40–\$110. This prior seems to be saying that we believe customer valuations to be contained roughly in the range $[50, 100]$. However, we also create a prior covariance matrix

$$\Sigma^0 = \begin{bmatrix} 30 & 0 \\ 0 & 0.03^2 \end{bmatrix} \tag{8.26}$$

which allows for some uncertainty in this belief. Note the difference in magnitude between the variances on the market share and price sensitivity. Figure 8.7 also displays three possible true demand curves, representing scenarios where customer valuations are higher, lower, or about the same relative to the prior. The parameters of these truths are as follows:

High truth:	$\mu_1 = 25, \mu_2 = 0.25$
Medium truth:	$\mu_1 = 10, \mu_2 = 0.125$
Low truth:	$\mu_1 = 20, \mu_2 = 0.3$

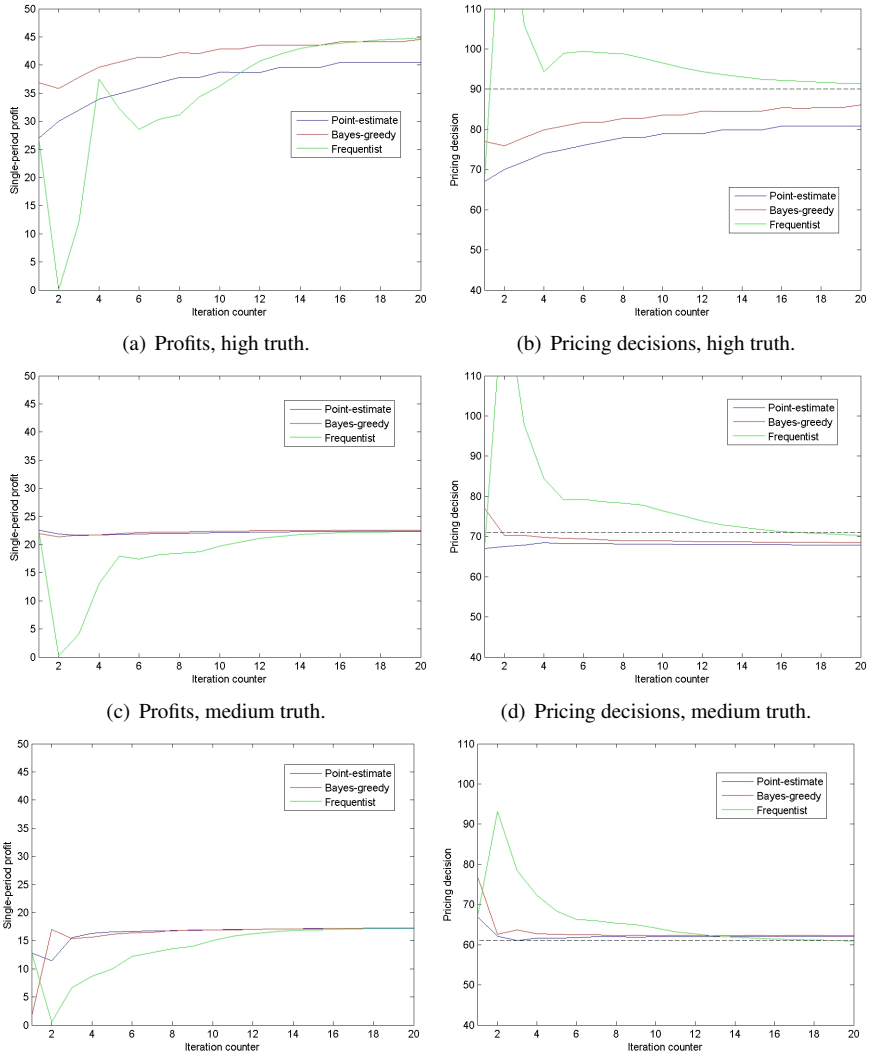
The variation in the price sensitivity for these three scenarios is much smaller than the variation in the market share. This is reflected in our choice of Σ^0. By starting with a prior curve roughly in the middle of our price range, then placing some uncertainty on the parameters, we are able to allow for a wide range of true demand curves.

Figure 8.8(a) shows the uncertainty factor $\gamma^0\,(p)$ as a function of the pricing decision p, for the first time step of this problem. We see that $\gamma^0\,(p)$ increases with price. We can think of this as expressing the risk involved in choosing higher prices: the penalty for losing the sale is greater, but so is the potential reward if the truth is higher than we think. Figure 8.8(b) compares the estimated profit curves

$$P^{PE}\left(p; \bar{\theta}_1^0, \bar{\theta}_2^0\right) = \frac{p-c}{1+e^{-\left(\bar{\theta}_1^0 - \bar{\theta}_2^0 p\right)}}, \tag{8.27}$$

$$P^{BG}\left(p; \bar{\theta}_1^0, \bar{\theta}_2^0\right) = \frac{p-c}{1+e^{-\frac{\bar{\theta}_1^0 - \bar{\theta}_2^0 p}{\gamma^0(p)}}}, \tag{8.28}$$

based on the point estimate and the Bayesian distribution, respectively. These are the functions maximized by the point-estimate and Bayes-greedy policies. We see that the Bayes-greedy curve is wider, representing higher uncertainty or variation in the



(a) Profits, high truth.

(b) Pricing decisions, high truth.

(c) Profits, medium truth.

(d) Pricing decisions, medium truth.

(e) Profits, low truth.

(f) Pricing decisions, low truth

Figure 8.9 Average profits and pricing decisions for different policies across 20 iterations.

profit, and the maximum is shifted right. In other words, the Bayes-greedy policy places more value on exploratory behavior (setting higher prices), and makes more aggressive pricing decisions than the point-estimate policy.

It remains to show how this aggressive pricing impacts performance. Figure 8.9 compares the performance (profit, averaged over 1000 sample paths) and pricing decisions of the point-estimate and Bayes-greedy policies in the first 20 iterations, on each of the three truths graphed in Figure 8.7. We also compare these policies to a frequentist logistic regression technique, a standard approach for fitting a logistic distribution to binary observations. We have focused on Bayesian models and algorithms in most earlier chapters. In this setting, however, the Bayesian modeling

assumptions do not hold, and our updating equations are approximations. Thus, it is relevant to compare to a frequentist technique in order to see whether the inaccuracy of the Bayesian model has any negative impact on performance. We deliberately use the same set of axes for each comparison, to get a better sense of the magnitude of the difference between policies.

The graphs reveal several interesting behaviors. First, the Bayes-greedy policy consistently prices more aggressively than the point-estimate policy, particularly in the very early iterations. If these aggressive decisions are unsuccessful (for example, if the truth is lower than we believe), the policy quickly corrects its behavior and chooses lower prices. If the optimal price (shown as a dashed line) is higher than expected, both point-estimate and Bayes-greedy gradually ramp up their pricing decisions. However, because Bayes-greedy starts out more aggressively, it is able to make a bigger profit more quickly. Thus, the Bayes-greedy policy exhibits a kind of robustness. If the truth is higher than expected, Bayes-greedy adds considerable value (roughly an extra $10 profit per iteration in Figure 8.9(a)), and if the truth is lower than expected, Bayes-greedy tends to lose the first sale, but adjusts almost immediately and obtains comparable profits to the point-estimate policy in subsequent iterations.

The frequentist method is able to fit a good model eventually. After around 20 iterations, it comes closer to the optimal pricing decision than the two policies using the approximate Bayesian model. At the same time, it does not use any prior information, and so it requires some time in order to fit a good model. In the first ten iterations, it exhibits volatile behavior and consistently chooses prices that are unreasonably high, even in the case when the truth itself is higher than we think. It would seem that Bayes-greedy is able to achieve robust performance in the critical early stages of the problem. In some applications, such as the setting of pricing industrial contracts, ten or twenty iterations may be all we ever get.

8.2 A SAMPLED REPRESENTATION

A powerful approach for handling nonlinear models is to use a sampled representation of the different possible values of the parameter vector θ. When we worked with lookup table representations or linear parametric models, we treated the unknown parameters as following a multivariate normal distribution. While this is a fairly complex representation, the linear structure made it possible to derive simple and elegant updating formulas.

With nonlinear models, it is much harder to develop recursive equations if we were to characterize the unknown parameter vector θ as multivariate normal. We might also note that the multivariate normal distribution can cause difficulties since it allows for virtually every possible value for θ, which can produce unexpected (and inappropriate) behaviors. For example, we might use a quadratic function to approximate a concave surface, but if a sampled value of the coefficient of the quadratic term comes out positive, the curve flips around to a convex surface.

8.2.1 The belief model

We start by assuming that we have a model $f(x|W,\theta)$ that has a known structure, characterized by unknown parameters θ and an exogenous source of experimental noise W, that combines to produce a noisy estimate

$$\hat{y} = f(x|W,\theta),$$

where both W and θ are random variables, although they behave differently. The random variable W changes each time we observe our process. The parameter θ, however, is fixed, but it is fixed at a value that we do not know.

We might think of W as an additive error, as in

$$\hat{y} = f(x|\theta) + W.$$

However, W in $f(x|W,\theta)$ can represent uncertain parameters in a nonlinear function.

We assume that $f(x|W,\theta)$ is nonlinear in θ, which is going to cause us some problems when we need to take expectations as we do in the knowledge gradient. There is a simple way to handle a nonlinear model, which is to assume that θ can only take on one of a sampled set $\hat{\Omega} = \{\theta_1, \ldots, \theta_K\}$. We let our belief state about θ be the set of probabilities $S^n = (p_k^n)_{k=1}^K$ where

$$p_k^n = Prob[\theta = \theta_k | H^n].$$

The variable H^n is called the history of the process, which is given by

$$H^n = (S^0, W^1, \ldots, W^n)$$

A natural way to initialize the probabilities is to start with a uniform prior where

$$p_k^0 = \frac{1}{|\hat{\Omega}|},$$

for all k. For example, if we have 20 samples in $\hat{\Omega}$, we would set $p_k^0 = .05$ for each $k = 1, \ldots, 20$. We then let the results of experiments quickly update these probabilities, giving us our belief state

$$S^n = (p_1^n, \ldots, p_K^n).$$

Some thought and care has to be put into how the sample $\theta_1, \ldots, \theta_K$ is created so that we create a reasonable approximation of the distribution of different models that might arise. There are three strategies we might use to generate a sample $\theta_1, \ldots, \theta_k$:

Discretized sample If θ has one to three dimensions, we can simply discretize each dimension and create a discretized version of the parameter space.

Random sampling We could generate each element of θ from a pre-specified range. The problem is that we generally will not know the natural range for each element of θ, especially in the context of a nonlinear, parametric model.

Fitting with prior data Assume we have a set of inputs $x^0, x^1, \ldots, x^n$ with corresponding responses $y^1, y^2, \ldots, y^{n+1}$. We can take a series of subsamples (say, half of the dataset), and then fit a model to this data, producing an estimate θ^k. Repeating this K times provides an initial set of possible values of θ, which automatically captures the correlations between the elements. We suggest using this information to create a multivariate normal distribution, which allows us to scale the covariance matrix if we feel there is not enough variability in the sample.

Fitting with synthetic data It will generally be easier to specify the range over which each dimension of x might vary, as well as the response y, since these have well defined units derived from the contextual setting. We can specify a range for each dimension of x, and then generate random samples (treating each element randomly). We then have to devise a way to generate the responses y, since these are clearly not independent of x. However, we might assume they are independent, since our only goal is to generate a range of $\theta' s$ from which to sample. The results of real experiments will quickly fix this.

8.2.2 The experimental model

The next step is to design the experimental model that determines the probability of the response $\hat{y} \sim f^y(y|x, \theta)$. This typically comes from the nature of the problem. Some examples include

- Normal distribution - This is a natural choice when the responses are continuous, and where the noise from an observation of $\hat{y}$ is due to experimental variability, or a range of complex factors. Keep in mind that the normal allows negative outcomes, which may not be appropriate for many situations. However, it is often the case that a normal approximation of the error is very accurate.
- Poisson distribution - This is appropriate when the response represents a count, such as the number of sales of a product, or the number of ad-clicks for an online ad.
- Logistic distribution - This would be appropriate when the response is a 0/1 outcome such as a success or failure.
- Exponential distribution - Exponential distributions are often a good fit when we are observing a time interval, especially when the interval might be quite small. For example, we might be modeling the time between the arrival of patients with a particular condition.

For each distribution, θ captures the parameters of the distribution, such as the mean and variance for a normal distribution, the mean λ

The point is that the distribution $f^y(y|\theta)$ is determined by the characteristics of the problem, guided by a (presumably) unknown parameter vector θ.

8.2.3 Sampling policies

With our sampled belief model,

8.2.4 Resampling

8.3 THE KNOWLEDGE GRADIENT FOR SAMPLED BELIEF MODEL

We are going to demonstrate a powerful method for calculating the knowledge gradient that works with very general nonlinear parametric models. We first return to the knowledge gradient formula, which we repeat here for convenience:

$$\nu^{KG,n}(x) = \mathbb{E}_\theta \mathbb{E}_{W|\theta}\{\max_{x'} f(x', \theta^{n+1}(x))|S^n, x = x^n\} - \mathbb{E}_\theta\{\max_{x'} f(x', \theta)|S^n\}. \tag{8.29}$$

It is useful to break down what is involved in computing the expectations. The first and most complex expectation is over the unknown parameter vector θ which is typically multidimensional, with as few as one or two but perhaps hundreds of dimensions. This same expectation appears in the second term. The second expectation is over the outcome of the experiment, which is a scalar, and possibly binomial (success/failure).

We are going to demonstrate a powerful and practical way for approximating the knowledge gradient by using the sampled belief model, which we first saw in section 2.4. We begin by assuming that the true θ may take on one of a finite (and not too large) set $\theta_1, \ldots, \theta_K$. Let $p_k^n = \mathbb{P}[\theta = \theta^k]$ be the probability that θ_k is the true value, given what we know after n experiments. Finally, we let our belief state $S^n = (\theta_k, p_k^n)_{k=1}^K$.

Using the sampled belief model allows us to compute the conditional expectation given our belief S^n using

$$\begin{aligned}\mathbb{E}_\theta\{\max_{x'} f(x', \theta)|S^n\} &= \bar{f}^n(x') \\ &= \sum_{k=1}^K p_k^n f(x', \theta_k).\end{aligned}$$

Now assume we run an experiment using settings $x^n = x$, and observe the outcome W^{n+1} from the $n + 1st$ experiment. Assume that the distribution of W^{n+1} is given by

$$f^W(w|x, \theta) = \mathbb{P}[W^{n+1} = w|x, \theta].$$

We then use this to update our probabilities using the Bayesian updating logic we presented in 2.4, which is given by

$$p_k^{n+1}(x = x^n|W^{n+1} = w) = \frac{1}{C_w} f^W(W^{n+1} = w|x^n, \theta_k) p_k^n, \tag{8.30}$$

where C_w is the normalizing constant given $W^{n+1} = w$, which is calculated using

$$C_w = \sum_{k=1}^K f^W(W^{n+1} = w|x^n, \theta_k) p_k^n.$$

Below, we are going to treat C_W (with capital W) as a random variable with realization C_w.

Using the posterior distribution of belief allows us to write $f(x', \theta^{n+1}(x))$ given S^n using

$$f(x', \theta^{n+1}(x)) = \sum_{k=1}^{K} p_k^{n+1}(x) f(x', \theta_k).$$

Substituting this into equation (8.29) gives us

$$\nu^{KG,n}(x) = \mathbb{E}\left\{ \max_{x'} \sum_{k=1}^{K} p_k^{n+1}(x) f(x', \theta_k) | S^n, x = x^n \right\} - \sum_{k=1}^{K} p_k^n f(x, \theta_k). \quad (8.31)$$

Substituting $p_k^{n+1}(x = x^n|W)$ from equation (8.30) into (8.31) gives us

$$\nu^{KG,n}(x) = \mathbb{E}_\theta \mathbb{E}_{W|\theta} \left\{ \max_{x'} \frac{1}{C_W} \sum_{k=1}^{K} \left(f^W(W|x^n, \theta_k) p_k^n \right) f(x', \theta_k) | S^n, x = x^n \right\} - \sum_{k=1}^{K} p_k^n f(x, \theta_k). \quad (8.32)$$

We now have to deal with the two outer expectations. The inner expectation $\mathbb{E}_{W|\theta}$ is not too bad since the outcome of an experiment W is a scalar which might be normally distributed or binary (success/failure). The more difficult problem is the outer expectation $\mathbb{E}_\theta$ since θ is often a vector, but here is where we again use our sampled representation of θ.

Keeping in mind that the entire expression is a function of x, the expectation can be written

$$\begin{aligned}
&\mathbb{E}_\theta \mathbb{E}_{W|\theta} \left\{ \max_{x'} \frac{1}{C_W} \sum_{k=1}^{K} p_k^n f^W(W|x^n, \theta_k) f(x', \theta_k) | S^n, x = x^n \right\} \\
&= \mathbb{E}_\theta \mathbb{E}_{W|\theta} \frac{1}{C_W} \left\{ \max_{x'} \sum_{k=1}^{K} p_k^n f^W(W|x^n, \theta_k) f(x', \theta_k) | S^n, x = x^n \right\} \\
&= \sum_{j=1}^{K} \left(\sum_{\ell=1}^{L} \frac{1}{C_{w_\ell}} \left\{ \max_{x'} \sum_{k=1}^{K} p_k^n f^W(W = w_\ell|x^n, \theta_k) f(x', \theta_k) | S^n, x = x^n \right\} f^W(W = w_\ell|x, \theta_j) \right) p_j^n.
\end{aligned} \quad (8.33)$$

If we had to stop here, we would find that equation (8.33) is computationally difficult to compute. The biggest problem is that there are three sums over the set of K values of θ: the outer sum over truths, the inner sum to compute the posterior distribution, and the sum to compute the denominator of the posterior distribution. We then also have the sum over outcomes of W, and the maximization over x'. Fortunately, we can simplify this.

We start by noticing that the terms $f^W(W = w_\ell|x, \theta_j)$ and p_j^n are not a function of x' or k, which means we can take them outside of the $\max$ operator. We can also

reverse the order of the other sums over k and w_ℓ, giving us

$$\mathbb{E}_\theta \mathbb{E}_{W|\theta} \left\{ \max_{x'} \frac{1}{C_W} \sum_{k=1}^{K} p_k^n f^W(W|x^n, \theta_k) f(x', \theta_k) | S^n, x = x^n \right\}$$
$$= \sum_{\ell=1}^{L} \sum_{j=1}^{K} \left(\frac{f^W(W = w_\ell|x, \theta_j) p_j^n}{C_{w_\ell}} \right) \left\{ \max_{x'} \sum_{k=1}^{K} p_k^n f^W(W = w_\ell|x^n, \theta_k) f(x', \theta_k) | S^n, x = x^n \right\}. \tag{8.34}$$

Using the definition of the normalizing constant C_w we can write

$$\begin{aligned} \sum_{j=1}^{K} \left(\frac{f^W(W = w_\ell|x, \theta_j) p_j^n}{C_{w_\ell}} \right) &= \left(\frac{\sum_{j=1}^{K} f^W(W = w_\ell|x, \theta_j) p_j^n}{C_{w_\ell}} \right) \\ &= \left(\frac{\sum_{j=1}^{K} f^W(W = w_\ell|x, \theta_j) p_j^n}{\sum_{k=1}^{K} f^W(W = w_\ell|x, \theta_k) p_k^n} \right) \\ &= 1. \end{aligned}$$

We just simplified the problem by cancelling two summations over the K values of θ. This is a significant simplification, since these sums were nested. This allows us to write (8.34) as

$$\mathbb{E}_\theta \mathbb{E}_{W|\theta} \left\{ \max_{x'} \frac{1}{C_W} \sum_{k=1}^{K} p_k^n f^W(W|x^n, \theta_k) f(x', \theta_k) | S^n, x = x^n \right\}$$
$$= \sum_{\ell=1}^{L} \left\{ \max_{x'} \sum_{k=1}^{K} p_k^n f^W(W = w_\ell|x^n, \theta_k) f(x', \theta_k) | S^n, x = x^n \right\}. \tag{8.35}$$

Equation (8.35) means that we can compute fairly good approximations of the knowledge gradient even in the presence of a nonlinear belief model, although some care would have to be used if θ was high dimensional.

8.4 LOCALLY QUADRATIC APPROXIMATIONS

8.5 BIBLIOGRAPHIC NOTES

Section 7.2 -

PROBLEMS

8.1 Imagine that we are using a logistics curve to model the probability that a customer will accept a bid, which gives us the revenue function

$$R(p; \theta_1, \theta_2) = \frac{p}{1 + e^{-(\theta_1 - \theta_2 p)}}. \tag{8.36}$$

Assume the prior is $\bar{\theta}^0 = (4, 2)$ for prices that are between 0 and 5.

a) Discretize prices into increments of 0.10 and find the optimal price $p^{Exp,1}$ given this prior. This represents the choice you would make using a pure exploitation policy.

b) Assume we choose a price $p^0 = 2$ and we observe that the customer accepts the bid (that is, $W^1 = 1$). Use equations (8.16) and (8.17) to find an updated model. Note that we did not need to assume a variance for the experimental noise. Why is this?

c) Now use the Bayes-greedy policy in equation (8.25) to compute the price $p^{BG,1}$. Try to explain intuitively the difference between $p^{BG,1}$ and $p^{Exp,1}$.

d) Simulate the Bayes-greedy policy for 20 experiments, using (8.16) and (8.17) to update your beliefs. At each iteration, using the beliefs generated by the Bayes-greedy policy, compute $p^{Exp,1}$, and compare the two policies over 20 iterations.

e) Finally, simulate the pure exploitation policy and the Bayes-greedy policy for 20 iterations. Repeat this process 100 times and summarize the difference.

CHAPTER 9

LARGE CHOICE SETS

While there are problems where the number of alternative decisions $\{x_1, \ldots, x_M\}$ is small enough to be enumerated, it is quite easy to encounter problems where the potential set is extremely large (tens of thousands to millions). This typically arises when x is a vector, but sometimes there are simply many choices (imagine choosing ads or movies to show an online user).

In this chapter, we consider three classes of problems where large choice sets arise:

Subset selection These arise when we have to choose L elements out of a set K, such as choosing five basketball players from a team of 12. The number of potential subsets grows very quickly as K and L increase.

Hierarchical models Imagine having to pick the best person to do a job, or the best ad to maximize ad-clicks, where the choice is characterized by a multidimensional attribute vector, producing an extremely large set of choices.

Continuous vectors There are many problems where x is a multi-dimensional vector $(x_1, \ldots, x_K)$, where x_k might be continuous or an integer.

These problems will allow us to illustrate several powerful strategies that should provide a good starting toolbox for a wide range of problem settings.

Optimal Learning. By Warren B. Powell and Ilya O. Ryzhov

9.1 SAMPLED EXPERIMENTS

A powerful strategy for handling large experimental spaces is to turn a large set of experiments into a small one by choosing a smaller sample from the original large set. If x is multidimensional, we can sample each dimension independently, rejecting any that are not feasible or acceptable. This is a simple but powerful idea that is easy to implement. How well it works depends on the structure of the problem and the nature of the underlying belief model.

Sampling is effective when we use a belief model that allows us to generalize what we learn. This is the case if we have correlated beliefs (presented in section 4.5), a linear (or nonlinear) belief model (which we presented in chapters 7 and 8), or a hierarchical belief model (which we present below in section 9.3). The core idea is that with some form of generalized learning, we do not have to test every experiment in $\mathcal{X}$. Rather, we just need to sample enough to produce a reasonable estimate of our belief model, from which we will then find the best value in the set $\mathcal{X}$.

It is easy to overlook that most statistics is done using sampled experiments. The most standard is when we are given a dataset consisting of N data points $(y^n, x^n)_{n=1}^N$ from which we have to fit a statistical model. We may have no idea how the inputs x^n were chosen. Often, they come from a process over which we have no control. For example, x^n might be the attributes of a patient and y^n is the cost of the medical procedure. We typically cannot control which patients walk through the door.

The size of $\mathcal{X}$ is much more important for a value of information policy than it is for the other heuristics. Thus, we might need to draw a relatively smaller sample for the purpose of computing the knowledge gradient or expected improvement just to simplify these calculations. Once the experiments are complete, we can use a much larger sample to try to find the best design.

There are different strategies for generating and using sampled experiments:

Static samples We can draw a single sample from the set $\mathcal{X}$ and then proceed using just this sampled set for all calculations.

Iterative resampling Since our belief model provides estimates for all $x \in \mathcal{X}$, and not just the sampled experiments, we can draw an entirely new sample each time we want to find an experiment to run. This reduces the exposure to the vagaries of a single sample.

Using a sampled subset of experiments from $\mathcal{X}$, combined with a generalized belief model, is an exceptionally powerful strategy. The size of the sample depends on the behavior of the belief model.

It is possible to approach virtually any problem regardless of the size of the experimental set $\mathcal{X}$, and regardless of the complexity of the underlying belief model, using the idea of a sampled experimental set. You simply have to choose a sample $x_1, \ldots, x_M$, and then use the tools for discrete alternatives that we have presented in depth. However, even if you can exploit the structure of correlated beliefs, it will still be harder to estimate $\mu_1, \ldots, \mu_M$ (where M might be 1000), when you can reduce this to a parametric belief model $f(x|\theta)$ where $\theta = (\theta_1, \ldots, \theta_K)$.

9.2 SUBSET SELECTION

Consider the problem where we have to pick the four best rowers to man a four-person shell for rowing competitions. The coach can rank the 15 rowers on his team based on how they perform on ergonomic machines, but it is well known that people perform differently in a boat, and people interact in ways that affect their performance. However, there is over 1,365 ways of choosing four people out of a set of 15, so it is not possible to evaluate every possible team.

Our challenge is to find the best subset of a discrete choice set. This problem, known as *subset selection*, arises in numerous settings and has a classical formulation as a mathematical program. For example, we might have M items, where $x_i = 1$ if we have chosen i to be in our subset. Normally we have some sort of budget constraint, so we have to choose the vector x to satisfy this constraint. If we knew that item i made a contribution c_i, we could choose our elements by solving

$$\max_x \sum_{i=1}^{M} c_i x_i$$

subject to

$$\begin{aligned} \sum_{i=1}^{M} a_i x_i &\leq B, \\ x_i &\in (0,1). \end{aligned}$$

If $a_i = 1$, then B is simply a limit on how many items we can have (for example, four rowers). The coefficient a_i might be the cost of using i, in which case B is a monetary budget constraint. In this case, we have an instance of the well-known knapsack problem.

Subset selection problems include forming a team, identifying investment portfolios, prescribing a course of treatment for an illness, and choosing tools to solve a problem. In all of these problems, we have to choose a subset of elements that interact with each other in a relatively complex way. Our specific interest is in problems where we do not have a simple deterministic, linear objective function. Because of the interactions between elements, the problem is harder than simply not knowing the coefficients c_i. However, we may have a learning budget allowing us to experiment with several subsets before we have to choose one. For example, a basketball coach may hold some practice games to learn how well different players work together, before committing to a lineup for the season.

We can apply optimal learning concepts to subset selection. In theory, the tools developed in Chapters 3 and 4 already allow us to handle this problem. We can view subset selection as an instance of a ranking and selection problem with correlated beliefs. Each alternative in the ranking and selection problem corresponds to one possible subset. In the problem of choosing four rowers out of 15 possible candidates, we thus have a total of $M = \binom{15}{4} = 1,365$ alternatives. We model the values of different subsets separately. If we choose rowers $\{\text{Anne}, \text{Mary}, \text{Cathy}, \text{Tara}\}$ and $\{\text{Anne}, \text{Mary}, \text{Susan}, \text{Tara}\}$, these two subsets would be considered as two distinct

alternatives, each with its own value. However, our beliefs about these two values are heavily correlated, because these two subsets have three elements in common. Learning about the effectiveness of one subset should also provide information about the other subset.

As long as we do a good job modeling our prior beliefs (particularly our beliefs about the correlations between alternatives), we can then apply one of the learning techniques from Chapter 3, or the knowledge gradient method from Chapter 4, to decide which subset we want to learn about. Conceptually, we can approach the problem of subset selection entirely within the framework of ranking and selection. For smaller problems such as choosing four rowers out of 15, this approach will work fine.

However, one characteristic specific to subset selection problems is that they tend to be quite large. Suppose that we need to choose five items from a list of ten. Five and ten are small numbers, but the number of possible subsets is $\binom{10}{5} = 252$. If we slightly increase the size of the problem, and now choose six items from a list of twelve, the number of subsets is $\binom{12}{6} = 924$. The number of alternatives grows combinatorially as we add more items to our list. Choosing ten items from a list of twenty results in $184,756$ alternatives!

In Chapter 3, we did not really specify a problem size. Rather, we analyzed a generic problem with M alternatives. In theory, the techniques we developed can handle any M, but as a general rule we are limited to M equal to around 1,000. When we are choosing subsets, the number of subsets can quickly grow much larger than this.

Subset selection problems come in many varieties. We are primarily interested in applications where there is some relationship between subsets so we can draw on our tools using correlated beliefs.

- Designing a diabetes treatment - Type 2 diabetes is an illness where the body is unable to properly react to insulin, a hormone used to remove sugar from the bloodstream. Most diabetes drugs fall into one of several categories; for example, sensitizers increase the body's sensitivity to insulin, whereas secretagogues stimulate the body to secrete more insulin. A course of treatment for a single patient may contain drugs from multiple categories, as well as multiple complementary drugs within a single category. Clinical trials can help us learn how well two drugs complement each other.

- Choosing a starting basketball lineup - The coach wants to find five players who provide the best performance over the first quarter of a basketball game. He knows how many points each player scores from past games, and has other statistics such as rebounds, assists and blocks, but the simple reality is that there is no easy formula that predicts how people will interact with each other. Each time he tries out a team, he observes the total score (which might be viewed as a correction to a simpler formula). The coach has 12 players, implying that he has to pick the best set of 5 out of a team of 12, which means evaluating 792 different teams. Fortunately, the observations should be correlated. We would not know the correlations exactly, but we might start with a covariance matrix that is proportional to the number of players in common between two teams.

- Choosing movies to display to a user - We wish to select a sample of movies (perhaps 6 or 8) out of a population of thousands to display to a user in the hope that she will select one to rent or purchase. The value of a set is given by the probability that she selects one, which requires guessing whether she is interested in action, drama, comedy, or is more influenced by the lead actor/actress. We have to find the best subset of movies to display to maximize the probability of a purchase. It is important to try a mixture, because if we display all action movies to someone who prefers drama, we will not be successful.

- Product assortment planning - A retail company has to choose a set of products to offer at its outlet store, or to display in a storefront window. The profitability of a product, or its effectiveness at drawing customers to a store, may also depend on the presence of other products in the assortment (e.g. accessories for an iPod). It is also important to display a diverse set of products that may attract buyers from different demographic groups. We have a short period of time to experiment with different assortments at a particular store in an attempt to discover the best one.

- Shortest path problems - Finding a path through a network with uncertain costs (or travel times) involves finding a subset of links out of the full set of links in the network. Of course, these links have to form a path that joins origin to destination.

9.2.1 Choosing a subset

We show how ranking and selection can be used to find a subset in the context of designing a diabetes treatment. In the latter stages of clinical trials, the search has typically been narrowed to a relatively small set of the most promising drugs. We can create a combination treatment consisting of multiple drugs from this set.

The effectiveness of a diabetes treatment can be measured in terms of blood sugar reduction. Blood sugar level is measured in millimoles per liter, after the patient has not eaten for eight hours; this is known as the "fasting plasma glucose" or FPG level. For a healthy person, the FPG level is about 4-6 mmol/L. A diabetic can show FPG levels of up to 10-15 mmol/L. A single drug can reduce FPG level by an amount between 0.5 and 2 mmol/L.

We design a combination treatment for two reasons. First, two drugs yield a bigger FPG reduction than one, and second, combination treatments tend to have lower risks of side effects. Prescribing three drugs instead of one may give us an FPG reduction between 2 and 5 mmol/L. Due to interactions between drugs, FPG reduction obtained from the combination is not a straightforward sum of the reductions we might observe from individual drugs.

9.2.2 Setting prior means and variances

Let us consider an example with six drugs. Our list may include certain basic elements such as conventional treatment (recommending diet and exercise to the patient), metformin (the first drug of choice in most cases), and insulin injections, as well as newer compounds such as rosiglitazone, glibenclamide and chlorpropamide.

Our goal is to find the best possible combination of three of these drugs. The total number of subsets is still quite small, $\binom{6}{3} = 20$.

Let $\mathcal{S} = \{1, ..., 6\}$ be our set of $S = 6$ drugs, from which we have to choose a subset of size s. (These should be pronounced "script S" and "big S.") Let μ_x be the true value of a subset $x \subseteq S$, that is, the true FPG reduction achieved by treatment x on average. Now suppose that we have a large population of patients in roughly the same poor health condition. We divide them into N groups and then we prescribe a treatment for each group. When we assign the $(n+1)$st group to treatment x, our observation W_x^{n+1} is the average FPG reduction across all the patients in the group. Sample averages are approximately normal, so we make our usual modeling assumption $W_x^{n+1} \sim \mathcal{N}\left(\mu_x, \sigma_W^2\right)$, where the variance σ_W^2 is assumed known (but somehow estimated or guessed in practice).

All we need now in order to apply optimal learning is a prior distribution on the values μ_x. We will use a multivariate normal distribution (accounting for correlations between subsets). The prior mean $\bar{\mu}_x^0$ on the true value of treatment x tends to be easier to choose, because it comes directly from our domain knowledge about the problem. In light of what we know about blood sugar levels, we might let $\bar{\mu}_x^0$ be a number between 2 and 5 mmol/L. The prior variance Σ_{xx}^0 represents a rough guess about the range of our uncertainty about the true value. If we are reasonably sure that the true value is between 2 and 5, we might let the variance be 0.5, meaning that we believe that μ_x falls in the range $\bar{\mu}_x^0 \pm 2 \cdot \sqrt{0.5}$.

9.2.3 Two strategies for setting prior covariances

While prior means and variances can be chosen with the help of knowledge about the problem, covariances between the true values are trickier to estimate. In actual clinical trials, we can observe the effectiveness of a treatment directly, but covariances between treatments can only be inferred from the results of individual trials. Fortunately, we can still obtain good performance with a simple, heuristic covariance structure. Our prior mainly needs to capture the fact that subsets are more heavily correlated if they have more elements in common.

Our first rule of thumb is based purely on the number of such elements. In the problem of choosing three drugs from a list of six, let $x = \{1, 2, 3\}$ and $y = \{2, 3, 5\}$ be two possible treatments. These subsets have $2/3$ elements in common, so we can simply choose $2/3$ as the correlation coefficient of these two subsets in our prior distribution. That is, the prior correlation coefficient is set using

$$\rho_{xy}^0 = \frac{1}{S} |x \cap y| .$$

The prior covariance is simply

$$\Sigma_{xy}^0 = \rho_{xy}^0 \sigma_x^0 \sigma_y^0. \tag{9.1}$$

We calculate these covariances for every possible x and y. This simple heuristic produces high correlations between subsets with more common elements.

In some problems, we may have more domain knowledge about individual items on our list than about subsets. For example, we may have access to clinical data about the drug rosiglitazone, tested independently of other drugs. Or, we may have some data on the travel time on a particular street or region, but no travel time data for a complex travel route that takes us through multiple regions. In this case, it may be easier to construct a prior on the value of a subset using our information about individual components.

Let μ_i^{ind} be the true value of the individual component $i \in \mathcal{S}$ (e.g. the FPG reduction achieved by the single diabetes drug i). Using our domain knowledge, we construct a prior $\mu_i^{ind} \sim \mathcal{N}\left(\bar{\mu}_i^{ind,0}, \left(\sigma_i^{ind,0}\right)^2\right)$. For the diabetes setting, we may believe that μ_i^{ind} falls in the range 1.25 ± 0.75 mmol/L. Then, our belief about a treatment x is given by

$$\bar{\mu}_x^0 = \sum_{i \in x} \bar{\mu}_i^{ind,0}, \tag{9.2}$$

and the covariance of our beliefs about treatments x and y is given by

$$\Sigma_{xy}^0 = \sum_{i \in x \cap y} \left(\sigma_i^{ind,0}\right)^2. \tag{9.3}$$

This heuristic implicitly assumes that the values of individual diabetes drugs are independent, which may not be the case. For example, the FPG reductions for multiple drugs in the same class will be correlated. If a patient reacts adversely to sensitizers, all drugs of this type will tend to perform poorly. In any case, however, our prior is not able to capture all the nuances of the problem. Instead, the prior is meant to provide some rough guidelines about the values of different alternatives and the interactions between them. Even if we assume that individual items on our list are independent, subsets will still be correlated if they have common elements. The correlation structure in (9.3) once again captures the fact that subsets with many common elements should have similar values. As we make decisions, the information we collect will allow us to obtain more accurate beliefs. To put it another way, we start by assuming a simple linear objective function, then leave it to our learning algorithm to refine these initial beliefs.

We now proceed as usual. Once we have a prior $\left(\bar{\mu}^0, \Sigma^0\right)$, we rely on our standard Bayesian updating equations to change our beliefs from that point. If we choose treatment x for the trial at time n, our beliefs for subsets y will change according to the equations

$$\begin{aligned}
\bar{\mu}_y^{n+1} &= \bar{\mu}_y^n - \frac{W_x^{n+1} - \bar{\mu}_x^n}{\sigma_W^2 + \Sigma_{xx}^n} \Sigma_{xy}^n, \\
\Sigma_{y,y'}^{n+1} &= \Sigma_{y,y'}^n - \frac{\Sigma_{x,y}^n \Sigma_{x,y'}^n}{\sigma_W^2 + \Sigma_{xx}^n}.
\end{aligned}$$

This gives us everything we need to run an algorithm such as knowledge gradient. If there are only 20 possible subsets, it is smooth sailing from here.

9.2.4 Managing large sets

Let us switch gears and consider a slightly larger problem, namely the energy portfolio selection example we introduced above. A recent study by McKinsey & Company (2007) has identified a number of promising technologies for reducing greenhouse gas emissions. These new technologies have a higher energy efficiency, and also help to reduce energy costs. A partial list of these technologies includes: residential lighting; energy-efficient water heaters and appliances; improved ventilation systems and air conditioners; heating systems and furnaces; solar technology; shell improvements for residential buildings (e.g. insulation); and fuel economy packages for cars.

Suppose that we have a total of 12 promising technologies, and our goal is to create a portfolio of six. The McKinsey study provides estimates for the abatement potential (the emissions reduction) of these technologies. The estimates represent the total abatement, measured in gigatons of CO_2 per year (ranging from 0.2 to 3.0 for different technologies), that could be achieved by implementing the technology across the entire United States. We can scale down these estimates and use the strategy from (9.2) and (9.3) to create a prior for the energy efficiency of a single portfolio of six technologies, applied to a single building.

The total number of energy portfolios in this example is 924, about 50 times more than we had in the problem of designing a diabetes treatment. We can still run the correlated KG algorithm from Chapter 4, but we will see that it will run much more than 50 times slower on this problem. The reason is because, in a problem with M subsets, the time required to compute correlated KG factors for all the subsets is proportional to $M^2 \log M$. This computational cost grows faster than the problem size. Considering that subset selection problems are already prone to very large sizes, this is a serious issue. We now discuss a way to reduce this cost using Monte Carlo simulation.

9.2.5 Using simulation to reduce the problem size

We are going to propose an idea which can be viewed as a hybrid of Thompson sampling and any other policy, although this is of greatest value for the knowledge gradient which is harder to compute. We use posterior sampling (as is done in Thompson sampling) to identify a subset of promising alternatives. We are then going to use this more reasonable subset to use tools such as correlated beliefs.

Suppose that we are at time n, with point estimate $\bar{\mu}^n = (\bar{\mu}^n_{x_1}, \ldots, \bar{\mu}^n_{x_M})$ and covariance matrix Σ^n with element $\Sigma^n_{xx'} = Cov^n(\mu_x, \mu_{x'})$. Note that each x is an entire subset, such as

$$x = (0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1).$$

A subset x might also be a path in our network, or a set of drugs in a drug regimen.

What we want to do is to create K samples of the vector μ from the distribution $\mathcal{N}(\bar{\mu}^n, \Sigma^n)$. Let $\hat{\mu}^k$ be the kth sample of this vector, and let

$$\hat{x}^k = \arg\max_{x \in \mathcal{X}} \hat{\mu}^k_x$$

be the best alternative out of this sample of beliefs across all alternatives. Constructing this set might require computing $\hat{\mu}_x^k$ for each alternative x, but not always. For example, if we are finding paths through a graph, we would randomly generate costs on each link, and then solve a shortest path problem for this set of random costs, which has the effect of implicitly enumerating all possible paths without actually enumerating them.

If we repeat this K times, we get a set of alternatives $\hat{x}^1, \ldots, \hat{x}^K$, which may have some duplicates which have to be removed (if we insist on having a full set K, we can continue the process until we have K distinct alternatives). This set is likely to consist of K high quality alternatives, but as with Thompson sampling, there is an element of exploration.

To see how this works, consider a numerical example. Suppose that we have five alternatives with $\bar{\mu}^n = (10, 13, 12, 16, 8)^T$ and

$$\Sigma^n = \begin{bmatrix} 22 & 13 & 14 & 15 & 14 \\ 13 & 16 & 18 & 13 & 12 \\ 14 & 18 & 24 & 12 & 14 \\ 15 & 13 & 12 & 14 & 11 \\ 14 & 12 & 14 & 11 & 16 \end{bmatrix}.$$

We decide to reduce the number of alternatives. To that end, we generate $K = 4$ samples from the distribution $\mathcal{N}(\bar{\mu}^n, \Sigma^n)$. Table 9.1 gives the simulated values for all five alternatives on all four sample paths. Observe that, according to our prior, alternative 4 is believed to be the best. This same alternative is also the winner on $2/4$ sample paths. However, alternatives 2 and 3 each win on a single sample path, even though these alternatives are believed to be the second- and third-best according to the prior. The set of winners is thus $\{2, 3, 4\}$. In this way, Monte Carlo sampling can give us a reasonably diverse choice set, which includes many promising alternatives, but does not include alternatives that seem to be hopelessly bad.

It can be shown that the marginal distribution of (μ_2, μ_3, μ_4) is multivariate normal with parameters

$$\bar{\mu}^{MC,n} = \begin{bmatrix} 13 \\ 14 \\ 15 \end{bmatrix}, \qquad \Sigma^{MC,n} = \begin{bmatrix} 16 & 18 & 13 \\ 18 & 24 & 12 \\ 13 & 12 & 14 \end{bmatrix}.$$

x	$\hat{\mu}_x^n(\omega^1)$	$\hat{\mu}_x^n(\omega^2)$	$\hat{\mu}_x^n(\omega^3)$	$\hat{\mu}_x^n(\omega^4)$
1	15.59	11.38	3.73	10.48
2	17.24	18.49	7.30	**18.98**
3	18.06	**20.77**	2.61	18.88
4	**19.32**	17.97	**12.53**	18.73
5	12.37	13.41	4.08	11.55

Table 9.1 Simulated values of five alternatives based on the prior distribution.

That is, if we throw out alternatives 1 and 5, the distribution of our belief about the remaining alternatives is the same as before. We just need to throw out the rows and columns of Σ^n, and the elements of $\bar{\mu}^n$, having to do with alternatives 1 and 5. We obtain the reduced prior $\mathcal{N}\left(\bar{\mu}^{MC,n}, \Sigma^{MC,n}\right)$, where the notation MC refers to the use of Monte Carlo sampling to reduce the prior.

We can formalize this procedure as follows. Let K_0 be the number of unique winners obtained from Monte Carlo sampling (with duplicates removed; this number would be 3 in the preceding example). We can enumerate the winners as $\bar{x}_1^{MC,n}$, ..., $\bar{x}_{K_0}^{MC,n}$. We can define a matrix A^n of size $M \times K_0$ by

$$A^n = \left[e_{\bar{x}_1^{MC,n}}, ..., e_{\bar{x}_{K_0}^{MC,n}}\right].$$

Recall that e_x is a vector of zeroes with only the xth component set to 1. In the preceding example, we would write

$$A^n = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}.$$

Then, we can calculate the reduced prior using the equations

$$\begin{aligned} \bar{\mu}^{MC,n} &= \left(A^n\right)^T \bar{\mu}^n, \\ \Sigma^{MC,n} &= \left(A^n\right)^T \Sigma^n A^n. \end{aligned}$$

We can now run the knowledge gradient algorithm using $\bar{\mu}^{MC,n}$ and $\Sigma^{MC,n}$ as the prior instead of the original parameters $\bar{\mu}^n$ and Σ^n. The computational cost is thus at most $K^2 \log K$, and likely less if there are many duplicates. We choose K to be much smaller than M. In a problem with 924 alternatives, we can obtain good results with $K = 30$.

9.2.6 Computational issues

Monte Carlo simulation allows us to greatly reduce the cost of calculating knowledge gradients. However, it carries a new cost for running the simulations. Recall from Section 2.7 that a single sample from a multivariate normal distribution can be created using the equation

$$\hat{\mu}^n = \bar{\mu}^n + C^n Z$$

where $Z = (Z_1, ..., Z_M)$ is a vector of standard normals, and C^n is a "square-root matrix" satisfying $C^n \left(C^n\right)^T = \Sigma^n$. The standard normals are easy enough to generate, but calculating the square root matrix can be costly. In the worst case, the time required to compute C^n is proportional to M^3, which is actually greater than the $M^2 \log M$ needed to compute the full set of knowledge gradients in the first place.

Fortunately, there is an elegant work-around for this issue. The square-root matrix can be computed recursively using the formula

$$C^{n+1} = C^n - \frac{\Sigma^n e_{x^n} e_{x^n}^T C^n}{(\sigma_W^2 + \Sigma^n_{x^n x^n}) \left(1 + \sqrt{\frac{\sigma_W^2}{\sigma_W^2 + \Sigma^n_{x^n x^n}}}\right)}. \tag{9.4}$$

This equation is entirely analogous to (2.19). Given our decision x^n at time n, C^{n+1} can be calculated directly from the beliefs at time n. The updating equation (9.4) even has a similar form to (2.19), and requires the same computational effort.

Thus, we only need to perform the expensive calculations required to compute the square root of a matrix once, to obtain C^0 from Σ^0. In MATLAB, this can be done using the routine `chol` (for Cholesky factorization, the name for the square root procedure). After that, we keep track of the square root matrix at each time step, and the next square root is computed recursively at a low cost. We are saving a lot of computing time by extending our belief state to include C^n as well as Σ^n.

It may also sometimes happen that, due to numerical rounding errors, the prior covariance matrix Σ^0 that we build using the methods from Section 9.2.3 may be reported by MATLAB as not being positive semidefinite (a necessary condition for computing square root matrices). This issue can often be resolved by simply adding a diagonal matrix $\varepsilon \cdot I$ to Σ^0. Here, I is the identity matrix and ε is a small positive number (e.g. 0.001). This quick fix is often enough for computing C^0, while still preserving the general correlation structure of subset selection.

9.2.7 Experiments

Earlier, we mentioned that the number K of Monte Carlo samples can be chosen to be much smaller than the total number M of subsets. One important question is exactly how many samples we need to take. Figure 9.1 shows how the size K_0 of the reduced choice set (recall that this is the number of winners obtained from Monte Carlo sampling) increases with the sample size K on a variant of the energy portfolio selection problem. Due to the random sampling, the curve is noisy, but we can see that K_0 grows fairly slowly. With $K = 30$ samples, we obtain a reduced choice set of $K_0 \approx 20$ alternatives. This number is roughly doubled when K increases to 300. Thus, as long as our prior presents a roughly accurate picture of the values of the alternatives (in Chapter 6 we referred to this as a "truth-from-prior" problem), we can safely ignore the vast majority of the possible subsets and focus only on several dozen of the most promising choices.

Figure 9.2.7 illustrates the performance, expressed using the average opportunity cost metric from (6.10), of the Monte Carlo KG (MCKG) policy as a function of the sample size K. We see that, once $K \geq 30$, performance begins to level off, and adding more samples does not yield substantially better results. Here, we see that small sample sizes lead to more erratic behavior. In each iteration, our reduced choice set is randomly missing some important, promising portfolios, and it also randomly includes some portfolios that have previously been under-represented. Our policy thus measures more portfolios in an effort to gain enough information about the most relevant alternatives. On the other hand, once the sample size is large enough that all

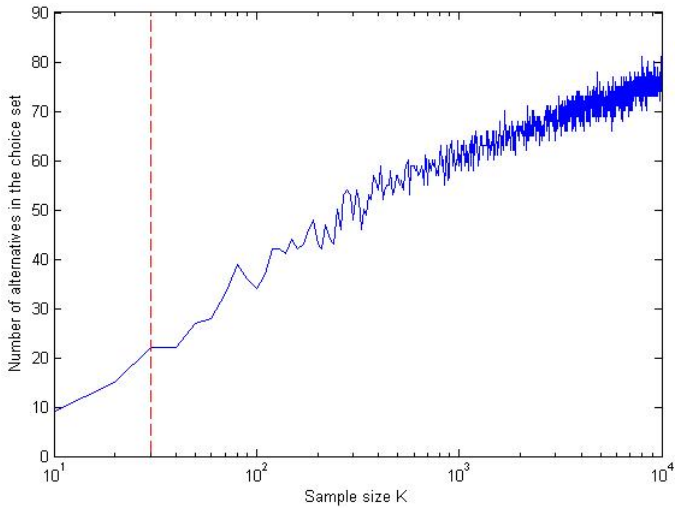


Figure 9.1 Growth of the size K_0 of the reduced choice set as a function of the sample size K.

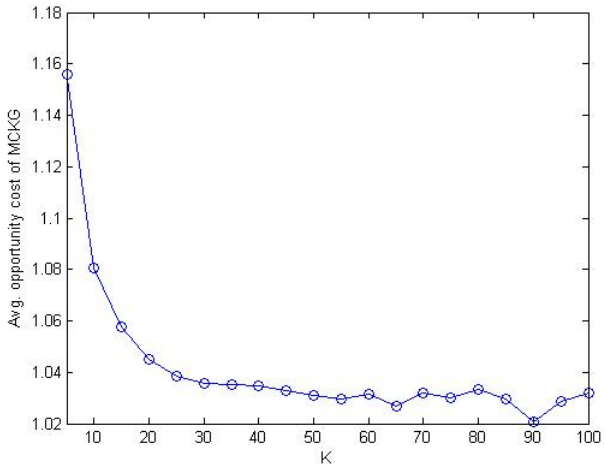


Figure 9.2 Illustration of performance of the Monte Carlo KG (MCKG) policy as the sample size K increases (from Ryzhov & Powell (2009*b*)).

of these important portfolios are consistently included in the reduced choice set, the policy is able to more accurately discern which alternatives need to be measured.

Finally, Figure 9.3 compares the performance of MCKG for a fixed sample size $K = 25$ to several other policies. In addition to our usual mainstays – approximate Gittins indices, interval estimation, and pure exploitation – we also apply the independent KG policy, in which we apply the formula from (4.13) to make a decision, thus temporarily ignoring the correlations between subsets when we make decisions, but incorporating them into our learning model. In Section 4.5, this approach was also called "hybrid KG." Although this idea is clearly not as good as correlated KG, it

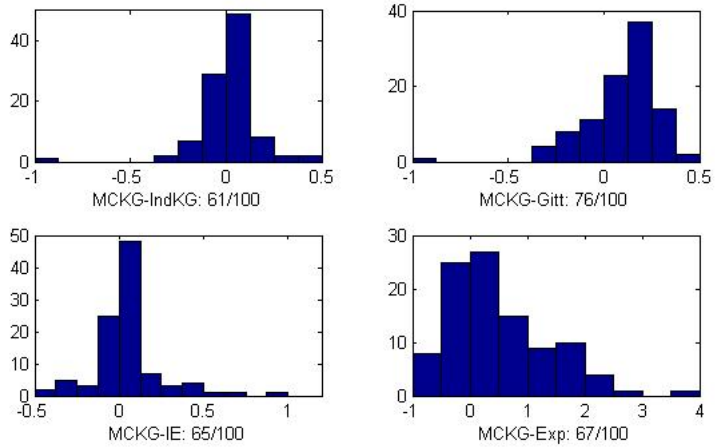


Figure 9.3 Performance of the MCKG policy relative to other policies for $K = 25$. The numbers indicate the number of problems (e.g. 61/100) where MCKG outperformed the competing policy. Results taken from Ryzhov & Powell (2009*b*).

makes sense to use it in this setting because of its much lower computational cost: the time to calculate the formula for each alternative is proportional to M, as opposed to $M^2 \log M$ for correlated KG.

We find that Monte Carlo KG still outperforms the other policies. These results are consistent with what we know about correlated KG from Section 4.5. The Monte Carlo component does not seem to harm performance, as long as K is around 20 or 30. Note that Figure 9.3 does not include a comparison with the full correlated KG policy (computing correlated KG factors for the full choice set). Even in our version of energy portfolio selection, which is still not unreasonably large, it simply takes too much time to run correlated KG. The Monte Carlo modification allows us to work around this computational limitation.

9.3 HIERARCHICAL LEARNING FOR MULTIATTRIBUTE CHOICES

There are situations where we face a large number of possible choices. Some examples are:

■ **EXAMPLE 9.1**

An online video store needs to display a set of movies that a customer is likely to choose. This set can be customized to the particular user, but we need to identify which movies out of thousands that this customer is most likely to choose.

■ **EXAMPLE 9.2**

A company is growing quickly and needs to hire people to fill a variety of new positions. After posting the job requirements on various internet job boards, the human resources director is quickly overwhelmed with thousands of applications.

■ **EXAMPLE 9.3**

The Department of Homeland Security needs to identify websites that suggest terrorist origin or intent. Algorithms can process the websites, but it is necessary to show some of them to experts who can do a better job of assessing risk. Which of the many thousands of websites should we show to the expert?

These are problems with extremely large choice sets, but they lack the nice structure of a subset selection problem. However, each choice is described by a set of attributes which can be exploited to create a natural hierarchy. For example, when organizing movies, we might decide that genre is most important, followed by leading actor/actress, director, and year it appeared. We can then use this structure to create a series of estimates that balance bias and variance.

We have to begin by describing how to compute estimates of a function at different levels of aggregation, since we did not cover this in our first pass through statistical estimation in chapter 9.5. Fortunately, the equations for doing this are relatively simple and quite easy to implement. We do this in two stages. Section 9.3.1 lays the foundation by showing how to estimate bias and variance. Then, section 9.3.2 describes how to perform hierarchical estimation using a simple weighting formula.

Once we have this belief model in hand, we can use this with any of our tunable policies, just as we could when we first encountered correlated beliefs. Value of information policies such as the knowledge gradient, however, are another matter, since we need to capture the updating of the belief model when we calculate the value of information. We show how to calculate the knowledge gradient in section 9.3.3.

9.3.1 Laying the foundations

Aggregation can be achieved in different ways. If x is multidimensional, we can simply ignore dimensions. Example, if estimating the value of posting a movie on a website, we might use genre as the most important attribute, followed by the year in which it was made, and then the acting lead. But we can also aggregate by using coarse representations of a continuous variable, such as the number of words in an ad, or by using advanced statistical models to pick out the most important features.

However we handle aggregation, we can represent our procedure as a mapping that reduces our original set of choices $\mathcal{X}$ to a more aggregate set that we write $\mathcal{X}^{(g)}$, which write write as

$$G^g : \mathcal{X} \rightarrow \mathcal{X}^{(g)}.$$

$\mathcal{X}^{(g)}$ represents the g^{th} level of aggregation of the domain $\mathcal{X}$. Let

$$\begin{aligned}\mathcal{G} &= \text{The set of indices corresponding to the levels of aggregation,}\\ &= \{0, 1, \ldots, G\}.\end{aligned}$$

Throughout we assume that the 0^{th} level of aggregation is the true function. Figure 9.4 illustrates the hierarchical ordering.

$g = 2$	13								
$g = 1$	10			11			12		
$g = 0$	1	2	3	4	5	6	7	8	9

Figure 9.4 Example with nine alternatives and three aggregation levels

We assume that we are sampling the original function μ_x according to

$$\hat{\mu}_x^n = \mu_x + \varepsilon^n,$$

where we may write W_x^n or just W^n if the choice of experimental decision x is clear. We need to characterize the errors that arise in our estimate of the function. Let

$$\begin{aligned}\mu_x^{(g)} &= \text{The true estimate of the } gth \text{ aggregation of the original function } \mu_x\\ \bar{\mu}_x^{(g,n)} &= \text{Our estimate of } \mu_x^{(g)} \text{ after we have run a total of } n \text{ experiments,}\\ \hat{\mu}_x^{(g,n)} &= \text{The observation } \hat{\mu}_x^n \text{ mapped to the } gth \text{ level of aggregation.}\end{aligned}$$

This estimate depends, of course, on the probability that we are going to run experiment x. In our estimation problem, this experimental distribution will be built into our estimates $\bar{\mu}_x^{(g,n)}$.

When we are working at the most disaggregate level ($g = 0$), the experiment x that we run is the observed state $x = \hat{x}^n$. For $g > 0$, the subscript x in $\bar{f}_x^{(g,n)}$ refers to $G^g(x^n)$, or the g^{th} level of aggregation of $f(x)$ at $x = x^n$. Given an observation $(x^n, \hat{f}^n)$, we would update our estimate of the $f^{(g)}(x)$ using

$$\bar{\mu}_x^{(g,n)} = (1 - \alpha_{n-1}^{(g)})\bar{\mu}_x^{(g,n-1)} + \alpha_{n-1}^{(g)}\hat{\mu}_x^n. \tag{9.5}$$

Note that equation (9.5) is performing the process of taking the observation of the function $\hat{\mu}_x^n$ at the disaggregate level x, and is smoothing it into an aggregated estimate $\bar{\mu}_x^{(g,n)}$. The aggregation function is built into the estimate $\bar{\mu}_x^{(g,n)}$.

To illustrate this, imagine that we have a truck driver who is characterized by the attributes

$$x = \begin{pmatrix} \text{Loc} \\ \text{Equip} \\ \text{Home} \\ \text{HoS} \\ \text{Days} \end{pmatrix},$$

where Loc is the current location of the driver, $Equip$ is the type of equipment he is driving, $Home$ is the location of where he lives, HoS stands for "hours of service"

which captures how many hours he has worked today, and $Days$ is the number of days he has been away from home. Assume we have an observation of the amount of money the driver makes on day n, where we might write

$$\hat{\mu}^n \begin{pmatrix} \text{Loc} \\ \text{Equip} \\ \text{Home} \\ \text{DOThrs} \\ \text{Days} \end{pmatrix} = \text{Money earned on day } n .$$

We can create estimates at three different levels of aggregation using

$$\bar{\mu}^{(1,n)} \begin{pmatrix} \text{Loc} \\ \text{Equip} \\ \text{Home} \end{pmatrix} = (1-\alpha^{(1)}_{x,n-1})\bar{\mu}^{(1,n-1)} \begin{pmatrix} \text{Loc} \\ \text{Equip} \\ \text{Home} \end{pmatrix} + \alpha^{(1)}_{x,n-1}\hat{\mu}^n \begin{pmatrix} \text{Loc} \\ \text{Equip} \\ \text{Home} \\ \text{HoS} \\ \text{Days} \end{pmatrix},$$

$$\bar{\mu}^{(2,n)} \begin{pmatrix} \text{Loc} \\ \text{Equip} \end{pmatrix} = (1-\alpha^{(2)}_{x,n-1})\bar{\mu}^{(2,n-1)} \begin{pmatrix} \text{Loc} \\ \text{Equip} \end{pmatrix} + \alpha^{(2)}_{x,n-1}\hat{\mu}^n \begin{pmatrix} \text{Loc} \\ \text{Equip} \\ \text{Home} \\ \text{HoS} \\ \text{Days} \end{pmatrix},$$

$$\bar{\mu}^{(3,n)} \begin{pmatrix} \text{Loc} \end{pmatrix} = (1-\alpha^{(3)}_{x,n-1})\bar{\mu}^{(3,n-1)} \begin{pmatrix} \text{Loc} \end{pmatrix} + \alpha^{(3)}_{x,n-1}\hat{\mu}^n \begin{pmatrix} \text{Loc} \\ \text{Equip} \\ \text{Home} \\ \text{HoS} \\ \text{Days} \end{pmatrix}.$$

We need to estimate the variance of our estimate $\bar{\mu}^{(g,n)}_x$. To do this we are going to need

$(s^{(g,n)}_x)^2$ = The bias-corrected sum of squares in the estimate of $\bar{\mu}^{(g,n-1)}_x$.

$(s^{(g,n)}_x)^2$ is the estimate of the variance of the observations $\hat{\mu}_x$ when we observe the function at $x = x^n$ which aggregates to $x^{(g)}$ (that is, $G^g(x^n) = x^{(g)}$).

To compute $(s^{(g,n)}_x)^2$, we have to estimate the total variation in the estimate of $\bar{\mu}^{(g,n-1)}_x$, and then separate out the bias. We first define

$$\begin{aligned} \bar{\nu}^{(g,n)}_x &= \text{The total variation in the estimate } \bar{\mu}^{(g,n-1)}_x. \\ \bar{\delta}^{(g,n)}_x &= \text{The estimated bias in } \bar{\mu}^{(g,n-1)}_x \text{ due to aggregation.} \end{aligned}$$

We compute $\bar{\nu}^{(g,n)}_x$ using

$$\bar{\nu}^{(g,n)}_x = (1-\eta_{n-1})\bar{\nu}^{(g,n-1)}_x + \eta_{n-1}(\bar{\mu}^{(g,n-1)}_x - \hat{\mu}^n_x)^2,$$

where η_{n-1} follows some stepsize rule (which is typically just a constant such as 0.1). We refer to $\bar{\nu}^{(g,n)}_x$ as the total variation because it captures deviations that arise both due to measurement noise (the randomness when we compute $\hat{\mu}^n_x$) and bias (since $\bar{\mu}^{(g,n-1)}_x$ is a biased estimate of the mean of $\hat{\mu}^n_x$).

We next estimate the bias using

$$\bar{\delta}_x^{(g,n)} = \bar{\mu}_x^{(g,n)} - \bar{\mu}_x^{(0,n)}. \tag{9.6}$$

We then separate out the effect of bias to obtain

$$(s_x^{(g,n)})^2 = \begin{cases} 0 & n = 1, \\ \frac{\bar{\nu}_x^{(g,n)} - (\bar{\delta}_x^{(g,n)})^2}{1+\lambda^{n-1}} & n = 2, 3, \ldots, \end{cases} \tag{9.7}$$

We are not done. What we are really looking for is the variance of $\bar{\mu}_x^{(g,n-1)}$, which we define as

$(\bar{\sigma}_x^{(g,n)})^2 =$ The estimate of the variance of $\bar{\mu}_x^{(g,n)}$.

This is given by the formula

$$\begin{aligned} (\bar{\sigma}_x^{(g,n)})^2 &= \mathit{Var}[\bar{\mu}_x^{(g,n)}] \\ &= \lambda_x^{(g,n)} (s_x^{(g,n)})^2, \end{aligned} \tag{9.8}$$

where $\lambda_x^{(g,n)}$ can be computed from the recursion

$$\lambda_x^{(g,n)} = \begin{cases} (\alpha_{x,n-1}^{(g)})^2, & n = 1, \\ (1 - \alpha_{x,n-1}^{(g)})^2 \lambda_x^{(g,n-1)} + (\alpha_{x,n-1}^{(g)})^2, & n > 1. \end{cases}$$

The derivation of this formula is given in section 9.4.1 at the end of the chapter.

The different estimates are illustrated using the function depicted in figure 9.5, where we show a function at two levels of aggregation, allowing us to depict the different estimates and bias. Note that bias is very dependent on the decision x, since our underlying function may be smooth (producing very little bias at the aggregate level), or sloped, in which case the bias may be quite large or, in some places, very small.

9.3.2 Combining multiple levels of aggregation

Our next challenge is to create an estimate $\bar{\mu}_x^n$ of μ_x using the data we have collected, which we are going to do using our family of estimates $\bar{\mu}_x^{g,n}$ over the different levels of aggregation. The most natural strategy (we believe) is to take a weighted average.

$$\bar{\mu}_x^n = \sum_{g \in \mathcal{G}} w^{(g)} \bar{\mu}_x^{(g)}, \tag{9.9}$$

where $w^{(g)}$ is the weight applied to the g^{th} level of aggregation.

While this estimate looks appealing, the weights really need to depend on x since the relative accuracy of each level of aggregation will generally depend on x. For this reason, we write our weighted estimate as

$$\bar{\mu}_x^n = \sum_{g \in \mathcal{G}} w_x^{(g)} \bar{\mu}_x^{(g,n)}. \tag{9.10}$$

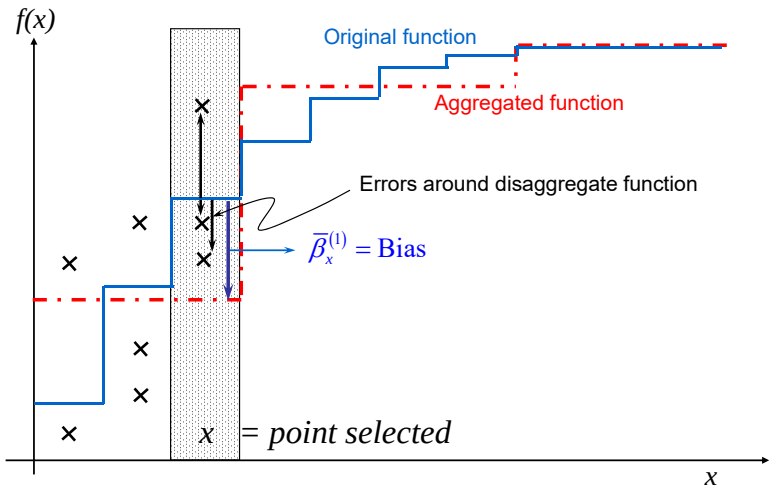


Figure 9.5 Illustration of a disaggregate function, an aggregated approximation and a set of samples. For a particular state s, we show the estimate and the bias.

If the estimates $\bar{\mu}_x^{(g,n)}$ were unbiased, we could show that the best weights are proportional to the estimates of the variance of $\bar{\mu}_x^{(g,n)}$, which is to say

$$w_x^{(g)} \propto \frac{1}{(\bar{\sigma}_x^{(g,n)})^2}.$$

Since the weights should sum to one, we obtain

$$w_x^{(g)} = \left(\frac{1}{(\bar{\sigma}_x^{(g,n)})^2}\right)\left(\sum_{g\in\mathcal{G}}\frac{1}{(\bar{\sigma}_x^{(g,n)})^2}\right)^{-1}. \tag{9.11}$$

The problem here is that the estimates $\bar{\mu}_x^{(g,n)}$ are clearly not unbiased, but this is easy to fix using the calculations we presented in the previous section. Instead of using the variance, we instead use the variance plus bias squared, where the bias is given by $\bar{\delta}_x^{(g,n)}$. This gives us the weighting formula

$$w_x^{(g,n)} \propto \frac{1}{\left((\bar{\sigma}_x^{(g,n)})^2 + \left(\bar{\delta}_x^{(g,n)}\right)^2\right)} \tag{9.12}$$

Introducing the normalization constant then gives us

$$w_x^{(g,n)} = \frac{1}{\left((\bar{\sigma}_x^{(g,n)})^2 + \left(\bar{\delta}_x^{(g,n)}\right)^2\right)}\left(\sum_{g'\in\mathcal{G}}\frac{1}{\left((\bar{\sigma}_x^{(g',n)})^2 + \left(\bar{\delta}_x^{(g',n)}\right)^2\right)}\right)^{-1}. \tag{9.13}$$

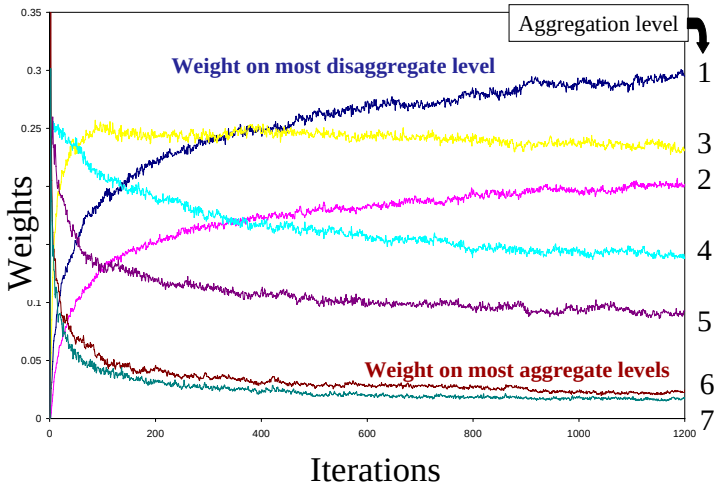


Figure 9.6 Average weight (across all states) for each level of aggregation using equation (9.13).

As might be anticipated, the weights tend to concentrate on the most aggregate levels in the early estimations when there is a lot of noise, which limits our ability to even estimate a bias. As more data comes in, we can put more weight on the more disaggregate levels, although this is not universally true across all experimental decisions x. This behavior can be seen quite clearly in figure 9.6, which plots the average weight for each level of aggregation.

9.3.3 Hierarchical knowledge gradient

We now address the challenge of computing the knowledge gradient, which we restate as

$$\nu_x^{KG}(S^n) = \mathbb{E}\left[\max_{x' \in \mathcal{X}} \bar{\mu}_{x'}^{n+1}(x) | S^n\right] - \max_{x' \in \mathcal{X}} \bar{\mu}_{x'}^n. \tag{9.14}$$

The key to handling our hierarchical belief model in the knowledge gradient is taking advantage of its structure. We begin by repeating the updating equations for the mean and precision at level of aggregation g:

$$\bar{\mu}_x^{(g,n+1)} = \frac{\beta_x^{(g,n)} \bar{\mu}_x^{(g,n)} + \beta_x^W \hat{\mu}_x^{n+1}}{\beta_x^{(g,n)} + \beta_x^W}, \tag{9.15}$$

$$\beta_x^{(g,n+1)} = \beta_x^{(g,n)} + \beta_x^W, \tag{9.16}$$

For reasons that become clear later, we rewrite equation (9.15) in the form

$$\bar{\mu}_x^{(g,n+1)} = \bar{\mu}_x^{(g,n)} + \frac{\beta_x^W}{\beta_x^{(g,n)} + \beta_x^W}\left(\hat{\mu}_x^{n+1} - \bar{\mu}_x^{(g,n)}\right).$$

We further rewrite this equation by splitting the second term using

$$\bar{\mu}_x^{(g,n+1)} = \bar{\mu}_x^{(g,n)} + \frac{\beta_x^W}{\beta_x^{(g,n)} + \beta_x^W}\left(\hat{\mu}_x^{n+1} - \bar{\mu}_x^n\right) + \frac{\beta_x^W}{\beta_x^{(g,n)} + \beta_x^W}\left(\bar{\mu}_x^n - \bar{\mu}_x^{(g,n)}\right).$$

We are next going to rewrite our updating formula as we did in section 4.5 using the standard normal variable Z,

$$\bar{\mu}_x^{(g,n+1)} = \bar{\mu}_x^{(g,n)} + \frac{\beta_x^W}{\beta_x^{(g,n)} + \beta_x^W}\left(\bar{\mu}_x^n - \bar{\mu}_x^{(g,n)}\right) + \tilde{\sigma}_x^{(g,n)} Z, \tag{9.17}$$

where, with some work, we can show that the predictive variance is given by

$$\tilde{\sigma}_x^{(g,n)} = \frac{\beta_x^W \sqrt{\frac{1}{\beta_x^{(g,n)}} + \frac{1}{\beta_x^W}}}{\beta_x^{(g,n)} + \beta_x^W}. \tag{9.18}$$

We next define the sets

$\mathcal{G}(x, x')$ Set of all aggregation levels that the alternatives x and x' have in common, with $\mathcal{G}(x, x') \subseteq \mathcal{G}$. In the example in figure 9.4 we have $\mathcal{G}(2, 3) = \{1, 2\}$, since $x = 2$ and $x' = 3$ have common points at aggregation levels 1 and 2.

$\mathcal{X}^{(g)}(x)$ Set of all alternatives that share the same aggregated alternative $G^{(g)}(x)$ at the g^{th} aggregation level, with $\mathcal{X}^{(g)}(x) \subseteq \mathcal{X}$. In the example in figure 9.4 we have $\mathcal{X}^1(4) = \{4, 5, 6\}$.

We need to capture the impact of a decision to run experiment x on all the estimates $\mu_{x'}$ for $x' \in \mathcal{X}$. With the hierarchical learning logic, running one experiment x can affect the estimates of $\mu_{x'}$ for all the experiments. This will happen whenever x and x' share an aggregation level. This might include everything if the highest level of aggregation is simply a constant baseline over the entire surface. To help with this process, we break our original weighted estimate

$$\bar{\mu}_x^n = \sum_{g \in \mathcal{G}} w_x^{(g)} \bar{\mu}_x^{(g,n)},$$

into two parts:

$$\bar{\mu}_{x'}^{n+1} = \sum_{g \notin \mathcal{G}(x',x)} w_{x'}^{(g,n+1)} \bar{\mu}_{x'}^{(g,n+1)} + \sum_{g \in \mathcal{G}(x',x)} w_{x'}^{(g,n+1)} \bar{\mu}_x^{(g,n+1)}.$$

After substitution of (9.17) and rearranging gives us

$$\begin{aligned} \bar{\mu}_{x'}^{n+1} &= \sum_{g \in \mathcal{G}} w_{x'}^{(g,n+1)} \bar{\mu}_{x'}^{(g,n)} + \sum_{g \in \mathcal{G}(x',x)} w_{x'}^{(g,n+1)} \frac{\beta_x^W}{\beta_x^{(g,n)} + \beta_x^W}\left(\bar{\mu}_x^n - \bar{\mu}_x^{(g,n)}\right) \\ &\quad + Z \sum_{g \in \mathcal{G}(x',x)} w_{x'}^{(g,n+1)} \tilde{\sigma}_x^{(g,n)}. \end{aligned} \tag{9.19}$$

When we run experiment x and observe $\hat{\mu}_x^n$, we will of course have to update $\bar{\mu}_x^{(g,n)}$ for all aggregation levels g. It is tempting to assume that the weights $w_x^{(g)}$ remain constant, but we found that this did not work, primarily in the early iterations. For example, it is natural to initialize $w_x^{(G)} = 1$ for the highest level of aggregation, but then we have to understand that an experiment may change this, since otherwise the value of information may be zero. For this reason, we have to predict the updating weights from running experiment x. We approximate the weights using

$$\overline{w}_{x'}^{(g,n)}(x) = \frac{\left(\left(\beta_{x'}^{(g,n)} + I_{x',x}^{(g)}\beta_{x'}^{W}\right)^{-1} + \left(\beta_{x'}^{(g,n)}\right)^2\right)^{-1}}{\sum_{g'\in\mathcal{G}}\left(\left(\beta_{x'}^{g',n} + I_{x',x}^{g'}\beta_{x'}^{W}\right)^{-1} + \left(\beta_{x'}^{g',n}\right)^2\right)^{-1}}, \tag{9.20}$$

where

$$I_{x',x}^{(g)} = \begin{cases} 1 & \text{if } g \in \mathcal{G}(x',x) \\ 0 & \text{otherwise} \end{cases}.$$

Combining (9.14) with (9.19) and (9.20), gives us the following formula for the knowledge gradient

$$\nu_x^{KG}(S^n) = \mathbb{E}\left[\max_{x'\in\mathcal{X}} a_{x'}^n(x) + b_{x'}^n(x)Z|S^n\right] - \max_{x'\in\mathcal{X}}\bar{\mu}_{x'}^n, \tag{9.21}$$

where

$$a_{x'}^n(x) = \sum_{g\in\mathcal{G}}\overline{w}_{x'}^{(g,n)}(x)\bar{\mu}_{x'}^{(g,n)} + \sum_{g\in\mathcal{G}(x',x)}\overline{w}_{x'}^{(g,n)}(x)\frac{\beta_x^W}{\beta_x^{(g,n)}+\beta_x^W}\left(\bar{\mu}_x^n - \bar{\mu}_x^{(g,n)}\right), \tag{9.22}$$

$$b_{x'}^n(x) = \sum_{g\in\mathcal{G}(x',x)}\overline{w}_{x'}^{(g,n)}(x)\tilde{\sigma}_x^{(g,n)}. \tag{9.23}$$

We now use the procedure described in section 4.5 for the knowledge gradient for correlated beliefs, where we have to identify the lines $a_{x'}^n + b_{x'}^n z$ which are dominated by the others. We refer to non-dominated lines by $\tilde{a}_{x'}^n + \tilde{b}_{x'}^n z$ over a reduced set of alternatives that are now numbered from $1,\ldots,\tilde{M}$, giving us

$$\nu_x^{KG,n} = \sum_{i=1,\ldots,\tilde{M}-1}\left(\tilde{b}_{i+1}^n(x) - \tilde{b}_i^n(x)\right) f\left(-\left|\frac{\tilde{a}_i^n(x) - \tilde{a}_{i+1}^n(x)}{\tilde{b}_{i+1}^n(x) - \tilde{b}_i^n(x)}\right|\right), \tag{9.24}$$

where (as we saw before in chapter 4),

$$f(\zeta) = \zeta\Phi(\zeta) + \phi(\zeta),$$

where $\Phi(\zeta)$ is the cumulative normal distribution and $\phi(\zeta)$ is the standard normal density, given respectively by

$$\Phi(\zeta) = \int_{-\infty}^{\zeta}\phi(z)dz,$$

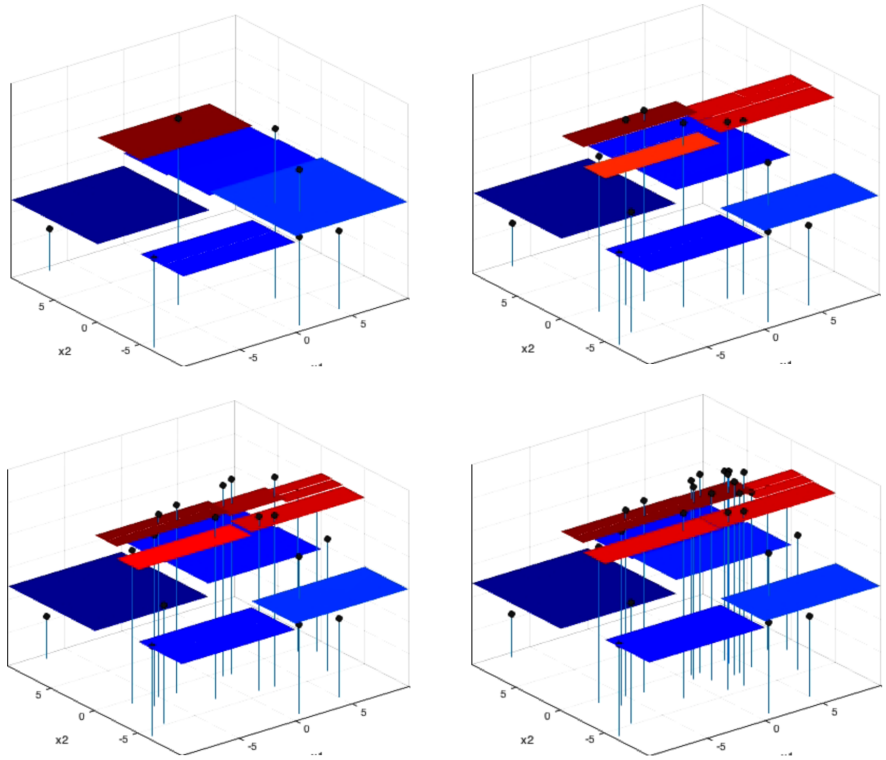


Figure 9.7 Sequence of estimates of a function using the hierarchical knowledge gradient policy, showing initial exploration to a more focused search as it identifies the optimum.

and

$$\phi(\zeta) = \frac{1}{\sqrt{2\pi}} e^{-\frac{\zeta^2}{2}}.$$

Thus, the knowledge gradient policy for hierarchical beliefs is comparable to the KG policy for correlated beliefs.

Figure 9.7 demonstrates the hierarchical knowledge gradient as it learns a smooth function with a single maximum. The first quadrant (upper left) shows initial exploration, steadily narrowing (left to right) to the region of the optimum as it learns to exclude certain regions.

Hierarchical belief models are fairly powerful for strategy for choosing among complex alternatives (such as people, movies and drugs). The speed with which it learns, of course, depends completely on the complexity of the underlying function. For example, if $\mathcal{X}$ is a large discrete set (imagine, for example, the set of all movies that can be watched on the internet, or the set of all possible molecular combinations for a drug), and assume that there is no relationship at all between the value of two different alternatives μ_x and $\mu_{x;}$. If this actually were the situation, no learning policy would perform well without an exceptionally large budget.

Fortunately, most real applications (especially large ones) have a lot of structure. If the belief model captures this structure, then a good learning policy can do quite well. The power of the Bayesian knowledge gradient, of course, is that it assumes that we start the process using some prior knowledge, which we want to exploit.

9.4 WHY DOES IT WORK?

From OUU book - needs rewriting

9.4.1 Computing bias and variance

Before we present our methods for hierarchical aggregation, we are going to need some basic results on bias and variance in statistical estimation. Assume we are trying to estimate a true but unknown parameter μ which we can observe, but we have to deal with both bias δ and noise ε, which we write as

$$\hat{\mu}^n = \mu + \delta + \varepsilon^n. \tag{9.25}$$

Both the mean μ and bias δ are unknown, but we are going to assume that we have some way to make a noisy estimate of the bias that we are going to call $\hat{\delta}^n$. Later we are going to provide examples of how to get estimates of δ.

Now let $\bar{\mu}^n$ be our estimate of μ after n observations. We will use the following recursive formula for $\bar{\mu}^n$

$$\bar{\mu}^n = (1 - \alpha_{n-1})\bar{\mu}^{n-1} + \alpha_{n-1}\hat{\mu}^n.$$

We are interested in estimating the variance of $\bar{\mu}^n$ and its bias δ^n. We start by computing the variance of $\bar{\mu}^n$. We assume that our observations of μ can be represented using equation (9.25), where $\mathbb{E}\varepsilon^n = 0$ and $\mathit{Var}[\varepsilon^n] = \sigma^2$. With this model, we can compute the variance of $\bar{\mu}^n$ using

$$\mathit{Var}[\bar{\mu}^n] = \lambda^n \sigma^2, \tag{9.26}$$

where λ^n can be computed from the simple recursion

$$\lambda^n = \begin{cases} (\alpha_{n-1})^2, & n = 1, \\ (1 - \alpha_{n-1})^2 \lambda^{n-1} + (\alpha_{n-1})^2, & n > 1. \end{cases} \tag{9.27}$$

To see this, we start with $n = 1$. For a given (deterministic) initial estimate $\bar{\mu}^0$, we first observe that the variance of $\bar{\mu}^1$ is given by

$$\begin{aligned} \mathit{Var}[\bar{\mu}^1] &= \mathit{Var}[(1 - \alpha_0)\bar{\mu}^0 + \alpha_0\hat{\mu}^1] \\ &= (\alpha_0)^2 \mathit{Var}[\hat{\mu}^1] \\ &= (\alpha_0)^2 \sigma^2. \end{aligned}$$

For $\bar{\mu}^n$ for $n > 1$, we use a proof by induction. Assume that $Var[\bar{\mu}^{n-1}] = \lambda^{n-1}\sigma^2$. Then, since $\bar{\mu}^{n-1}$ and $\hat{\mu}^n$ are independent, we find

$$\begin{aligned} Var[\bar{\mu}^n] &= Var\left[(1-\alpha_{n-1})\bar{\mu}^{n-1} + \alpha_{n-1}\hat{\mu}^n\right] \\ &= (1-\alpha_{n-1})^2 Var\left[\bar{\mu}^{n-1}\right] + (\alpha_{n-1})^2 Var[\hat{\mu}^n] \\ &= (1-\alpha_{n-1})^2\lambda^{n-1}\sigma^2 + (\alpha_{n-1})^2\sigma^2 \qquad (9.28) \\ &= \lambda^n\sigma^2. \qquad (9.29) \end{aligned}$$

Equation (9.28) is true by assumption (in our induction proof), while equation (9.29) establishes the recursion in equation (9.27). This gives us the variance, assuming of course that σ^2 is known.

Using our assumption that we have access to a noisy estimate of the bias given by $\hat{\delta}^n$, we can compute the mean-squared error using

$$\mathbb{E}\left[\left(\bar{\mu}^{n-1} - \mu(n)\right)^2\right] = \lambda^{n-1}\sigma^2 + (\hat{\beta}^n)^2. \qquad (9.30)$$

See exercise **??** to prove this. This formula gives the variance around the known mean, $\bar{\mu}^n$. For our purposes, it is also useful to have the variance around the observations $\hat{\mu}^n$. Let

$$\nu^n = \mathbb{E}\left[\left(\bar{\mu}^{n-1} - \hat{\mu}^n\right)^2\right]$$

be the mean squared error (including noise and bias) between the current estimate $\bar{\mu}^{n-1}$ and the observation $\hat{\mu}^n$. It is possible to show that (see exercise **??**)

$$\nu^n = (1+\lambda^{n-1})\sigma^2 + (\beta^n)^2, \qquad (9.31)$$

where λ^n is computed using (9.27).

In practice, we do not know σ^2, and we certainly do not know the bias β. As a result, we have to estimate both parameters from our data. We begin by providing an estimate of the bias using

$$\bar{\delta}^n = (1-\eta_{n-1})\bar{\delta}^{n-1} + \eta_{n-1}\hat{\delta}^n,$$

where η_{n-1} is a (typically simple) stepsize rule used for estimating the bias and variance. As a general rule, we should pick a stepsize for η_{n-1} which produces larger stepsizes than α_{n-1} because we are more interested in tracking the true signal than producing an estimate with a low variance. We have found that a constant stepsize such as .10 works quite well on a wide range of problems, but if precise convergence is needed, it is necessary to use a rule where the stepsize goes to zero such as the harmonic stepsize rule (equation (**??**)).

To estimate the variance, we begin by finding an estimate of the total variation ν^n. Let $\bar{\nu}^n$ be the estimate of the total variance which we might compute using

$$\bar{\nu}^n = (1-\eta_{n-1})\bar{\nu}^{n-1} + \eta_{n-1}(\bar{\mu}^{n-1} - \hat{\mu}^n)^2.$$

Using $\bar{\nu}^n$ as our estimate of the total variance, we can compute an estimate of σ^2 using

$$(\bar{\sigma}^n)^2 = \frac{\bar{\nu}^n - (\bar{\delta}^n)^2}{1+\lambda^{n-1}}.$$

We can use $(\bar{\sigma}^n)^2$ in equation (9.26) to obtain an estimate of the variance of $\bar{\mu}^n$.

If we are doing true averaging (as would occur if we use a stepsize of $1/n$), we can get a more precise estimate of the variance for small samples by using the recursive form of the small sample formula for the variance

$$(\hat{\sigma}^2)^n = \frac{n-2}{n-1}(\hat{\sigma}^2)^{n-1} + \frac{1}{n}(\bar{\mu}^{n-1} - \hat{\mu}^n)^2. \tag{9.32}$$

The quantity $(\hat{\sigma}^2)^n$ is an estimate of the variance of $\hat{\mu}^n$. The variance of our estimate $\bar{\mu}^n$ is computed using

$$(\bar{\sigma}^2)^n = \frac{1}{n}(\hat{\sigma}^2)^n.$$

We are going to draw on these results in two settings, which are both distinguished by how estimates of the bias δ^n are computed:

Hierarchical aggregation We are going to estimate a function at different levels of aggregation. We can assume that the estimate of the function at the most disaggregate level is noisy but unbiased, and then let the difference between the function at some level of aggregation and the function at the most disaggregate level as an estimate of the bias.

Transient functions Later, we are going to use these results to approximate value functions. It is the nature of algorithms for estimating value functions that the underlying process varies over time (see see this most clearly in chapter **??**). In this setting, we are making observations from a truth that is changing over time, which introduces a bias.

9.5 BIBLIOGRAPHIC NOTES

Section **??** - See also Miller (2002) for additional applications of subset selection in statistics, and Horrace et al. (2008) for applications in economics.

Section 9.2.4 - The energy portfolio application is also considered by Ryzhov & Powell (2009*c*) and Ryzhov & Powell (2009*b*). The latter reference also first proposes the Monte Carlo KG policy. The recursive updating equation for the square root matrix can be found e.g. in Kaminski et al. (1971).

PROBLEMS

9.1 There are six primary drugs that can be used to treat diabetes, and these are often prescribed in groups of three at a time. There are 20 ways of choosing three drugs from a set of six (known as a "cocktail"), assuming all possible combinations make sense. Enumerate these 20 cocktails, and create a covariance matrix Σ where entry $\Sigma_{ij} = \sigma^2 N_{ij}$, where N_{ij} is the number of drugs in common between drug cocktail i and drug cocktail j (so, $0 \leq N_{ij} \leq 3$). Let $\sigma^2 = .35^2$ be the variance in our prior

distribution of belief about the effect of a drug cocktail. Finally, let $\sigma_\epsilon^2 = .55^2$ be the variance of an observation.

Assume that our prior on the blood sugar level produced by each cocktail is the same and is equal to 5.0.

a) Randomly generate a truth from this prior for each cocktail. Create the covariance matrix and use the knowledge gradient for correlated beliefs to search for the best cocktail using $N = 20$ experiments. You may use the MATLAB code (first introduced in Chapter 4) that can be downloaded from.

`http://optimallearning.princeton.edu/exercises/KGCorrBeliefs.m`

An example illustration of the KGCB algorithm is given in

`http://optimallearning.princeton.edu/exercises/KGCorrBeliefsEx.m`

b) Now repeat (a) for 100 different truths, and summarize how well you discover each truth using 20 observations.

c) Next we are going to use the methods described in this chapter to solve a much larger problem, but we are going to use them on this small test problem. Repeat (a), but this time you are going to randomly sample 5 out of the 20 combinations, and limit your calculation of the KG factor to these three (the easiest way to modify the code is to compute the knowledge gradient for all 20 as you are doing, but then choose a subset of 5, and finally choose the drug cocktail with the best knowledge gradient out of these five). Perform $N = 100$ samples and compare the performance to the results you obtained when you computed the knowledge gradient for all the alternatives.

d) Finally, repeat (d) for all 100 truths and report on the average performance.

9.2 Verify that 9.4 produces a valid square root matrix, that is, $C^{n+1}\left(C^{n+1}\right)^T = \Sigma^{n+1}$.

CHAPTER 10

OPTIMIZING A SCALAR FUNCTION

Optimizing scalar functions arises in a variety of settings, such as choosing the price of a product, the amount of memory in a computer, the temperature of a chemical process or the diameter of the tube in an aerosol gun. It can even be used for optimizing a tunable parameter in a learning policy! Important special cases are functions which are unimodular, as depicted in Figure 10.1, where there is a single local maximum, or concave. In this chapter, we review some specialized algorithms designed for this particular problem class.

We begin our presentation using an unusual problem setting for this volume, which is an unknown but deterministic function which can be measured perfectly. We then transition back to our more familiar setting of noisy functions.

10.1 DETERMINISTIC EXPERIMENTS

We are going to consider three settings where we wish to find the best estimate of the optimum of a deterministic, unimodular function within a given experimental budget. The settings include

- Differentiable functions.
- Nondifferentiable functions, with a finite experimental budget.

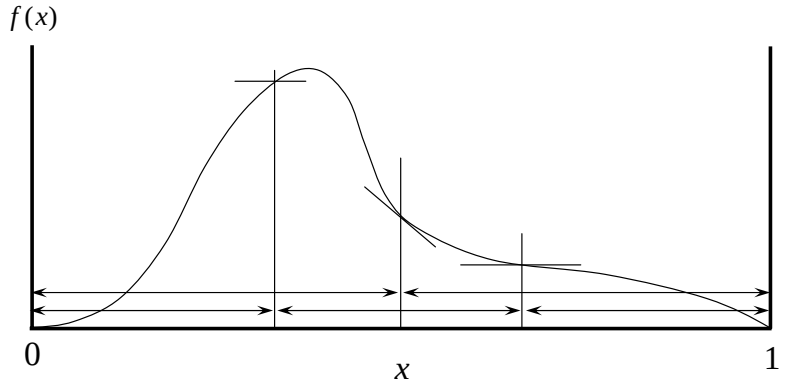


Figure 10.1 A unimodular function, showing the slope at the midpoint, and the height of the function at 1/3 and 2/3.

- Nondifferentiable function with an unlimited experimental budget, but where we want the fastest possible learning rate.

10.1.1 Differentiable functions

We begin by assuming that $f(x)$ is differentiable and unimodular (that is, it has a single local maximum). We assume the region we are searching is bounded, so we can scale the search region to be between 0 and 1. Initially, we assume the optimum x^* can be anywhere in this interval with equal likelihood. If we measure the derivative $f'(x)$ at $x = 0.5$, we can observe whether $f'(.5) > 0$ or $f'(.5) < 0$. If the derivative is negative (as shown in Figure 10.1), then we know that $0 \leq x^* \leq .5$. We can eliminate the portion of the interval greater than .5. This means we can redefine the entire problem on the interval $(0, .5)$ and repeat the process. Let ℓ^n be the length of the interval in which the optimum may lie after n iterations, where $\ell^0 = 1$. It is easy to see that $\ell^n = .5^n$. Under the assumptions of the problem, this is the fastest possible rate of reduction that we can achieve.

10.1.2 Nondifferentiable functions, finite budget

There are many problems where we cannot compute the derivative, but we can compute the function. For example, a transportation company may have a model that evaluates the on-time service when the fleet size is x. The company may vary x, re-running the model each time, but the model may not provide a derivative of the performance measure with respect to x. Instead, we have to search over a set of discrete values for x, just as we have done throughout this volume.

Normally we would create some sort of belief model, but now we are going to use the unimodular structure (along with our ability to perfectly observe the function). For example, we can try $x^1 = .2$ and $x^2 = .8$. If $f(.2) > f(.8)$, then we can eliminate the

region $(.8, 1)$ from further consideration. Of course, this only eliminates 20 percent of the interval. Our goal now is to eliminate the largest possible interval.

Imagine that we have an experimental budget of $N = 2$ experiments. In this case, the best strategy is to measure $x^1 = .5^-$ and $x^2 = .5^+$, by which we mean slightly less than .5 and slightly more than .5. Comparing $f(.5^-)$ and $f(.5^+)$ allows us to effectively eliminate half the interval, just as we did when we could compute a derivative (we are basically computing a numerical derivative).

Next consider the case where $N = 3$. If we first measure $x^1 = 1/3$ and $x^2 = 2/3$, we eliminate either $(0, 1/3)$ or $(2/3, 1)$. Assume that we eliminate the upper interval (as we did in Figure 10.1). Now we are left with the interval $(0, 2/3)$, but we have already run an experiment at the midpoint $x = 1/3$. We are going to use our final experiment at a point slightly above or below $1/3$ to determine the final interval of uncertainty, which will have width $1/3$. This last function evaluation will allow us to eliminate half of the remaining interval, giving us a region of width 1/6 where the optimum must lie.

We repeat this exercise one more time for $N = 4$, but now we are going to assume that the optimum is at $x = 0$ (but we have to construct our experiments without knowing this), so that we are always eliminating the upper part of the remaining interval. If we measure $x^1 = 2/5$ and $x^2 = 3/5$, we would eliminate $(3/5, 1)$, and we are left with the interval $(0, 3/5)$ with an experiment at $2/5$. Conveniently, this is at the two-thirds point of the interval $(0, 3/5)$, with two remaining experiments. If we measure $x = 1/5$, we are now in the same situation we were when we started with $N = 3$, but on an interval of width $3/5$. We eliminate the upper interval, leaving us with the interval $(0, 2/5)$ and an experiment at the midpoint $1/5$. We use our final experiment at a point slightly higher or lower than the experiment at $1/5$, giving us a final interval of $1/5$.

Now compare this result to what we would have obtained if we had used the bisection search with numerical derivatives. If $N = 2$, we would have measured just above and below .5, giving us a final interval of width .5. If $N = 4$, we would have done this twice, giving us an interval of width $.5^2 = .25$, which is greater than the interval we obtained of $1/5 = .2$. How did we accomplish this?

There is a pattern in this logic. When we eliminated the interval $(3/5, 1)$, we were left with the interval $(0, 3/5)$ and an experiment at $2/5$. Rescaling all the numbers so that the interval is of length 1, we get an interval of $(0, 1)$ with an experiment at $2/3$. If we eliminated the lower part of the interval $(0, 2/5)$, we would be left with the interval $(2/5, 1)$ (which still has length $3/5$) and an experiment at $3/5$. Rescaling gives us an interval of length 1, and an experiment at $1/3$. Either way, we end up with an experiment we would have made anyway if we only had 3 experiments.

We can formalize the algorithm as follows. Let $f^1 < f^2 < f^3 < \ldots$ be an increasing sequence of integers. If we are allowed three experiments, assume we first measure f^1/f^3 and f^2/f^3. If we eliminate the upper part of the interval, we are left with an interval $(0, f^2/f^3)$. If we eliminate the lower part of the interval, we are left with $(f^1/f^3, 1)$. We would like the width of these intervals to be the same, so we are

going to require that

$$\frac{f^2}{f^3} = 1 - \frac{f^1}{f^3},$$

or, rearranging slightly,

$$f^3 = f^1 + f^2.$$

Similarly if we have run four experiments, we first measure f^2/f^4 and f^3/f^4. Repeating the exercise above, rejecting the upper range leaves us with an interval of f^3/f^4, while rejecting the lower range leaves us with the interval $(1 - f^2/f^4)$. Again equating these gives us

$$\frac{f^3}{f^4} = 1 - \frac{f^2}{f^4},$$

or

$$f^4 = f^3 + f^2.$$

Using proof by extrapolation, we see that if we are making N experiments, we want

$$f^N = f^{N-1} + f^{N-2}. \tag{10.1}$$

Furthermore, this has to be true for all $n < N$. Equation (10.2) defines what is known as the Fibonacci sequence, comprising the numbers $(1, 1, 2, 3, 5, 8, \ldots)$, where we initialize the sequence using $f^0 = 1$, and then let $f^1 = 1, f^2 = 2, f^3 = 3, \ldots$.

It is possible to show that the Fibonacci sequence produces an optimal search sequence for a finite experimental budget (under the assumptions of our problem). By this we mean that for a given value of N, no other method will produce a smaller final interval in which the optimal solution may lie. This means that our search is optimal, because we learn the most within our budget.

10.1.3 Nondifferentiable functions, infinite budget

So what if our budget is unlimited? We are going to start by hypothesizing that in the limit, we are going to measure the interval at two points, which we are going to denote by $1 - r$ and r, where $.5 < r < 1$. If we measure $1 - r$ and r, and then eliminate the upper interval, we are left with the interval $(0, r)$. Now assume we are going to measure the same points within this interval, which would occur at $r(1 - r)$ and r^2. We want the larger of these two points to coincide with the smaller of the two experiments in the original interval, so that at each iteration, we are measuring only one additional point (as we did with the Fibonacci search). This means that we require

$$r^2 = 1 - r,$$

or

$$r^2 + r - 1 = 0.$$

We solve this using the quadratic formula which gives the roots for the equation $ar^2 + br + c = 0$ as

$$r = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

Using only the positive part gives us the solution

$$\begin{aligned} r &= \frac{-1 + \sqrt{1+4}}{2} \\ &= \frac{-1 + \sqrt{5}}{2} \\ &= 0.618. \end{aligned}$$

The quantity $r = .618$ is known as the *golden section* or *golden ratio*. We note that we get the same result if we had eliminated the lower portion of the interval, since our experiments are naturally symmetric. We also note that if f^n is the nth Fibonacci number, then

$$\lim_{n\to\infty} \frac{f^{n-1}}{f^n} \to r = 0.618.$$

Thus, the golden section search is the limit of the Fibonacci search.

The Fibonacci search and the golden section search are both examples of optimal learning in that they give the fastest improvement in our knowledge, measured by the length of the interval where we think the optimum might be. The Fibonacci search guarantees the smallest possible interval after a fixed (and known) number of experiments. The golden section search gives the fastest *rate* of convergence for an algorithm that will be run an infinite number of times. To put it another way, we account for the effect that our next experiment will have on the length of the uncertain interval.

10.2 NOISY EXPERIMENTS

We now return to our more familiar setting where experimental observations of the function are noisy. We consider the stochastic version of bisection search, where we assume we have access to a noisy indicator that tells us whether we think the optimum is to the left or the right of our experiment. In generalizing the bisection search we make a particular assumption about the experimental noise. We explain this assumption by noting that the bisection search operates by separating two regions of the search space: the region to the left of x^*, and the region to the right. Measuring the derivative of the function $f(x)$ at a point x reveals in which part of the search space x belongs.

Even without a function $f(x)$, we can use a bisection search to solve any problem where we want to find the boundary between two regions, and where measuring a point reveals in which region it resides. In the error-free case, this revelation is always correct. In the noisy version of the problem we instead assume that the revelation is incorrect with a probability that is *known* and *constant*. While these conditions are

typically not going to be satisfied in practice, they provide an elegant search model and they are certainly more realistic than assuming the observation of the function is perfect.

This assumption of constancy would tend to be met in applications where the transition between regions is abrupt. As an example, if a city's water supply were contaminated with a dangerous chemical we would want to localize the extent of contamination as quickly as possible, and if the chemical did not dissolve well in water but instead tended to stay concentrated, we would find a situation with this abrupt transition between contaminated and uncontaminated water. In contrast, when we measure a smooth function with additive noise, noise tends to cause incorrect region assignments more frequently near the function's maximum. With this in mind, we should be careful in applying the stochastic bisection algorithm presented here to situations not meeting its assumptions.

10.2.1 The model

We formulate our problem mathematically by again supposing that we have a point x^* whose location is unknown beyond that it resides in the interval $[0, 1]$. The point x^* corresponds to the boundary between the two regions in $[0, 1]$, so in the water contamination example, the water in region $(x^*, 1]$ would be contaminated and the water in region $[0, x^*]$ would not. We adopt a Bayesian prior density p_0 on the location of this point x^*, where $p_0(x)$ gives the likelihood (density) that $x^* = x$. It could be uniform on $[0, 1]$ if we had little real information about its location, or it could be some more complicated distribution expressing a stronger belief. We then suppose that we are offered the opportunity to take a sequence of experiments $x^0, x^1, \ldots, x^N$ in the interval, and with each experiment x^n we get a noisy response $\hat{y}^{n+1}$ suggesting into which region x^n resides. Given x^n and x^*, this response will be independent of all other responses and will have the distribution

$$\hat{y}^{n+1} = \begin{cases} I_{\{x^* \le x^n\}} & \text{with probability } q, \\ I_{\{x^* > x^n\}} & \text{with probability } 1 - q. \end{cases}$$

We may also express this as $\mathbb{P}\{\hat{y}^{n+1} = I_{x^* \le x^n}\} = q$. Here q is the probability our experiment is correct, and we assume this probability is known and constant. Equivalently, $1 - q$ is the error rate in our experiments.

10.2.2 Finding the posterior distribution

These experiments alter our prior belief about the location of x^*, giving us a posterior belief, all according to Bayes' rule. We use the notation p^n to denote the posterior density at time n. To write the updating rule for p^n explicitly, we introduce two pieces of notation. Let F^n be the cumulative distribution function of the posterior at time n, by which we mean that, for any dummy variable x,

$$F^n(x) := \mathbb{P}\{x^* \le x \mid p^n\} = \int_{[0,x]} p^n(x)\, dx.$$

We may also think of $F^n(x)$ as giving the probability that x is in the region to the left of x^*, and in our water contamination example as giving the probability that the water at x is not contaminated. As our second piece of notation, let g be the function defined by $g(a,1) = a$ and $g(a,0) = 1-a$, where a will be a probability (such as the probability the true value is to the left or the right). Now we are ready to compute our updating rule.

Noting that nature's correct response to the experiment x^n would be $I_{\{x^* \le x^n\}}$, we write

$$\begin{aligned}\mathbb{P}\{\hat{y}^{n+1} = y \mid x^*, x^n\} &= \begin{cases} q & \text{if } y = I_{\{x^* \le x^n\}} \\ 1-q & \text{if } y \neq I_{\{x^* \le x^n\}} \end{cases} \\ &= g(q, I_{\left\{y = I_{\{x^* \le x^n\}}\right\}}).\end{aligned}$$

So, $g(q,1) = q$ corresponds to two cases: either $x^n < x*$, and we observe $\hat{y}^{n+1} = 1$ which correctly indicates that the optimum $x*$ is greater than our measured point; or $x^n > x*$, and $\hat{y}^{n+1} = 0$, which correctly indicates that the optimum x^* is less than our measured point. The outcome $g(q,0) = 1-q$ corresponds to the opposite of both of these cases.

We may also write

$$\begin{aligned}\mathbb{P}\{\hat{y}^{n+1} = y \mid p^n\} &= \mathbb{P}\{x^* \le x^n \mid p^n\}\mathbb{P}\{\hat{y}^{n+1} = y \mid x^* \le x^n\} \\ &\quad +\mathbb{P}\{x^* > x^n \mid p^n\}\mathbb{P}\{\hat{y}^{n+1} = y \mid x^* > x^n\} \\ &= F^n(x^n)g(q,y) + (1 - F^n(x^n))g(1-q,y) \\ &= g(qF^n(x^n) + (1-q)(1-F^n(x^n)), y),\end{aligned}$$

where the last line may be seen by considering the cases $y = 0$ and $y = 1$ separately. Fixing some dummy variable y, we may then use Bayes' rule and these two relations to write,

$$\begin{aligned}p^{n+1}(x)\,dx &= \mathbb{P}\{x^* \in dx \mid p^n, \hat{y}^{n+1} = y\} \\ &= \frac{\mathbb{P}\{\hat{y}^{n+1} = y \mid p^n, x^* = x\}\mathbb{P}\{x^* \in dx \mid p^n\}}{\mathbb{P}\{\hat{y}^{n+1} = y \mid p^n\}} \\ &= \frac{g\left(q, I_{\left\{y = I_{\{x \le x^n\}}\right\}}\right)}{g\left(qF^n(x^n) + (1-q)(1-F^n(x^n)), y\right)} p^n(x)\,dx.\end{aligned}$$

Substituting $\hat{y}^{n+1}$ for y shows that our updating rule is

$$p^{n+1}(x) = \frac{g\left(q, I_{\left\{\hat{y}^{n+1} = I_{\{x \le x^n\}}\right\}}\right)}{g\left(qF^n(x^n) + (1-q)(1-F^n(x^n)), \hat{y}^{n+1}\right)} p^n(x). \tag{10.2}$$

The essential content of this updating rule is as follows. Our observation, if correct, would tell us into what region x^n lies and would tell us whether x^* is to the left or right of the experiment x^n. Let us give the name "suggested region" to the region in which the observation, if correct, would indicate x^* resides. Since we know that the observation is only correct with probability q, we multiply the density in the

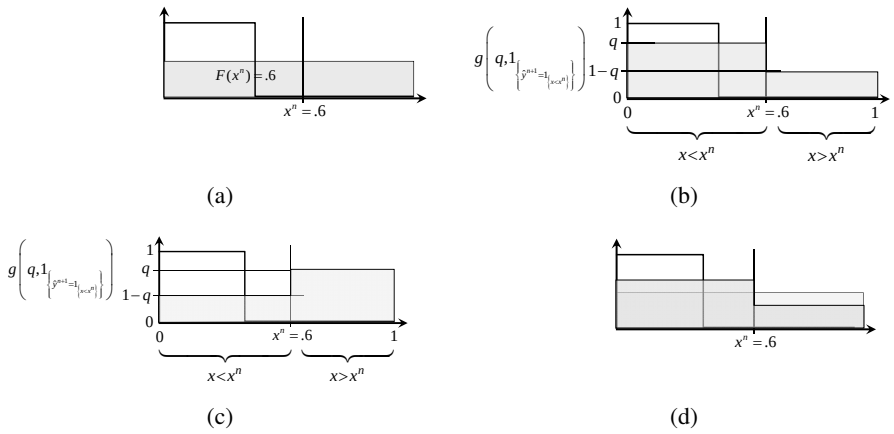


Figure 10.2 Illustration of the effect of an observation of $\hat{y}$ for $x^n = .6$, starting with an initial uniform prior (a). (b) and (c) show the new beliefs for $\hat{y} = 1$ and $\hat{y} = 0$. The updated distribution $p^{n+1}(x)$ is given in (d).

suggested region by q and the density in the other region by $1-q$. This leaves us with a density which does not integrate to 1, so we then finish the update by normalizing.

As a numerical example, assume that $q = .7$, $x^n = .6$, and that we observe $\hat{y}^{n+1} = 1$. Assume that the distribution $p^n(x)$ is uniform, as depicted in Figure 10.2(a). Figures 10.2(b) and 10.2(c) show the conditional distribution given $\hat{y}^{n+1} = 1$ and $\hat{y}^{n+1} = 0$, respectively. Finally, Figure 10.2(d) shows the updated distribution for p^{n+1} given $\hat{y}^{n+1} = 1$, overlaid on top of the original uniform distribution for $p^n(x)$. This distribution is computed using

$$p^{n+1}(x) = \begin{cases} \frac{g(.7,1)}{g(.7(.6)+.3(.4),0)} p^n(x) = \frac{0.7}{.54} p^n(x) = 1.296 p^n(x) & x < x^n \\ \frac{g(.7,0)}{g(.7(.6)+.3(.4),0)} p^n(x) = \frac{0.3}{.54} p^n(x) = 0.556 p^n(x) & x > x^n \end{cases}.$$

Let us briefly consider the case when our observations are always correct. Then $q = 1$ and we are back in the deterministic case. Suppose for the moment that p^n is uniform on some interval $(a^n, b^n]$. Then our updating rule can be simplified to

$$p^{n+1}(x) = \frac{g\left(1, I_{\{\hat{y}^{n+1}=I_{\{u\leq x^n\}}\}}\right)}{g(F^n(x^n)), \hat{y}^{n+1})} \frac{I_{\{x\in[a^n,b^n]\}}}{b^n - a^n} = \begin{cases} \frac{I_{\{x\in(a^n,x^n]\}}}{x^n - a^n} & \text{if } \hat{y}^{n+1} = 1, \\ \frac{I_{\{x\in(x^n,b^n]\}}}{b^n - x^n} & \text{if } \hat{y}^{n+1} = 0.. \end{cases}$$

Thus we see that the posterior is now still uniform but on some smaller interval, where either the points to the left or right of x^n have been removed. Thus, if we begin with a uniform prior, i.e., with $p^0(x) = I_{x\in[0,1]}$, then our posterior at each time will again be uniform on a smaller interval. Comparing our knowledge in the deterministic case, where we had an interval in which we knew x^* resided, to our knowledge here suggests that a natural way to express knowledge that x^* lies in an interval is through a uniform probability distribution on that interval.

10.2.3 Choosing the experiment

Now let us return to the stochastic case where $q < 1$. With our updating rule in hand, we could take any given method for choosing our experiments, and compute the posterior density of the location of x^* after our allotted number of experiments N, but before proceeding to say which experimental methods are best we need a way to evaluate the quality of the knowledge that we arrive to at the final time. In the deterministic case we knew that x^* resided in an interval, and we evaluated how happy we were with our final knowledge according to the length of this interval. In the stochastic case, however, we no longer have an interval but instead a density expressing a continuum of belief about the location of x^*.

The objective function we use should correspond to our notion of length in the deterministic case, and it should punish greater uncertainty. There are many choices, but one possibility is the entropy. Denote the entropy of any particular density p on the location of x^* by $H(p)$,

$$H(p) := -\int_0^1 \log_2(p(x))p(x)dx$$

The entropy corresponds to uncertainty about x^* in several senses. First, experiments always decrease entropy on average, no matter what is measured. Second, the entropy is largest for the uniform prior, which we may understand intuitively as the density we have when we are most uncertain about x^*. Third, the entropy approaches $-\infty$ as our posterior density sharpens to a point at the true location of x^*.

Additionally, the entropy corresponds in a very nice way to our use of interval length as the objective function in the deterministic case. In the deterministic case, we may know at a point in time that x^* is in $[a, b]$, but we have no information about it beyond that. A natural belief to take on the location of x^* in this situation is the uniform density on $[a, b]$, which is $p(x) = I_{x\in[a,b]}/(b-a)$. This density has entropy

$$H(p) = -\int_a^b \log_2(1/(b-a))/(b-a)dx = \log_2(b-a),$$

which is a strictly increasing function of the length $b - a$ of the interval. Thus minimizing the length of the interval $[b, a]$ is in some sense equivalent to minimizing the entropy. We characterize this equivalence more concretely later, when we show that the stochastic bisection algorithm is the same as the deterministic bisection algorithm when the probability q of experimental error is 0. Finally, an additional and very important reason for using entropy is that it provides an analytic and easy to use solution to the sequential problem.

Now with the transition function worked out and the entropy as our objective function, we have a well defined sequential information collection problem. This problem can be solved and the optimal solution computed using dynamic programming. Let us define our value function V^n by taking $V^n(p^n)$ to be the smallest value of $\mathbb{E}\left[H(p^N) \mid p^n\right]$ than can be achieved starting from the density p^n at time N. Bellman's principle then tells us that

$$V^n(p^n) = \min_x \mathbb{E}\left[V^{n+1}(p^{n+1}) \mid p^n, x^n = x\right].$$

In addition, we know that $V^N(p^N) = H(p^N)$ since there are no experiments left to make at time N.

We can use Bellman's recursion to compute V^{N-1} from V^N. Since $V^N = H$, this recursion is $V^{N-1}(p^{N-1}) = \min_x \mathbb{E}\left[H(p^N) \mid x^{N-1} = x, p^{N-1}\right]$. Rather than computing this here, we simply state the following formula, which can be confirmed by direct computation.

$$\min_x \mathbb{E}\left[H(p^{n+1}) \mid x^n = x, p^n\right] = H(p^n) - q\log_2 q - (1-q)\log_2(1-q) - 1, \quad (10.3)$$

and the minimum is achieved by choosing x to be a median of p^n. The median of p^n is defined to be the point where $F^n(x) = 1/2$. If there is more than one median, any point satisfying $F^n(x) = 1/2$ achieves the minimum.

Now, using (10.3) and the Bellman recursion, we see that V^{N-1} is given by

$$V^{N-1}(p^{N-1}) = H(p^{N-1}) - q\log_2 q - (1-q)\log_2(1-q) - 1,$$

and that the optimal decision x^{N-1} is the median of p^{N-1}. Moreover, we see that the form of the value function at time $N-1$ is the same as it is at time N, but with a constant subtracted. This constant does not depend on p^{N-1}, nor does it depend on n. This tells us that if we repeat the computation of the Bellman recursion we will find that $V^{N-2}(p^{N-2}) = H(p^{N-2}) - 2\left(q\log_2 q + (1-q)\log_2(1-q) + 1\right)$, and that in general

$$V^n(p^n) = H(p^n) - (N-n)\left(q\log_2 q + (1-q)\log_2(1-q) + 1\right). \quad (10.4)$$

Furthermore, since the minimizer of the Bellman recursion at each time n is the median of the density p^n at that time, we have discovered that the optimal policy is to always measure at the median. Denoting the optimal experiment at time n by $x^{*,n}$, we summarize this conclusion by saying that $x^{*,n}$ is such that

$$F^n(x^{*,n}) = 1/2.$$

Let us spend a few moments interpreting these results. First, the computation (10.3) tells us that if our goal is to minimize the expected entropy after one experiment, then the best x^n has the expected posterior entropy equal to the original entropy at time n minus a deterministic factor $q\log_2 q + (1-q)\log_2(1-q) + 1$. This factor is actually the mutual information between the experiment $\hat{y}^{n+1}$ and x^*, given that we measure at the median. This fact can be confirmed by computing the mutual information directly from its definition, although we do not perform this computation here.

We can view this reduction another way: the expected reduction in entropy about x^* is equal to the information about x^* contained in the outcome of the experiment, and this mutual information is maximized if we measure at the median of x^*. The mutual information is largest at the median because at this point we are "maximally uncertain" about to which region, $[0, x^*]$ or $(x^*, 1]$, it belongs since our belief assigns an equal probability of $1/2$ to each possibility. This conveys a general principle of information collection: often the experiment that is most valuable is the one whose result is least predictable. Put another way, if we already knew the result of an experiment before we took it, there would be no point in actually taking that experiment.

Then, the formula (10.4) shows that this general principle applies not just to single but multiple experiments. That is, the best we can do is to measure each time at the median of our belief, which is the experiment whose outcome is most uncertain, and the resulting decrease in expected entropy of x^* is equal to the sum of the mutual information in all the experimental outcomes combined. From experiment n onward, this decrease is $(N-n)\,(q\log_2 q + (1-q)\log_2(1-q)+1)$ since there are $N-n$ experiments left to make and each contributes $q\log_2 q + (1-q)\log_2(1-q)+1$.

We may also gain insight by taking the special case $q = 1$ and comparing to the deterministic case. As previously noted, when $q = 1$ and when we begin with a uniform prior on $[0, 1]$, the posterior remains uniform on a smaller interval. Then, the median of any such uniform posterior is simply the middle of the interval, and so we again always measure in the middle of the current interval, just as we did with deterministic bisection. Thus we see that the stochastic bisection algorithm reduces in the error-free case to exactly the classic bisection algorithm.

10.2.4 Discussion

We make one additional note about the usefulness of the stochastic bisection algorithm when our objective function is something different than the entropy objective we have assumed here. Although we have not shown it here, the decrease in entropy one obtains from measuring at the median is *deterministic*, even though the final density p^N itself is certainly random. This is similar to the situation in the deterministic case, where the location of the final interval containing x^* is unknown a priori, but the length of that interval is deterministic as long as we use the bisection rule.

This is a very nice property because it means that the optimal policy for the entropy objective function is also optimal for a broader class of objective functions. In particular, if our objective function is $\mathbb{E}[L(H(p^N))]$, where L is some concave increasing function, then again one can show that the same stochastic bisection rule is optimal. We can think of L as inducing some kind of risk aversion, in that using it would indicate we fear uncertainty about x^* more than than we hope for certainty. For example, earlier we saw that the length of an interval is equal to the logarithm of the entropy of a uniform distribution on this interval, and so if we wanted to minimize the length of the final interval in the deterministic case, then perhaps we should minimize the logarithm of the entropy rather than just the entropy. But we have already minimized using this criterion, since the logarithm is concave and increasing and the stochastic bisection algorithm is optimal for this objective function as well.

10.3 BIBLIOGRAPHIC NOTES

Section 10.1 - For a proof of the optimality of the Fibonacci search sequence for unimodular functions, see Avriel & Wilde (1966).

Section 10.2 - The development in this section is due to Jedynak et al. (2011).

PROBLEMS

10.1 How many iterations of a Fibonacci search sequence are needed to ensure that we can find the optimum of a unimodular function within 1 percent?

10.2 Consider the function $f(x|\alpha,\beta) = x^{\alpha-1}(1-x)^{\beta-1}$ for $0 \leq x \leq 1.0$. Perform the Fibonacci search to find the optimum of this function for the following values of (α,β).

a) $\alpha = 8, \beta = 2$.

b) $\alpha = 3, \beta = 12$.

c) $\alpha = 6, \beta = 8$.

10.3 Repeat exercise 10.2 using the golden section search.

10.4 Assume that we are trying to find the maximum of the function $f(x) = x^6(1-x)^2$, but now we are going to assume that we can compute the derivative.

a) Using the derivative to indicate whether the optimum is to the left or the right, perform eight iterations of deterministic bisection search and report your best estimate of the optimal solution. Plot your distribution of belief describing where the optimum lies after eight observations.

b) Now assume that we can estimate the sign of the derivative correction with probability $q = .70$ (even though we are computing it perfectly). Use the method described in Section 10.2 to find the optimum. Again plot your distribution of belief describing the location of the optimum.

c) How would your answer to (b) change if $q = .5$? You should be able to answer this without repeating any of the calculations.

CHAPTER 11

STOPPING PROBLEMS

Stopping problems are a simple but important class of learning problems. In this problem class, information arrives over time, and we have to choose whether to view the information or stop and make a decision. In this setting, "learning" means to continue to continue to receive information. We do not choose which information to observe, as we have done in the past, but we do have to decide whether to continue observing or stop and make a decision.

In this chapter we consider two classical stopping problems. The first is the sequential probability ratio test, where we have to quickly identify when a signal from some source is changing. The second is a classic problem known as the secretary problem, where we have to find the best out of a sequence of offers which arrive one at a time.

This chapter will illustrate learning using very different tools than we have seen up to now. Optimal stopping problems are both a very important problem class, as well as being a very special case which offers structure that we can exploit to derive optimal policies, rather than the well tuned approximations that have been the focus of the book until now.

11.1 SEQUENTIAL PROBABILITY RATIO TEST

A fundamental problem arises when we need to decide as quickly as possible when something is changing. For example, we may think we are observing data being generated by the sequence

$$W^n = \bar{\mu}^0 + \epsilon^n \tag{11.1}$$

where we assume that ϵ is normally distributed with mean 0 and variance σ^2. But we are aware that the mean might change to $\bar{\mu}^1$, which means the observations would come from the model

$$W^n = \bar{\mu}^1 + \epsilon^n. \tag{11.2}$$

We would like to design a method that allows us to determine when the mean has changed as quickly as possible.

A more general statement of the problem would be that the observations W^n are coming from an initial model that we refer to as the null hypothesis H_0. We then want to determine when the data is coming from a different model that we refer to as the alternative hypothesis, H_1. We start with a prior probability $\rho_0 = \rho_0^0$ that H_0 is true. Similarly, ρ_1^0 is our prior probability that H_1 is true. After we observe W^1, we would update our prior using Bayes' rule to obtain

$$\begin{aligned}\rho_0^1 &= P(H_0|W^1) \\ &= \frac{P(W^1|H_0)P(H_0)}{P(W^1)},\end{aligned}$$

where $P(W^1) = \rho_0 P(W^1|H_0) + \rho_1^0 P(W^1|H_1)$. The quantity $\rho_1^1 = P(H_1|W^1)$ would be worked out similarly. For example, if our data is normally distributed, we would write

$$P(W^1 = w|H_0) = \frac{1}{\sqrt{2\pi}\sigma} \exp -\frac{1}{2}\left(\frac{(w - \bar{\mu}^0)}{\sigma}\right)^2.$$

Let $p_0(w^n) = P(W^n = w^n|H_0)$. After n observations, we can write the posterior probability as

$$\begin{aligned}\rho_0^n &= \frac{\rho_0 \Pi_{k=1}^n p_0(w^k)}{\rho_0 \Pi_{k=1}^n p_0(w^k) + \rho_1^0 \Pi_{k=1}^n p_1(w^k)} \\ &= \frac{\rho_0 \lambda^n(w^1, \ldots, w^n)}{\rho_0 + \rho_1^0 \lambda^n(w^1, \ldots, w^n)}\end{aligned}$$

where

$$\lambda^n(w^1, \ldots, w^n) = \Pi_{k=1}^n \frac{p_0(w^k)}{p_1(w^k)}.$$

We can write $S^n = (W^1, W^2, \ldots, W^n)$ as being the set of all experiments, or it could be a sufficient statistic (such as the mean and variance of the normal distribution)

that captures everything we need to know from previous experiments. Later, we let $\lambda^n = \lambda^n(S^n) = \lambda^n(w^1, \ldots, w^n)$.

To solve our problem, we need to make two decisions. The first is whether to continue to observe W^n, or to stop and make a decision. If we decide to stop, we then have to choose between H_0 and H_1. We let

$$X^\pi(S^n) = \begin{cases} 1 & \text{if we decide to stop and make a decision,} \\ 0 & \text{if we continue observing.} \end{cases}$$

$$Y^\pi(S^n) = \begin{cases} 1 & \text{if we decide } H_1 \text{ is correct,} \\ 0 & \text{if we decide } H_0 \text{ is correct.} \end{cases}$$

We let π denote a policy consisting of the two functions (X^π, Y^π). Given a policy π, there are two mistakes we can make. The first is a false alarm, which means we stop and conclude that H_1 is true when in fact H_0 is true, and a miss, which means that H_1 is true, but we did not pick it up and we still think H_0 is true. We define the probability of these two events using

$$\begin{aligned} P_F^\pi &= \text{the probability we conclude } H_1 \text{ is true (the false alarm) given } H_0 \text{ when using policy } \pi \\ &= \mathbb{E}[Y^\pi(S^n)|H_0], \\ P_M^\pi &= \text{the probability we conclude } H_0 \text{ is true given } H_1 \text{ when using policy } \pi \\ &= \mathbb{E}[1 - Y^\pi(S^n)|H_1]. \end{aligned}$$

Now let ρ_0 be the prior probability that H_0 is true. We can define the overall probability of an error using

$$P_e = (1 - \rho_0)P_F^\pi + \rho_0 P_M^\pi.$$

Of course, we can minimize this error by running many experiments. The number of experiments is given by

$$N^\pi = \min\{n|X^\pi(S^n) = 1\}.$$

N^π is a random variable that depends on our policy (the decision function X^π) and the observations $(W^1, W^2, \ldots, W^n)$. We can assign a "cost" c to each experiment, giving us a utility function

$$U^\pi(c) = P_e + c\mathbb{E}N^\pi.$$

Here, c is not a true cost in the sense of being measured in units of dollars per experiment. Rather, it is a scaling coefficient that allows us to combine the probability of being wrong with the number of experiments. Our challenge is to find a policy π that minimizes the utility function $U^\pi(c)$.

The problem can be solved (approximately) by exploiting some nice structural properties. We first write the conditional risk as

$$\begin{aligned} r_0^\pi &= P_F^\pi + c\mathbb{E}[N^\pi|H_0], \\ r_1^\pi &= P_M^\pi + c\mathbb{E}[N^\pi|H_1], \\ r^\pi &= \rho_0 r_0^\pi + (1 - \rho_0)r_1^\pi. \end{aligned}$$

We would like to find a policy π that minimizes the risk, given by

$$R^0(\rho_0) = \min_{\pi} r^{\pi}.$$

We start by observing that if $\rho_0 = 1$ (which means we are positive that H_1 is true), then we can stop and choose $Y = 1$ with no risk of a false positive (and $N = 0$), which means that the risk is $R^0(1) = 0$. The same reasoning tells us that $R^0(0) = 0$. It is also possible to show that $R^0(\rho)$ is concave in ρ.

Assume that we stop after running no experiments. If we choose $Y = 0$ then the risk is $R^0(\rho_0|Y = 0) = \rho_0$ (which is the same thing as saying that if I have to choose now with no information, my probability of being right is my original prior that H_0 is true). Similarly, if we choose $Y = 1$ then the risk is $R^0(\rho_0|Y = 1) = 1 - \rho_0$. Or, we could choose to make a single experiment (sort of like choosing curtain number 3). In this case, we want to choose the experimental policy π that solves

$$R^1(\rho_0^0) = \min_{\{\pi, N>0\}} \rho_0^0 r_0^{\pi} + (1 - \rho_0^0) r_1^{\pi}.$$

Here, we are solving the same problem as we were at time 0, but we are now forcing ourselves to take at least one experiment. So, our policy will be to run a single experiment (which means $N > 0$), and then we have to do the best we can choosing between $Y = 0$, $Y = 1$ or running yet another experiment.

So, we want the smallest of ρ_0^0 (corresponding to stopping and choosing H_0), $1 - \rho_0^0$ (corresponding to stopping and choosing H_1) and $R^1(\rho^0)$ (which means take another observation and repeat the process). The problem is depicted in Figure 11.1. Here, we plot the lines ρ_0^0 and $1 - \rho_0^0$, and the concave function $R^1(\rho^0)$, all as a function of ρ_0^0. If c is large enough, it is possible that the midpoint of $R^1(\rho^0)$ is greater than .5, in which case the best choice is to stop right away ($N = 0$) and choose between H_0 and H_1. Now assume that the maximum of $R^1(\rho^0)$ is less than .5. In this case, run one experiment and compute the posterior ρ_0^1. We then divide the horizontal axis into three regions: $\rho_0 < \rho^L$, $\rho^L \leq \rho_0 \leq \rho^U$, and $\rho_0 > \rho^U$. If $\rho_0 < \rho^L$, then the best choice is to choose $Y = 0$. If $\rho_0 > \rho^U$ then we stop and choose $Y = 1$. If $\rho^L \leq \rho_0 \leq \rho^U$, then we run another experiment and repeat the process. After each experiment, we face the same problem, where the only change is that we have a new prior.

This seems like a pretty simple rule. The only challenge is finding ρ^L and ρ^U. We begin by computing the likelihood ratio

$$L^n(S^n) = \Pi_{k=1}^n \frac{p_1(W^k)}{p_0(W^k)} = \Pi_{k=1}^n L(W^k)$$

where $L(W^k) = \frac{p_1(W^k)}{p_0(W^k)}$ is the likelihood ratio for a single observation. We next use Bayes' rule to compute the posterior $\rho_0^{n+1}(W^n)$ as follows

$$\begin{aligned}
\rho_1^{n+1}(S^n) &= \frac{p_1(S^n)(1 - \rho_0^n)}{\rho_0^n p_0(S^n) + (1 - \rho_0^n) p_1(S^n)} \\
&= \frac{L^n(S^n)}{L^n(S^n) + \rho_0^n/(1 - \rho_0^n)} \\
&= f(L^n(S^n), \rho_0^n/(1 - \rho_0^n)),
\end{aligned}$$

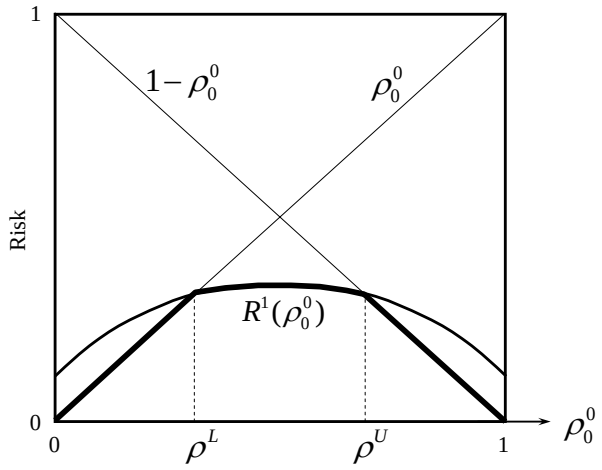


Figure 11.1 The expected risk as a function of the prior probability that H_0 is true.

where $f(\ell, \beta) = \ell/(\ell + \beta)$. For $\beta \geq 0$, $f(\ell, \beta)$ is strictly increasing for $\ell \geq 0$. This means that if $0 < \rho_0^n < 1$, $\rho_0^{n+1}(S^n)$ is strictly increasing with $L^n(S^n)$. This means that determining if $\rho_0^{n+1}(S^n) \leq \rho^L$ or $\rho_0^{n+1}(S^n) \geq \rho^U$ is the same as testing if $L^n(S^n) \leq A$ or $L^n(S^n) \geq B$, where A and B satisfy

$$\begin{aligned} f\left(A, \rho_0^n/(1-\rho_0^n)\right) &= \rho^L, \\ f\left(B, \rho_0^n/(1-\rho_0^n)\right) &= \rho^U. \end{aligned}$$

We can solve for A and B, which gives us

$$\begin{aligned} A &= \frac{\rho_0^n \rho^L}{(1-\rho_0^n)(1-\rho^L)}, \\ B &= \frac{\rho_0^n \rho^U}{(1-\rho_0^n)(1-\rho^U)}. \end{aligned}$$

This means that we can write our policy in the form

$$L^n(S^n) = \begin{cases} \geq B & \text{stop and choose } Y^n = 1 \\ \leq A & \text{stop and choose } Y^n = 0 \\ \text{otherwise} & \text{take an additional observation.} \end{cases}$$

Hence, it is easy to see why this rule is known as the *sequential probability ratio test* ($SPRT$). The $SPRT$ is controlled by the parameters A and B, and hence we refer to the rule as $SPRT(A, B)$.

Finding A and B exactly is difficult, but we can find good estimates using *Wald's approximation* which gives us

$$\begin{aligned} P_F^\pi &\approx \frac{(1-A)}{B-A}, \\ P_M^\pi &\approx \frac{A(B-1)}{B-A}. \end{aligned}$$

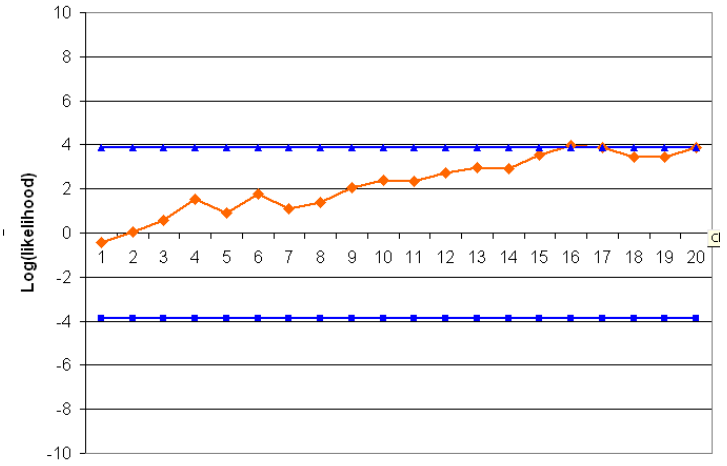


Figure 11.2 Sample path of ln of SPRT likelihood function

We then choose the acceptable probability of a false positive, P_F^π, and the probability of a miss, P_M^π, and then solve for A and B, giving us

$$\begin{aligned} A &= \frac{P_M^\pi}{1 - P_F^\pi}, \\ B &= \frac{1 - P_M^\pi}{P_F^\pi}. \end{aligned}$$

This rule is (approximately) optimal in the sense that it meets these goals for the probability of a false positive and the probability of missing a change with the fewest number of experiments. We note that it is customary to work in terms of the logarithm of L.

Figure 11.2 plots the log of the likelihood for a set of sample observations. After 16 observations, we conclude that H_1 is true.

11.2 THE SECRETARY PROBLEM

The so-called secretary problem is one of the first (formally defined) learning problems. The motivation of the problem is determining when to hire a candidate for a job (presumably a secretarial position), but it can also be applied to reviewing a series of offers for an asset (such as selling your house or car). The problem involves the tradeoff between observing candidates (which allows us to collect information) and making a decision (exploiting the information). As with the sequential probability ratio test in the previous section, our decision is when to stop reviewing new offers and accept the most recent offer. In contrast with our work on Bayesian models, at the heart of the secretary problem is that we assume that we know absolutely nothing about the distribution of offers.

11.2.1 Setup

Assume that we have N candidates for a secretarial position (you can also think of these as offers to purchase an asset). Each candidate is interviewed in sequence and assigned a score that allows us to compare him or her to other candidates (if we are trying to sell an asset, these scores are the offers to purchase the asset). While it may be reasonable to try to maximize the expected score that we would receive, in this case, we want to maximize the probability of accepting the highest score out of all that have been, or might be, offered. We need to keep in mind that if we stop at candidate n, then we will not interview candidates $n+1, \ldots, N$. Also, we only have the option of accepting the last candidate or interviewing the next one. Once we have turned down a candidate, we cannot return to that candidate at a later time.

Let

$$
\begin{aligned}
W^n &= \text{Score of the } n\text{th candidate.} \\
S^n &= \begin{cases} 1 & \text{If the score of the } n\text{th candidate is the best so far} \\ 0 & \text{If the score of the } n\text{th candidate is not the best so far} \\ \Delta & \text{If we have stopped already} \end{cases} \\
\mathcal{S} &= \text{State space, given by } (0, 1, \Delta), \text{ where the states 0 and 1 mean that we are still searching, and } \Delta \text{ means we have stopped the process.} \\
\mathcal{X} &= \{0 \text{ (continue)}, 1 \text{ (stop)}\}, \text{ where "stop" means that we hire the last candidate interviewed.}
\end{aligned}
$$

Because the decision function uses the most recent piece of information, we define our history as

$$h^n = \{W^1, \ldots, W^n\}.$$

To describe the system dynamics, it is useful to define an indicator function

$$I^n(h^n) = \begin{cases} 1 & \text{if } W^n = \max_{1 \le m \le n}\{W^m\} \\ 0 & \text{otherwise.} \end{cases}$$

which tells us if the last observation is the best. Our transition function can now be given by

$$S^{n+1} = \begin{cases} I^n(h^n) & \text{if } x^n = 0 \text{ and } S^n \neq \Delta \\ \Delta & \text{if } x^n = 1 \text{ or } S^n = \Delta. \end{cases}$$

To compute the one-step transition matrix, we observe that the event the $(n+1)$st applicant is the best has nothing to do with whether the nth was the best. As a result, we can write the conditional probability that $I^{n+1}(h^{n+1}) = 1$ using

$$\mathbb{P}[I^{n+1}(h^{n+1}) = 1 | I^n(h^n)] = \mathbb{P}[I^{n+1}(h^{n+1}) = 1].$$

This simplifies the problem of finding the one-step transition probabilities. By definition we have

$$\mathbb{P}[S^{n+1} = 1 | S^n, x^n = 0] = \mathbb{P}[I^{n+1}(h^{n+1}) = 1].$$

$I^{n+1}(h^{n+1}) = 1$ if the $(n+1)$st candidate is the best out of the first $n+1$, which clearly occurs with probability $1/(n+1)$. So

$$\begin{aligned}\mathbb{P}(S^{n+1} = 1|S^n, x^n = 0) &= \frac{1}{n+1},\\ \mathbb{P}(S^{n+1} = 0|S^n, x^n = 0) &= \frac{n}{n+1}.\end{aligned}$$

Our goal is to maximize the probability of hiring the best candidate. So, if we do not hire the last candidate, then the probability that we hired the best candidate is zero. If we hire the nth candidate, and the nth candidate is the best so far, then our reward is the probability that this candidate is the best out of all N. This probability is simply the probability that the best candidate out of all N is one of the first n, which is n/N. So, the conditional reward function is

$$C^n(S^n, x^n|h^n) = \begin{cases} n/N & \text{if } S^n = 1 \text{ and } x^n = 1,\\ 0 & \text{otherwise.}\end{cases}$$

With this information, we can now set up the optimality equations

$$V^n(s^n) = \max_{x^n \in \mathcal{X}} \mathbb{E}\{C^n(s^n, x^n|h^n) + V^{n+1}(S^{n+1})|s^n\}.$$

11.2.2 Solution

The solution to the problem is quite elegant, but the technique is unique to this particular problem. Readers interested in the elegant answer but not the particular proof (which illustrates dynamic programming but otherwise does not generalize to other problem classes) can skip to the end of the section.

Let $V^n(s)$ be the probability of choosing the best candidate out of the entire population, given that we are in state s after interviewing the nth candidate. Recall that implicit in the definition of our value function is that we are behaving optimally from time period t onward. The terminal reward is

$$\begin{aligned}V^N(1) &= 1\\ V^N(0) &= 0\\ V^N(\Delta) &= 0\end{aligned}$$

Let

$$\begin{aligned}C^{stop,n} &= \left(C^n(1, \text{stop}) + V^{n+1}(\Delta)\right),\\ C^{continue,n} &= \left(C^n(1, \text{continue}) + \sum_{s' \in \{0,1\}} p(s'|s)V^{n+1}(s')\right).\end{aligned}$$

The optimality recursion for the problem is given by

$$V^n(1) = \max\left\{C^{stop,n}, C^{continue,n}\right\}.$$

Noting that:

$$\begin{aligned} C^n(1,\text{continue}) &= 0, \\ C^n(1,\text{stop}) &= \frac{n}{N}, \\ V^{n+1}(\Delta) &= 0, \\ p(s'|s) &= \begin{cases} 1/(n+1) & s'=1 \\ n/(n+1) & s'=0 \end{cases}. \end{aligned}$$

We get

$$V^n(1) = \max\left\{\frac{n}{N}, \frac{1}{n+1}V^{n+1}(1) + \frac{n}{n+1}V^{n+1}(0)\right\}. \tag{11.3}$$

Similarly, it is easy to show that

$$\begin{aligned} V^n(0) &= \max\left\{0, \frac{1}{n+1}V^{n+1}(1) + \frac{n}{n+1}V^{n+1}(0)\right\} \qquad (11.4) \\ &= \frac{1}{n+1}V^{n+1}(1) + \frac{n}{n+1}V^{n+1}(0). \end{aligned}$$

Comparing (11.4) and (11.3), we can rewrite (11.3) as

$$V^n(1) = \max\left\{\frac{n}{N}, V^n(0)\right\}. \tag{11.5}$$

From this we obtain the inequality

$$V^n(1) \geq V^n(0) \tag{11.6}$$

which seems pretty intuitive (you are better off if the last candidate you interviewed was the best you have seen so far).

At this point, we are going to suggest a policy that seems to be optimal. We are going to interview the first $\bar{n}$ candidates, without hiring any of them. Then, we will stop and hire the first candidate who is the best we have seen so far. The decision rule can be written as

$$x^n(1) = \begin{cases} 0 \text{ (continue)} & n \leq \bar{n} \\ 1 \text{ (quit)} & n > \bar{n} \end{cases}.$$

To prove this, we are going to start by showing that if $V^m(1) > m/N$ for some m (or alternatively if $V^m(1) = m/N = V^m(0)$), then $V^{m'}(1) > m'/N$ for $m' < m$. If $V^m(1) > m/N$, then it means that the optimal decision is to continue. We are going to show that if it was optimal to continue at set m, then it was optimal to continue for all steps $m' < m$.

Assume that $V^m(1) > m/N$. This means, from equation (11.5), that it was better to continue, which means that $V^m(1) = V^m(0)$ (or there might be a tie, implying that $V^m(1) = m/N = V^m(0)$). This allows us to write

$$\begin{aligned} V^{m-1}(0) &= \frac{1}{m}V^m(1) + \frac{m-1}{m}V^m(0) \\ &= V^m(1) \qquad (11.7) \\ &\geq \frac{m}{N}. \qquad (11.8) \end{aligned}$$

Equation (11.7) is true because $V^m(1) = V^m(0)$, and equation (11.8) is true because $V^m(1) \geq m/N$. Stepping back in time, we get

$$\begin{aligned} V^{m-1}(1) &= \max\left\{\frac{m-1}{N}, V^{m-1}(0)\right\} \\ &\geq \frac{m}{N} \qquad (11.9) \\ &> \frac{m-1}{N}. \qquad (11.10) \end{aligned}$$

Equation (11.9) is true because $V^{m-1}(0) \geq m/N$. We can keep repeating this for $m-1, m-2, \ldots$, so it is optimal to continue for $m' < m$.

Now we have to show that if $N > 2$, then $\bar{n} \geq 1$. If this is not the case, then for all n, $V^n(1) = n/N$ (because we would never continue). This means that (from equation (11.4)):

$$\begin{aligned} V^n(0) &= \left(\frac{1}{n+1}\right)\left(\frac{n+1}{N}\right) + \left(\frac{n}{n+1}\right) V^{n+1}(0) \\ &= \frac{1}{N} + \left(\frac{n}{n+1}\right) V^{n+1}(0). \qquad (11.11) \end{aligned}$$

Using $V^N(0) = 0$, we can solve (11.11) by backward induction:

$$\begin{aligned} V^N(0) &= 0, \\ V^{N-1}(0) &= \frac{1}{N} + \frac{N-1}{N-1+1} V^N(0) \\ &= \frac{1}{N}, \\ V^{N-2}(0) &= \frac{1}{N} + \frac{N-2}{N-2+1}\left(\frac{1}{N}\right) \\ &= \frac{N-2}{N}\left(\frac{1}{N-2} + \frac{1}{N-1}\right). \end{aligned}$$

In general, we get

$$V^m(0) = \frac{m}{N}\left[\frac{1}{m} + \frac{1}{m+1} + \cdots + \frac{1}{N-1}\right].$$

We can easily see that $V^1(0) > \frac{1}{N}$; since we were always quitting, we had found that $V^1(1) = \frac{1}{N}$. Finally, equation (11.6) tells us that $V^1(1) \geq V^1(0)$, which means we have a contradiction.

This structure tells us that for $m \leq \bar{n}$

$$V^m(0) = V^m(1),$$

and for $m > \bar{n}$

$$\begin{aligned} V^m(1) &= \frac{m}{N}, \\ V^m(0) &= \frac{m}{N}\left[\frac{1}{m} + \frac{1}{m+1} + \cdots + \frac{1}{N-1}\right]. \end{aligned}$$

It is optimal to continue as long as $V^m(0) > m/N$, so we want to find the largest value for m such that

$$\frac{m}{N}\left[\frac{1}{m} + \frac{1}{m+1} + \cdots + \frac{1}{N-1}\right] > \frac{m}{N},$$

or

$$\left[\frac{1}{m} + \frac{1}{m+1} + \cdots + \frac{1}{N-1}\right] > 1.$$

If $N = 5$, then we can solve by enumeration:

$$\begin{array}{lll} \bar{n} = 1 & \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} & = 2.08 \\ \bar{n} = 2 & \frac{1}{2} + \frac{1}{3} + \frac{1}{4} & = 1.08 \\ \bar{n} = 3 & \frac{1}{3} + \frac{1}{4} & = 0.58 \end{array}$$

So for $N = 5$, we would use $\bar{n} = 2$. This means interview (and discard) two candidates, and then take the first candidate that is the best to date.

For large N, we can find a neat approximation. We would like to find m such that:

$$\begin{aligned} 1 &\approx \frac{1}{m} + \frac{1}{m+1} + \cdots + \frac{1}{N-1} \\ &\approx \int_M^N \frac{1}{x} dx \\ &= \ln N - \ln m \\ &= \ln\left(\frac{N}{m}\right). \end{aligned}$$

Solving for m means finding $\ln(N/m) = 1$ or $N/m = e$ or $m/N = e^{-1} = 0.368$. So, for large N, we want to interview 37 percent of the candidates, and then choose the first candidate that is the best to date.

The secretary problem is a classic, partly because it illustrates an interesting information collection problem, and partly because it yields such an elegant solution. In real applications, we can translate the result to a rough rule that says "look at a third of the candidates, and then choose the first candidate that is better than all the others."

11.3 BIBLIOGRAPHIC NOTES

Section 11.1 - The sequential probability ratio test is due to Wald & Wolfowitz (1948).

Section 11.2 - The secretary problem was first introduced in Cayley (1875). Our presentation is based on Puterman (1994). Vanderbei (1980) provides an elegant generalization of the secretary problem to one of finding the best subset. See also Bruss (1984) for extensions.

PROBLEMS

11.1 Download a spreadsheet illustrating the sequential probability ratio test from

`http://optimallearning.princeton.edu/exercises/SPRT.xls`

In the initial spreadsheet, the standard deviation of an experimental observation has been set to 5.

a) Translate the cells in row 12 to mathematics. Identify the equation in the book corresponding to each cell starting in column D.

b) Change the probability of missing from .02 to .10. How does this change the hypotheses $H0$ and $H1$? Now change the probability of a false alarm from .02 to .10? How does this change the hypotheses? When you are done this question, restore both probabilities back to .02.

c) The sequential probability ratio test stops with a conclusion that either hypothesis $H0$ is true or $H1$ is true when the red line crosses one of the blue lines. If it does not cross either line within the experimental budget, then the test is inconclusive. Hit the F9 key 20 times, and count the number of times the red line crosses one of the blue lines.

d) Now change the standard deviation (cell B9) to 3. Again perform 20 simulations and count how many times the red line crosses one of the blue lines. You should see that with a lower standard deviation, SPRT is more effective at declaring that one of the two hypotheses is true. Why does a smaller standard deviation make it easier to come to a conclusion?

11.2 Assume you think you can look at up to 20 bids for the house you are selling. Completely unknown to you, the bids can be modeled as being drawn from a uniform distribution between $380,000 and $425,000. Use the policy that you are going to reject the first seven bids (which is approximately 37 percent of 20), and then accept the first bid that is better than these seven. If none are better, you have to accept the very last bid. Start the process by generating all 20 bids in advance, so that when you are done, you can compare the bid you accepted against the best that you might have accepted with perfect foresight.

a) Repeat this policy 100 times and report on: i) how many times you accepted the very best bid out of the 20, and ii) the bid you accepted as a percentage of the very best bid.

b) Repeat (a), but now reject only the first four bids before you are ready to accept the bid.

c) Finally, repeat (a), but now you decide to reject the first 10 bids before you are ready to accept a bid.

d) Compare the results of the policies you simulated in (a), (b) and (c). Do you see evidence that one policy outperforms the others?

11.3 You have to choose the best out of up to 30 bids, where the i^{th} bid, R_i, follows an exponential distribution given by

$$f_R(y) = .02e^{-.02y}.$$

You can generate random observations from an exponential distribution using

$$R = -50 \ln U$$

where U is a random variable that is uniformly distributed between 0 and 1.

a) Use the policy where you look at the first 11 bids, and then pick the best bid which outperforms all previous bids. Repeat this policy 100 times and report on: i) how many times you accepted the very best bid out of the 20, and ii) the bid you accepted as a percentage of the very best bid.

b) Repeat (a), but now reject only the first six bids before you are ready to accept the bid.

c) Finally, repeat (a), but now you decide to reject the first 16 bids before you are ready to accept a bid.

d) Compare the results of the policies you simulated in (a), (b) and (c). Do you see evidence that one policy outperforms the others?

CHAPTER 12

ACTIVE LEARNING IN STATISTICS

We have focused our attention on problems where we are learning in order to make a better decision in some sort of optimization problem, which gives us an economic value of information. It is often the case, however, that we are just trying to fit a statistical model which might be used for a variety of decision problems. We may not know how the model may be used; we are simply interested in carefully choosing what to measure to get the best model.

Most applications of statistics involve problems where we may have to fit a model from a fixed set of observations. This is often referred to as batch statistics. We may also have a process where observations arrive over time, but where we do not have any control over what we observe. This is known as *passive learning*.

There are many situations where we can control the inputs of a process. For example, we may be able to set the price of a product and then observe sales. We may set user preferences on Netflix, thus affecting the movies that are displayed to us; we then choose a movie to rent, which in turn affects what Netflix observes. After we choose the inputs (these might be referred to as independent variables or covariates) x^n, we then observe the response y^{n+1} which is then used to fit the parameters of a model. The machine learning community refers to this process as *active learning*.

In statistics, active learning refers to the ability to control the choice of independent variables. In Chapter 1, we made a distinction between the broad umbrella of active learning, where we make choices with the intent to learn, and the subset of policies

Optimal Learning. By Warren B. Powell and Ilya O. Ryzhov

we call optimal learning, where our choice is guided by a well-defined objective function. We retain this distinction in this chapter, but cover a variety of heuristic and optimal policies for deciding what observations to make when fitting a function. We divide these methods into the following classes:

1) Deterministic policies - These determine all the points to observe in advance, without the benefit of learning the results of any experiments. We consider a special case where these policies are optimal.

2) Heuristic sequential policies - These are active learning policies where the choice of what to observe uses a rule that is not based on any particular performance metric.

3) Variance minimizing policies - These are sequential policies designed to minimize the variance of an estimator.

12.1 DETERMINISTIC POLICIES

There is an extensive literature in statistics that goes under the name design of experiments. The problem is to choose a set of independent variables $x^1, \ldots, x^n$, where x^m is an F-dimensional vector of features which generates an observation $\hat{y}^m$. Let x^n be the vector of independent variables (features), given by

$$x^n = \begin{pmatrix} x_1^n \\ x_1^n \\ \vdots \\ x_F^n \end{pmatrix},$$

where $|\mathcal{F}|$ is a set of features, with $F = |\mathcal{F}|$. Our goal is to fit a linear model

$$y = \sum_{f \in \mathcal{F}} \theta_f x_f + \varepsilon,$$

where ε is an error term.

We want to choose θ to minimize the total errors squared given by

$$\min_{\theta} F(\theta) = \sum_{m=1}^{n} \left(y^m - \sum_{f \in \mathcal{F}} \theta_f x_f^m \right)^2. \tag{12.1}$$

To find the optimal solution, we begin by defining the matrix X^n as

$$X^n = \begin{pmatrix} x_1^1 & x_2^1 & & x_F^1 \\ x_1^2 & x_2^2 & & x_F^2 \\ \vdots & \vdots & \cdots & \vdots \\ x_1^n & x_2^n & & x_F^n \end{pmatrix}.$$

The vector of observations $\hat{y}^1, \ldots, \hat{y}^n$ is represented using

$$Y^n = \begin{pmatrix} \hat{y}^1 \\ \hat{y}^2 \\ \vdots \\ \hat{y}^n \end{pmatrix}.$$

As in Chapter 7, the vector θ^n that solves (12.1) is given by

$$\theta^n = [(X^n)^T X^n]^{-1}(X^n)^T Y^n. \tag{12.2}$$

Equation (12.2) gives us an easy way to find the variance of our estimate θ^n. Let v be an n-dimensional random vector, let A be a $F \times n$ deterministic matrix, and let $u = Av$. Let $Cov(v)$ be the covariance matrix of v. Then we can use the identity that

$$Cov(u) = ACov(w)A^T$$

where $Cov(w)$ is the covariance matrix of w. Recall that for matrices A and B, $AB^T = (BA^T)^T$, and that $[(X^n)^T X^n]^{-1}$ is a symmetric matrix. We can use these observations to find the covariance matrix of θ^n if we let $A = [(X^n)^T X^n]^{-1}(X^n)^T$, giving us

$$\begin{aligned} Cov(\theta^n) &= [(X^n)^T X^n]^{-1}(X^n)^T Cov(Y^n)\left([(X^n)^T X^n]^{-1}(X^n)^T\right)^T \\ &= [(X^n)^T X^n]^{-1}(X^n)^T Cov(Y^n)(X^n)[(X^n)^T X^n]^{-1}. \end{aligned}$$

Since the elements of Y^n are independent, $Cov(Y^n) = \sigma_\epsilon^2 I$ where I is the identity matrix and σ_ϵ^2 is the variance of our experimental error . This allows us to write

$$\begin{aligned} Cov(\theta^n) &= [(X^n)^T X^n]^{-1}(X^n)^T X^n [(X^n)^T X^n]^{-1}\sigma_\epsilon^2 \\ &= [(X^n)^T X^n]^{-1}\sigma_\epsilon^2. \end{aligned}$$

The observation error σ_ϵ^2 is fixed; we cannot change it through our choice of x^n. However, we have direct control over the covariance of θ^n by our choice of X^n. Furthermore, we quickly see that the matrix $Cov(\theta^n)$ is a deterministic function of X^n, which means we do not gain anything by observing $\hat{y}^n$. It is for this reason that we can determine the precision of θ^n without making any observations. This is the theoretical foundation of deterministic policies.

So this leaves the question, what do we want to minimize? θ^n is a vector, and $Cov(\theta^n)$ is a matrix. While we would like to minimize all the elements of $Cov(\theta^n)$, we have to choose a single metric. This issue has created a variety of metrics that have come to be known as alphabet-optimality, for reasons that will become clear shortly. A sample of these metrics are

A-optimality - Minimize the average (or trace, which is the sum) of the diagonal elements of $[(X^n)^T X^n]^{-1}$. This is the same as minimizing the average of the variances of each element of θ^n.

C-optimality - Given a weighting vector c, minimize $c^T[(X^n)^T X^n]c$.

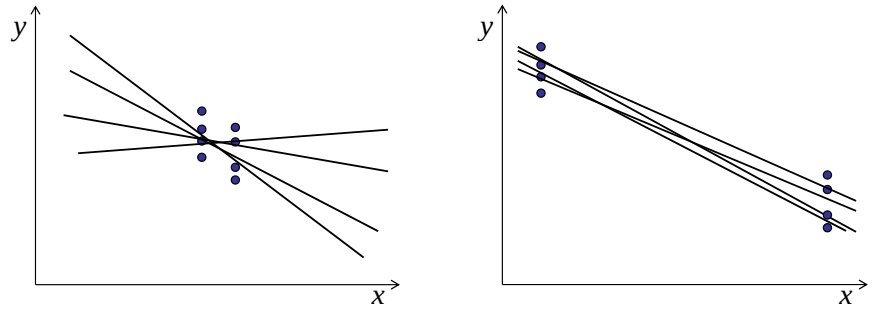


Figure 12.1 Learning a line with closely spaced experiments (a) and experiments with more separation (b).

D-optimality - Minimize $|[(X^n)^T X^n]^{-1}|$, or, equivalently, maximize the determinant of $[(X^n)^T X^n]$.

E-optimality - Maximize the minimum eigenvalue of $[(X^n)^T X^n]$.

G-optimality - Maximize the largest element in the diagonal of $X^n[(X^n)^T X^n](X^n)^T$, which has the effect of minimizing the largest variance of θ^n.

I-optimality - Minimize the average prediction variance over a particular region.

T-optimality - Maximize the trace of $[(X^n)^T X^n]$.

V-optimality - Minimize the average prediction variance over a specific set of points.

All of these methods aim at making the matrix $[(X^n)^T X^n]^{-1}$ smaller in some way, or, equivalently, making the matrix $[(X^n)^T X^n]$ bigger.

We can illustrate the central intuition behind these strategies using the simple example of fitting a line. Figure 12.1(a), reprised here for convenient reading from Figure 7.9, shows the lines that we might estimate if we make a small number of observations that are close to each other. Figure 12.1(b) illustrates the estimates that we might obtain if the experiments are spaced farther apart. It is not surprising that the more widely spaced observations provide a better estimate.

We can quantify this intuitive behavior by computing the matrix $[(X^n)^T X^n]$. Assume we make two observations each at two different locations. An observation x^n might be $(1, 5)$, where the 1 corresponds to the constant term, and the 5 is the value we are measuring. The closely spaced points might be

$$X^n = \begin{pmatrix} 1 & 5 \\ 1 & 5 \\ 1 & 6 \\ 1 & 6 \end{pmatrix}. \tag{12.3}$$

The matrix $[(X^n)^T X^n]$ is given by

$$[(X^n)^T X^n] = \begin{bmatrix} 4 & 22 \\ 22 & 122 \end{bmatrix}.$$

Now assume we measure at more extreme points (but where the average is still the same), given by

$$X^n = \begin{pmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 10 \\ 1 & 10 \end{pmatrix}.$$

The matrix $[(X^n)^T X^n]$ is given by

$$[(X^n)^T X^n] = \begin{bmatrix} 4 & 22 \\ 22 & 202 \end{bmatrix}.$$

The trace of the matrix $[(X^n)^T X^n]$ for the closely spaced points is $4 + 122 = 126$, while the trace for the points that are more spread out is 206. We would say that the second matrix has more "information," and the result is an estimate of θ with lower variance.

It is important that the data be scaled so that the average value of the measured values x remains the same (as we did above). The best way to do this is to simply average the outcomes of the experiments, and compute a corrected value $\bar{X}^n$ for each dimension, using

$$\bar{X}_i^n = X_i^n - \frac{1}{n}\sum_{m=1}^{n} X_i^m.$$

We then compute the matrix using $[(\bar{X}^n)^T \bar{X}^n]$. If we do this to the experimental outcomes in equation (12.3), we would obtain

$$\bar{X}^n = \begin{pmatrix} 0 & -0.5 \\ 0 & -0.5 \\ 0 & 0.5 \\ 0 & 0.5 \end{pmatrix}. \tag{12.4}$$

Regardless of which type of optimality we use, we now have a metric that we can use to help decide which experiments to run. Given a set of potential experiments, we generally want to choose experiments that are distributed around a center, but as far from the center as possible.

It is important to realize that we can choose the best set of potential experiments before making any observations. This property is a unique byproduct of the property that all statistics relating to the reliability of our estimates of θ are purely a function of the experiments and not the observations. We have not enjoyed this property in our previous applications.

12.2 SEQUENTIAL POLICIES FOR CLASSIFICATION

An important class of problems in machine learning is known as classification problems. In this problem class, we are given a set of features x^m for the mth object (this

might be a document, an email or a website), and we are asked to place this document into one of a set of discrete classifications, such as {dangerous, threatening, suspicious, safe}. Often, collecting information about the classification of a document for training purposes is expensive. For example, we may need to ask a security expert to assess the threat level of an email or website. However, given the sheer volume of these sources, we cannot ask a trained expert to provide this information on a large number of documents.

This section describes a series of primarily heuristic search policies for efficiently collecting information for classification problems.

12.2.1 Uncertainty sampling

Let $\hat{y}$ be a discrete quantity that indicates the classification of a document with attributes described by x, and let $P_\theta(\hat{y}|x)$ be the probability of the particular classification $\hat{y}$ if we observe x, given a parameter vector θ. Our goal is to observe the document where our prediction has the highest level of uncertainty. If the classification is binary (for example, dangerous or not), then we want to sample the documents where $P_\theta(\hat{y}|x)$ is as close as possible to 0.5.

If there are more than two outcomes, an alternative strategy is to choose to query the document whose prediction offers the lowest level of confidence, which we compute using

$$x^{LC} = \arg\max_x \left(1 - P_\theta(\hat{y}|x)\right)$$

where

$$\hat{y} = \arg\max_y P_\theta(y|x)$$

is the most likely classification. The idea here is that if the most likely classification in a set is small, then this represents a document (more precisely, a class of documents) where we have a lot of uncertainty, and would benefit from more information. We are trying to mitigate the effect of a worst-case scenario by learning about the document for which we are most likely to make an error in prediction.

This strategy focuses on the most probable classification, and as a result ignores the probabilities of other classifications. For example, we may be much more certain about one classification, implying that more testing is unlikely to change the classification. An alternative strategy is to focus on documents whose most likely classification is close to the second most likely classification. For fixed θ, let

$$\hat{y}_1 = \arg\max_y P_\theta(y|x)$$

and

$$\hat{y}_2 = \arg\max_{y \neq \hat{y}_1} P_\theta(y|x)$$

be the most likely and second most likely classification for a document. The marginal sampling policy chooses the document x where the two highest classification proba-

bilities are the closest. We state this policy using

$$x^M = \arg\min_x \big(P_\theta(\hat{y}_1|x) - P_\theta(\hat{y}_2|x)\big).$$

The previous two policies look at the level of uncertainty indicated by the most certain document or the difference between the two most certain documents. An idea that takes this thinking a step further uses entropy as a measure of uncertainty. The entropy maximization policy is computed using

$$x^H = \arg\max_x -\sum_i P_\theta(y_i|x) \log P_\theta(y_i|x)$$

where y_i represents a single classification. Entropy looks at all the potential classifications, and represents an information-theoretic measure of the information required to encode the classification probability distribution.

Empirical research with these policies has produced mixed results. Clearly there will be differences in behavior as the number of potential classifications changes. Not surprisingly, performance depends on the true utility function. The margin and level of confidence policies work better when the goal is to get the classification right, while the entropy maximization policy works if the objective is to minimize the log of the loss from an incorrect classification ("log-loss").

12.2.2 Query by committee

Imagine that we have several competing models for estimating the classification of a document. We would refer to this family of models as a "committee," where $c \in \mathcal{C}$ is a particular model in the set $\mathcal{C}$. Let $P_{\theta^{(c)}}(y_i|x)$ be the probability that model c (parameterized by $\theta^{(c)}$) returns classification y_i given the attributes x of a document.

There are several ways to perform active learning in this setting. One is to let each model vote for a classification. We might record a vote using the indicator function

$$I_c(y_i|x) = \begin{cases} 1 & \text{if } y_i = \arg\max_i P_{\theta^{(c)}}(y_i|x) \\ 0 & \text{otherwise.} \end{cases}$$

The indicator function $I_c(y_i|x)$ simply captures if model c thinks that y_i is the most likely classification. We can then count the number of votes using

$$V(y_i) = \sum_{c \in \mathcal{C}} I_c(y_i|x).$$

Alternatively, we could compute "soft votes" using

$$V(y_i) = \sum_{c \in \mathcal{C}} P_{\theta^{(c)}}(y_i|x).$$

We can then choose to sample the document with the highest *vote entropy*, giving us the policy

$$x^{VE} = \arg\max_x -\sum_i \frac{V(y_i)}{C} \log \frac{V(y_i)}{C}.$$

Another measure is the Kullback-Leibler divergence metric, which is a way of measuring the differences between two distributions. We first compute an average probability of classification across the competing models using

$$P_{\mathcal{C}}(y_i|x) = \frac{1}{|\mathcal{C}|}\sum_{c\in\mathcal{C}} P_{\theta^{(c)}}(y_i|x).$$

The KL divergence is then computed using

$$D(P_{\theta^{(c)}}\|P_{\mathcal{C}}) = \sum_i P_{\theta^{(c)}}(y_i|x)\log\frac{P_{\theta^{(c)}}(y_i|x)}{P_{\mathcal{C}}(y_i|x)}.$$

The Kullback-Leibler divergence is a type of distance metric since it is measuring the degree of similarity between two probability distributions: the distribution (across potential classifications) obtained when we average all the probabilities across the competing models, versus the distribution produced by each model. The quantity $D(P_{\theta^{(c)}}\|P_{\mathcal{C}})$ is the KL divergence for a particular model c. Our policy for deciding which document to evaluate is obtained by maximizing the average KL divergence across all the models, given by

$$x^{KL} = \arg\max_x \frac{1}{C}\sum_{c\in\mathcal{C}} D(P_{\theta^{(c)}}\|P_{\mathcal{C}}).$$

The idea with this policy is to choose to evaluate the document with the greatest disagreement among the different models. If the competing models largely agree, then it is unlikely that more information will change this consensus. Additional information is likely to have the greatest impact when competing models disagree.

12.2.3 Expected error reduction

An interesting policy uses an estimate of the degree to which information reduces the likelihood of being incorrect. We are going to assume that we have a set of unlabeled documents $\mathcal{U}$, which means that we have not solicited a classification from a domain expert. Assume we are considering the possibility of collecting information on the document x. We do not know its classification, but our current estimate of the probability that it will be classified as y_i is $P_\theta(y_i|x)$. If we choose document x and observe the classification y_i, we can use this information to update our classification probability, which we represent using $P_{\theta^{+(x,y_i)}}(y|x)$. This is analogous to a posterior belief.

Now, we are going to use this updated probability model on each document $u \in \mathcal{U}$, with feature vector x^u. We let

$$\hat{y}^u = \arg\max_y P_\theta(y|x^u)$$

be the most likely classification for a particular document x^u in our set $\mathcal{U}$. If $P_{\theta^{+(x,y_i)}}(\hat{y}^u|x^u)$ is the probability of this most likely classification, then $1 - P_{\theta^{+(x,y_i)}}(\hat{y}^u|x^u)$ represents the probability that the document u does not belong to the class that we think is the most likely. This is analogous to our posterior

belief about how likely we are to make a mistake about document u after observing a result for x. We then choose x to minimize this probability of error in expectation over all possible classifications y_i of document x. The policy, then, is given by

$$x^{ER} = \arg\min_x \sum_i P_\theta(y_i|x) \left(\sum_{u \in \mathcal{U}} \left(1 - P_{\theta^{+(x,y_i)}}(\hat{y}^u|x^u)\right) \right).$$

12.3 A VARIANCE MINIMIZING POLICY

We are going to describe an information collecting policy with the goal of reducing variance, but this time we are going to address a richer set of models and issues than we encountered in Section 12.1 when the goal was to minimize variance measures of the regression vector θ.

We start by assuming that there exists a model $y(x) = f(x) + \epsilon$ where $f(x)$ is the true model and ϵ is a source of experimental noise over which we have no control. Our goal is to collect a training dataset $\mathcal{D}^n = \left\{(x^0, y^1), (x^1, y^2), \ldots, (x^{n-1}, y^n)\right\}$. We group the choice $(x^0, \ldots, x^{n-1})$ of what to measure with the corresponding observations $(y^1, \ldots, y^n)$ into a single training dataset. We note that the standard notation in statistics is to let x^1 be the experiment that produces y^1, but as the development below illustrates, it is cleaner to let the superscript capture the information content; when we choose x^m, y^{m+1} is a random variable.

We assume we are using a sequential policy to determine $x^0, x^1, x^2, \ldots$ (starting with x^0) which may depend on the outcomes $y^1, y^2, \ldots$. This means that $\mathcal{D}^n$ is a random set, constructed sequentially. We use this information to fit an approximation $\bar{y}^n = \bar{f}^n(x|\mathcal{D}^n)$. Below, we are going to write the prediction as $\bar{y}^n(x|\mathcal{D}^n)$ to express its dependence on the query point x, and the data $\mathcal{D}^n$.

The goal is to design a policy that minimizes the variance in our predictions $\bar{y}^n(x|\mathcal{D}^n)$, but this depends on the points x where we *might* query the function. We did not have to deal with this issue in Section 12.1, but now we are going to assume that we are given a distribution $P(x)$ that gives the probability that we will want an estimate of the function at x. In practice, $P(x)$ may be chosen to be uniform over some region, or we may specify a normal distribution with some mean μ_x and a spread σ_x. Alternatively, we may have a sequence of observations $\tilde{x}^1, \ldots, \tilde{x}^k$ from history which can serve as a probability distribution. Either way, $P(x)$ serves as a weighting function that tells us the region in which we are interested. However, it is important to recognize that the probability that x is in the random set $\mathcal{D}^n$, which is influenced by our learning policy, may be completely different than $P(x)$.

If we make an observation at x, we are going to observe

$$y(x) = f(x) + \epsilon,$$

where $f(x)$ is our true (but unknown) function, and ϵ is the inherent noise in the observation. Our approximation is going to give us the estimate $\bar{y}^n(x|\mathcal{D}^n)$. A reasonable goal is to design a policy for choosing an experiment x to minimize the prediction error $(\bar{y}^n(x|\mathcal{D}^n) - y(x))^2$ for a single realization of $y(x)$ (which is random

because of ϵ), and a single estimate $\bar{y}^n(x|\mathcal{D}^n)$ from a dataset $\mathcal{D}^n$ (which is random because $\mathcal{D}^n$ is random). To formalize this idea, we need to take expectations. We let $\mathbb{E}_T(\cdot)$ be the total expectation over the observation noise imbedded in $y(x)$ and the observation of $\mathcal{D}^n$. These are independent, so we can write them as

$$\mathbb{E}_T(\bar{y}^n(x|\mathcal{D}^n) - y(x))^2 = \mathbb{E}_y\mathbb{E}_{\mathcal{D}}(y(x) - \bar{y}^n(x|\mathcal{D}^n))^2, \tag{12.5}$$

where $\mathbb{E}_{\mathcal{D}}$ is the expectation over all the possible outcomes of $\mathcal{D}^n$, and $\mathbb{E}_y$ is the expectation over all possible realizations of $y(x)$. We can break down the expected total variation (for a given x) in (12.5) using

$$\begin{aligned}\mathbb{E}_T(\bar{y}^n(x|\mathcal{D}^n) - y(x))^2 &= \mathbb{E}[(y(x) - \mathbb{E}[y(x)])^2] \\ &\quad + \big(\mathbb{E}_{\mathcal{D}}[\bar{y}(x|\mathcal{D}^n)] - \mathbb{E}[y(x)]\big)^2 \\ &\quad + \mathbb{E}_{\mathcal{D}}\big[(\bar{y}(x|\mathcal{D}^n) - \mathbb{E}_{\mathcal{D}}[\bar{y}(x|\mathcal{D}^n)])^2\big].\end{aligned} \tag{12.6}$$

The first term reflects the pure noise due to ϵ, which will not be affected by the learning policy. The second term captures the bias in the model, which is purely a function of the structural form of the underlying model, as well as the choice of x. Again, this is not affected by the choice of $\mathcal{D}^n$. The right-hand side of equation (12.6) is also known as a *bias-variance decomposition.*

The third term is the variance due to the estimation of the model from the training data. This is where we capture the variation in the estimated model (after n observations) due to the different variations in $\mathcal{D}^n$ that might be produced by following a specific learning policy. The variations in $\mathcal{D}^n$ arise because of differences in realizations in the observations $y^1, \ldots, y^n$ that lead to different decisions $x^2, \ldots, x^n$. This is the term that we wish to minimize by choosing a good learning policy.

For compactness, we are going to let

$$\sigma_y^{2,n}(x) = \mathbb{E}_{\mathcal{D}}\big(\bar{y}^n(x|\mathcal{D}^n) - \mathbb{E}_{\mathcal{D}}[\bar{y}^n(x|\mathcal{D}^n)]\big)^2.$$

Now assume that we are considering the possibility of adding (x^n, y^{n+1}) to create an expanded dataset

$$\mathcal{D}^{n+1} = \mathcal{D}^n \cup (x^n, y^{n+1}).$$

The choice x^n is a deterministic quantity (given $\mathcal{D}^n$) that we are thinking of measuring, while y^{n+1} is the random observation that we have not yet observed when we choose x^n. Let $\mathbb{E}_{\mathcal{D}}^n$ be the conditional expectation given $\mathcal{D}^n$. The choice of observation x^n is a deterministic function of $\mathcal{D}^n$. The new information in $\mathcal{D}^{n+1}$ is y^{n+1}. If we choose x^n, let $\tilde{\sigma}_y^{2,n}(x|x^n)$ be the conditional variance (given $\mathcal{D}^n$) of our estimate if we choose x^n, which is given by

$$\tilde{\sigma}_y^{2,n}(x|x^n) = \mathbb{E}_{\mathcal{D}}^n\big(\bar{y}^{n+1}(x|\mathcal{D}^{n+1}) - \mathbb{E}_{\mathcal{D}}^n[\bar{y}^{n+1}(x|\mathcal{D}^{n+1})]\big)^2. \tag{12.7}$$

The variance $\tilde{\sigma}_y^{2,n}(x|x^n)$ is defined in exactly the same way as $\tilde{\sigma}$ in (2.9). However, in Chapter 2, we also proved that this quantity coincided with the change in variance between two experiments, under a Bayesian learning model. This equivalence does *not* hold in the frequentist model we are considering here. In (12.7), $\tilde{\sigma}^{2,n}$ represents only the conditional variance of our estimate given a choice of experiment, and

minimizing this quantity is equivalent to minimizing the future variance of our prediction (similar to the posterior variance considered in the Bayesian models). If we were to adopt a Bayesian approach here, $\mathbb{E}[y(x)]$ would itself be random (since we do not know the true mean of the observation), and we would need to consider all the terms in (12.6) together in order to minimize the posterior variance.

In the frequentist model, it is enough to minimize the last term in (12.6). We want to choose x^n so as to minimize (12.7). However, we do not know that we are going to want to observe the function at x, so we need to take the expectation over potential observations using our marginal distribution $P(x)$, giving us

$$\bar{\sigma}_y^{2,n}(x^n) = \int_x \tilde{\sigma}_y^{2,n}(x|x^n)P(x)dx. \tag{12.8}$$

The variance $\bar{\sigma}_y^{2,n}(x^n)$ is the expected variance given that we run the experiment x^n. Our policy will be to choose the value x^n that produces the greatest reduction in variance. We state this policy using

$$x^n = \arg\min_{x'} \bar{\sigma}_y^{2,n}(x'). \tag{12.9}$$

The value of this policy is that it is fairly easy to compute. We illustrate the calculations for an approximation architecture that uses mixtures of Gaussians.

12.4 MIXTURES OF GAUSSIANS

In Section 12.1, we considered the case of estimating a single regression function. Here, we turn to a more general approximation architecture where we assume that observations come from one of a set of populations that we index $i = 1, \ldots, N$. We assume that each population produces a different behavior that can be reasonably approximated by a line, as illustrated in Figure 12.2.

12.4.1 Estimating parameters

We need to estimate the mean and variance of the experiments (independent variables) x, given by $\mu_{x,i}$ and $\sigma^2_{x,i}$, the mean and variance of the observations y, given by $\mu_{y,i}$ and $\sigma^2_{y,i}$, and the covariance between x and y, $\sigma_{xy,i}$. Temporarily assume that for each population i, we know the expected value of the observation x, $\mu_{x,i}$, and its variance $\sigma^2_{x,i}$. We are going to begin by presenting calculations for the remaining moments, after which we show how to compute $\bar{\sigma}^2(x^n)$.

We can represent the joint distribution of x and y for a population i. First define

$$z = \begin{bmatrix} x \\ y \end{bmatrix}, \ \mu_i = \begin{bmatrix} \mu_{x,i} \\ \mu_{y,i} \end{bmatrix}, \ \Sigma_i = \begin{bmatrix} \sigma^2_{x,i} & \sigma_{xy,i} \\ \sigma_{xy,i} & \sigma^2_{y,i} \end{bmatrix}.$$

We can now write the joint distribution of x and y for group i using

$$P(x, y|i) = \frac{1}{2\pi\sqrt{|\Sigma_i|}} \exp\left[-\frac{1}{2}(z-\mu_i)^T\Sigma_i^{-1}(z-\mu_i)\right].$$

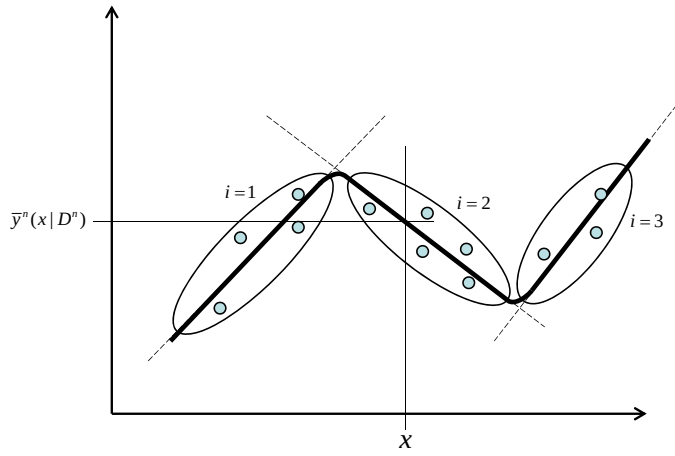


Figure 12.2 Illustration of fitting data using mixtures of Gaussians (based on Cohn et al. 1994).

We will not know the true means and variances, but we can estimate them from an initial sample.

The conditional variance of y given x is given by

$$\sigma^2_{y|x,i} = \sigma^2_{y,i} - \frac{\sigma_{xy,i}}{\sigma^2_{x,i}}.$$

The conditional expectation $\bar{y}^n_i(x|\mathcal{D}^n)$ and variance $\sigma^2_{\hat{y},i}$ given x are

$$\begin{aligned} \bar{y}^n_i(x|\mathcal{D}^n) &= \mu_{y,i}(x) + \frac{\sigma_{xy,i}}{\sigma^2_{x,i}}(x - \mu_{x,i}), \\ \sigma^2_{y,i} &= \frac{\sigma^2_{y|x,i}}{n_i}\left(1 + \frac{(x-\mu_{x,i})^2}{\sigma^2_{x,i}}\right). \end{aligned}$$

The scalar n_i can be viewed as the weight that group i should be given, calculated using

$$n_i = \sum_{j=1}^{m} \frac{P(x_j, y_j|i)}{\sum_{k=1}^{N} P(x_j, y_j|k)}.$$

We next compute the probability that population i contributes to the observation corresponding to x using

$$h_i(x) = \frac{P(x|i)}{\sum_{j=1}^{N} P(x|j)},$$

where

$$P(x|i) = \frac{1}{\sqrt{2\pi\sigma^2_{x,i}}} \exp\left[-\frac{(x-\mu_{x,i})^2}{2\sigma^2_{x,i}}\right]. \tag{12.10}$$

Given the observation x, the expectation across all the populations of $\bar{y}_i^n(x|\mathcal{D}^n)$ and its variance are given by

$$\begin{aligned}\bar{y}^n(x|\mathcal{D}^n) &= \sum_{i=1}^{N} h_i(x)\bar{y}_i^n(x|\mathcal{D}^n),\\ \sigma_y^2(x|\mathcal{D}^n) &= \sum_{i=1}^{N} \frac{h_i^2\sigma_{y|x,i}^2}{n_i}\left(1+\frac{(x-\mu_{x,i})^2}{\sigma_{x,i}^2}\right).\end{aligned}$$

We now have the foundation we need to estimate the expected value of an additional experiment.

12.4.2 Active learning

We have to calculate the variance given an experiment x^n, where we continue to assume that $P(x)$ is known. We have assumed that $\mu_{x,i}$ and $\sigma_{x,i}^2$ are known for each population i, but we cannot derive these statistics directly from the marginal distribution $P(x)$. For example, we may approximate $P(x)$ by assuming that it is uniformly distributed over some region, or we may represent it from a sample obtained from history.

We could try to estimate $\mu_{x,i}$ and $\sigma_{x,i}^2$ from training data, but these observations are chosen according to our learning policy, and may not reflect the true distribution that reflects the likelihood of actually needing to calculate the function at some x. If we use the training sample, we are effectively using the joint distribution $P(x^n, y|i)$ instead of $P(x, y|i)$. We can correct for the difference in the sampling distribution (at least in principle) using

$$P(x, y|i) = P(x^n, y|i)\frac{P(x|i)}{P(x^n|i)}.$$

We compute the conditional distribution $P(x|i)$ by using equation (12.10) with the mean and variance $\mu_{x,i}$ and $\sigma_{x,i}^{2,n}$ computing using data sampled from $P(x)$ or a reference sample. The distribution $P(x^n, y|i)$ is also computed using equation (12.10) with mean and variance calculated from a training dataset.

We now have to compute $\bar{\sigma}^{2,n}(x^n)$ which we use in equation (12.9) to determine the next point to measure. We first calculate $\tilde{\sigma}_y^{2,n}(x|x^n)$ for a particular query point x. The distribution of y^{n+1} given x^n can be calculated as a mixture of normal distributions, given by

$$\begin{aligned}P(y^{n+1}|x^n) &= \sum_{i=1}^{N} h_i(x^n)P(y^{n+1}|x^n, i)\\ &= \sum_{i=1}^{N} h_i(x^n)\mathcal{N}\big(\bar{y}_i^n(x^n|\mathcal{D}^n), \sigma_{y|x,i}^{2,n}(x^n)\big),\end{aligned}$$

where $\mathcal{N}(\mu, \sigma^2)$ represents the normal distribution, where $\bar{y}_i^n(x^n|\mathcal{D}^n)$ is the mean of y^{n+1} based on data for group i, and $\sigma_{y|x,i}^2(x^n)$ is the variance. The conditional

variance of our prediction of x given the choice of x^n can be found using

$$\tilde{\sigma}_y^{2,n}(x|x^n) = \sum_{i=1}^{n} \frac{h_i^2(x)\tilde{\sigma}_{y|x,i}^{2,n}(x|x^{n+1})}{n_i + h_i(x|x^{n+1})} \left(1 + \frac{(x-\mu_{x,i})}{\sigma_{x,i}^2}\right),$$

where

$$\begin{aligned}
\tilde{\sigma}_{y,i}^{2,n}(x|x^{n+1}) &= \frac{n_i\sigma_{y,i}^{2,n}}{n_i + h_i(x|x^{n+1})} \\
&\quad + \frac{n_i + h_i(x|x^{n+1})\left(\sigma_{y|x,i}^2 + (\bar{y}_i^{n+1}(x|x^{n+1}) - \mu_{y,i})^2\right)}{(n_i + h_i(x|x^{n+1}))^2}, \\
\tilde{\sigma}_{y|x,i}^{2,n}(x|x^{n+1}) &= \tilde{\sigma}_{y,i}^{2,n}(x|x^{n+1}) - \frac{\tilde{\sigma}_{xy,i}^{2,n}}{\sigma_{x,i}^2}, \\
\tilde{\sigma}_{xy,i}(x|x^{n+1}) &= \frac{n_i\sigma_{xy,i}}{n_i + h_i(x|x^{n+1})} \\
&\quad + \frac{n_i h_i(x|x^{n+1})(x^{n+1} - \mu_{x,i})(\bar{y}_i^{n+1}(x|x^{n+1}) - \mu_{y,i})}{(n_i + h_i(x|x^{n+1}))^2}, \\
\tilde{\sigma}_{xy,i}^2(x|x^{n+1}) &= (\tilde{\sigma}_{xy,i})^2 + \frac{n_i^2(h_i(x|x^{n+1}))^2\sigma_{y|x^{n+1},i}^2(x^{n+1} - \mu_{x,i})^2}{\cdot}
\end{aligned}$$

If we are estimating $\mu_{x,i}$ and $\sigma_{x,i}^2(x|x^{n+1})$ from data, we can use the following equations

$$\begin{aligned}
\bar{\mu}_{x,i}^{n+1}(x^{n+1}) &= \frac{n_i\bar{\mu}_{x,i}^n + h_i(x|x^{n+1})x^{n+1}}{n_i + h_i(x|x^{n+1})}, \\
\sigma_{x,i}^{2,n+1}(x|x^{n+1}) &= \frac{n_i\sigma_{x,i}^{2,n}}{n_i + h_i(x|x^{n+1})} + \frac{n_i h_i(x|x^{n+1})(x^{n+1} - \bar{\mu}_{x,i}^n)^2}{(n_i + h_i(x|x^{n+1}))^2}.
\end{aligned}$$

With $\tilde{\sigma}_y^{2,n}(x|x^n)$ in hand, we use equation (12.8) to integrate over the different query points x to obtain $\bar{\sigma}_y^{2,n}(x^n)$.

12.5 BIBLIOGRAPHIC NOTES

Section 12.1 - This section covers classic material from experimental design as it is known within the statistics community (see DeGroot 1970, Wetherill & Glazebrook 1986, Montgomery 2008).

Section 12.2 - This section is based on Settles (2009).

Section 12.3 - This section is from Geman et al. (1992), with material from Cohn et al. (1994), Cohn et al. (1996) and Settles (2009). See Hastie et al. (2005) for a nice discussion of bias-variance decomposition.

12.5.1 Optimal computing budget allocation

The value of the indifference zone strategy is that it focuses on achieving a specific level of solution quality, being constrained by a specific budget. However, it is often the case that we are trying to do the best we can within a specific computing budget. For this purpose, a line of research has evolved under the name *optimal computing budget allocation*, or OCBA.

Figure 12.3 illustrates a typical version of an OCBA algorithm. The algorithm proceeds by taking an initial sample $N_x^0 = n_0$ of each alternative $x \in \mathcal{X}$, which means we use $B^0 = Mn_0$ experiments from our budget B. Letting $M = |\mathcal{X}|$, we divide the remaining budget of experiments $B - B^0$ into equal increments of size Δ, so that we do $N = (B - Mn_0)\Delta$ iterations.

After n iterations, assume that we have measured alternative x N_x^n times, and let W_x^m be the mth observation of x, for $m = 1, \ldots, N_x^n$. The updated estimate of the value of each alternative x is given by

$$\theta_x^n = \frac{1}{N_x^n} \sum_{m=1}^{N_x^n} W_x^m.$$

Let $x^n = \arg\max \theta_x^n$ be the current best option.

After using Mn_0 observations from our budget, at each iteration we increase our allowed budget by $B^n = B^{n-1} + \Delta$ until we reach $B^N = B$. After each increment, the allocation $N_x^n,\ x \in \mathcal{X}$ is recomputed using

$$\frac{N_x^{n+1}}{N_{x'}^{n+1}} = \frac{\hat{\sigma}_x^{2,n}/(\theta_{x^n}^n - \theta_{x'}^n)^2}{\hat{\sigma}_{x'}^{2,n}/(\theta_{x^n}^n - \theta_{x'}^n)^2} \quad x \neq x' \neq x^n, \tag{12.11}$$

$$N_{x^n}^{n+1} = \hat{\sigma}_{x^n}^n \sqrt{\sum_{i=1, i\neq x^n}^{M} \left(\frac{N_x^{n+1}}{\hat{\sigma}_i^n}\right)^2}. \tag{12.12}$$

We use equations (12.11)-(12.12) to produce an allocation N_x^n such that $\sum_x N_x^n = B^n$. Note that after increasing the budget, it is not guaranteed that $N_x^n \geq N_x^{n-1}$ for some x. If this is the case, we would not measure these alternatives at all in the next iteration. We can solve these equations by writing each N_x^n in terms of some fixed alternative (other than x^n), such as N_1^n (assuming $x^n \neq 1$). After writing N_x^n as a function of N_1^n for all x, we then determine N_1^n so that $\sum N_x^n \approx B^n$ (within rounding).

The complete algorithm is summarized in figure 12.3.

Step 0. Initialization:

Step 0a. Given a computing budget B, let n^0 be the initial sample size for each of the $M = |\mathcal{X}|$ alternatives. Divide the remaining budget $T - Mn_0$ into increments so that $N = (T - Mn_0)/\delta$ is an integer.

Step 0b. Obtain samples W_x^m, $m = 1, \ldots, n_0$ samples of each $x \in \mathcal{X}$.

Step 0c. Initialize $N_x^1 = n_0$ for all $x \in \mathcal{X}$.

Step 0d. Initialize $n = 1$.

Step 1. Compute

$$\theta_x^n = \frac{1}{N_x^n} \sum_{m=1}^{N_x^n} W_x^m.$$

Compute the sample variances for each pair using

$$\hat{\sigma}_x^{2,n} = \frac{1}{N_x^n - 1} \sum_{m=1}^{N_x^n} (W_x^m - \theta_x^n)^2 .$$

Step 2. Let $x^n = \arg\max_{x \in \mathcal{X}} \theta_x^n$.

Step 3. Increase the computing budget by Δ and calculate the new allocation $N_1^{n+1}, \ldots, N_M^{n+1}$ so that

$$\begin{aligned} \frac{N_x^{n+1}}{N_{x'}^{n+1}} &= \frac{\hat{\sigma}_x^{2,n}/(\theta_{x^n}^n - \theta_{x'}^n)^2}{\hat{\sigma}_{x'}^{2,n}/(\theta_{x^n}^n - \theta_{x'}^n)^2} \quad x \neq x' \neq x^n, \\ N_{x^n}^{n+1} &= \hat{\sigma}_{x^n}^n \sqrt{\sum_{i=1, i \neq x^n}^{M} \left(\frac{N_x^{n+1}}{\hat{\sigma}_i^n} \right)^2}. \end{aligned}$$

Step 4. Perform $\max\left(N_x^{n+1} - N_x^n, 0\right)$ additional simulations for each alternative x.

Step 5. Set $n = n + 1$. If $\sum_{x \in \mathcal{X}} N_x^n < B$, go to step 1.

Step 6. Return $x^n \arg\max_{x \in \mathcal{X}} \theta_x^n$.

Figure 12.3 Optimal computing budget allocation procedure.

CHAPTER 13

LEARNING IN MATHEMATICAL PROGRAMMING

There are applications where we have to collect information to be used in a larger optimization problem such as a shortest path problem or linear program. Ranking and selection can be viewed as the solution to a simple linear program

$$v^n = \max_x \sum_{i=1}^{M} \mu_i x_i$$

subject to

$$\begin{aligned} \sum_{i=1}^{M} x_i &= 1 \\ x_i &\geq 0. \end{aligned}$$

The solution to this linear program requires sorting (μ_i) for all i and choosing the index $i*$ with the largest value of μ_i, which gives us a solution where $x_{i*} = 1$, and $x_i = 0,\ i \neq i*$. Our learning challenge is to choose an element j to measure, producing a value $v^{n+1}(j)$ computed using the vector μ_i. The goal is to choose the index j that maximizes the expected value of v^{n+1}. Note for this discussion that we are using indices i and j for our choices, and we are switching (for this chapter) to x as the vector of implementation decisions.

Optimal Learning. By Warren B. Powell and Ilya O. Ryzhov

We can take this perspective one step further. Assume now that we have a more general linear program which we write as

$$\min_x \sum_{i \in \mathcal{I}} c_i^T x_i$$

subject to

$$\begin{aligned} Ax &= b, \\ x_i &\geq 0, \quad i \in \mathcal{I}. \end{aligned}$$

Here, x is a vector with potentially hundreds or even tens of thousands of dimensions. More significantly, the A matrix might be general, although there are specific problem structures that are of interest to us.

Linear programming has numerous applications, including a broad class of problems on networks, where the matrix A is used to represent flow conservation equations. The objective can be to minimize the travel distance across the graph (shortest-path problems), or to efficiently move resources between nodes (such as the "transportation problem" or general network flow problems). Many other optimization problems can also be expressed as linear programs. A classic example is the problem of formulating a production plan to maximize profit. A company produces M products, and wishes to produce x_i units of product $i = 1, ..., M$ to maximize the total profit $c^T x$ with c_i being the profit from selling one unit of product i. The products are made using J different resources, with b_j being the total amount of resource j available. The matrix A denotes the production constraints, with $A_{j,i}$ being the amount of resource j needed to create one unit of product i.

In the basic LP model, we assume that the parameters A, b and c are known. In reality, we are unlikely to know them exactly, just as we do not know μ in ranking and selection. Suppose that our company has developed a new product, and now needs to decide how much of it should be produced. We may have an estimate of c_i based on some preliminary data. Perhaps our sales figures for our old line of MP3 players may give us some idea of the profitability of the new line. At the same time, we are still quite uncertain about the true value of c_i. But now, suppose that we have the ability to collect additional information about c_i before we commit to a production plan x. Perhaps we have a chance to run a test market where we can get a sense of how well product i might perform. The results might change our estimate of c_i, thus affecting our final production plan. Given our limited time and money, which products should be included in the test market? Or, in other words, which information will help us make the best possible decision?

Optimal learning has a role to play here, but it is not a simple matter of picking up and applying the formulas and algorithms from Chapters 3 and 4. In mathematical programming applications, our *experimental decision* (say, measuring the coefficient c_i) is distinct from our *implementation decision*, which we now represent as the vector x. We may learn about a single product or a single region in the network, but our overall goal is to solve an optimization problem. The coefficients $i = 1, ..., M$ in our LP are a bit like the "alternatives" from Chapter 3, in that we have a choice of which coefficients to learn about, but we are not simply interested in finding the largest coefficient. The part should teach us about the whole; we should learn about

those coefficients that contribute the most to our ability to optimize. Running a test market for product i should lead us to a better production plan.

Although we cannot recycle the knowledge gradient formulas from Chapter 4 in a straightforward way, we can still apply the concept of the knowledge gradient. In this chapter, we show how the value of information can be expressed in problems with uncertain objective functions. After describing several applications, we illustrate the application of the knowledge gradient first in the context of simple shortest path problems, and then for more general linear programs.

13.1 APPLICATIONS

In this section, we introduce three models where the implementation decision requires solving a mathematical program. We begin with a model for piloting a hot air balloon, where the challenge is moving the balloon up and down to find the fastest wind. Given a current set of estimates about the wind, the balloon can solve a linear program to find the best path to the destination. Then, we describe a common problem in the investment community where it is necessary to visit different companies to learn about their potential for future profits. The challenge is determining which companies to visit (we do not have the time to visit all of them), after which we have to take estimates of expected returns, variances and covariances to determine the best portfolio. We close with a discussion of graph problems.

13.1.1 Piloting a hot air balloon

Consider the problem faced by a balloonist trying to get from one point to another in the presence of changing winds that can be different at different altitudes. Her ability to make progress depends on finding wind that moves in the right direction. She observes the speed and direction of the wind at her current location, but she may wonder if the conditions are better at a different altitude. Observing the wind at one altitude provides some information about the wind at other altitudes, as well as the wind at later points in time. But it takes time and (if she has to raise her altitude) energy to make these observations.

The model Optimizing the trajectory of a hot air balloon is a fairly difficult control problem, even when we do not include learning as a dimension. For our purposes, we are going to simplify the problem by assuming that our balloon can only move in two dimensions: forward and up or down. Our interest is in finding a path that balances distance against changing (and uncertain) wind speeds so that we get to the destination as quickly as possible.

Our view of the problem is depicted in Figure 13.1, where our balloon is currently at horizontal location x, vertical location y, and is faced with one of three decisions: d^{up} takes the balloon on an upward trajectory, d^{stay} holds the balloon at its current altitude while d^{down} takes the balloon on a downward trajectory. Although we would like to model energy expenditure, we are only going to capture the time required

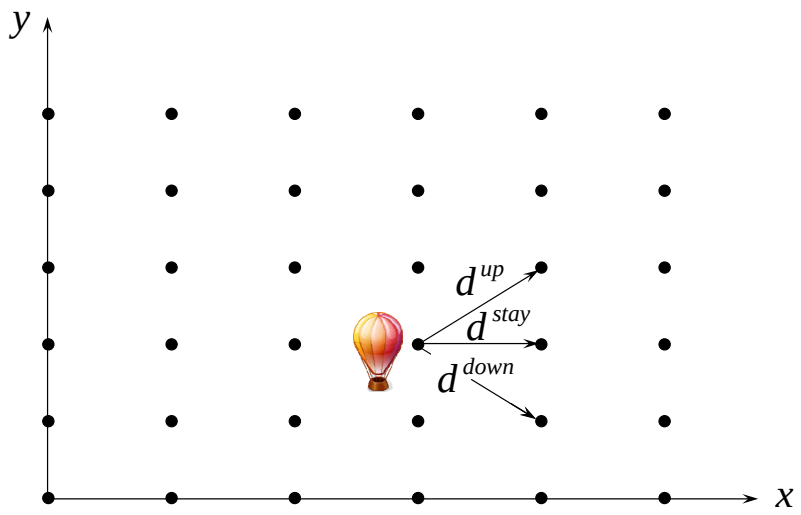


Figure 13.1 Configuration of a two-dimensional hot air balloon trajectory.

to complete each decision. We assume that the balloon moves 30 percent slower moving up, and 20 percent faster moving down.

The physical state $R^n = (x^n, y^n)$ captures the location of the balloon. Our belief state about the wind is given by $B^n = (\theta^n_{xy}, \sigma^{2,n}_{xy})_{x,y}$ where θ^n_{xy} is what we think the wind speed is at time n, location x, y, and $\sigma^{2,n}_{xy}$ is the variance in our distribution of belief. When the balloon is at location (x, y) at time n, the true wind speed is $\bar{\mu}^n_{xy}$ which we measure with noise according to

$$W^n_{xy} = \bar{\mu}^n_{xy} + \varepsilon^n.$$

We assume experiments are independent across space and time, and are normally distributed with mean 0 and variance λ^ϵ. The true wind speed also evolves over time according to

$$\theta^{n+1}_{xy} = \bar{\mu}^n_{xy} + \hat{\mu}^{n+1}_{xy},$$

where $\hat{\mu}^{n+1}_{xy}$ describes the actual change in the wind from time period to time period. The random changes in the wind, $\hat{\mu}^n_{xy}$, are correlated across both space and time, since if the wind is high at time n, location (x, y), then it is likely to be high at nearby locations, and at the same location at later points in time. The covariance between wind observations at two different altitudes at the same point in time is given by

$$Cov(W^n_{xy}, W^n_{xy'}) = \sigma^2_W e^{-\beta_y |y-y'|} + \lambda_{xy}.$$

Similarly, we assume the covariance between a location x and a location $x' > x$ at the same altitude is given by

$$Cov(W^n_{xy}, W^n_{xy'}) = \sigma^2_W e^{-\beta_x |x-x'|} + \lambda_{xy}.$$

We combine these two to give us a general covariance structure

$$Cov(W^n_{xy}, W^n_{xy'}) = \sigma^2_W e^{-\beta_y|y-y'|-\beta_x|x-x'|} + \lambda_{xy}.$$

Using this function, we can construct a covariance matrix Σ^n with elements $\Sigma^n_{xy,x'y'}$. We use this structure to initialize our covariance matrix Σ_0. Next let e_{xy} be a vector of 0's with a 1 corresponding to the location (x, y). The general formulas for updating the vector of estimates of the mean and covariance matrix for our distribution of belief about the wind is given by

$$\begin{aligned}
\theta^{n+1} &= \Sigma^{n+1}\left((\Sigma^n)^{-1}\theta^n + (\lambda_{xy})^{-1}W^{n+1}e_{xy}\right), \\
\Sigma^{B,n+1} &= ((\Sigma^n)^{-1} + (\lambda_{xy})^{-1}e_{xy}(e_{xy})')^{-1}.
\end{aligned}$$

This gives us the updated estimate of the mean and variance, where the variance takes into account the experimental error. But this ignores the changing velocity due to $\hat{\mu}^n$. When we take this into account, our updated covariance matrix is given by

$$\Sigma^{n+1} = \Sigma^{B,n+1} + \Sigma^{\mu}.$$

So, $\Sigma^{B,n+1}$ will shrink relative to Σ^n, but it will grow due to the effect of Σ^{μ}. We note that this is not a very realistic model for wind. For example, if we did not run any experiments (and possibly even if we do), the covariance matrix will grow without bound. However, this model will illustrate an interesting learning problem without becoming buried in the details of the physics.

Planning a trip The next step is to plan a trip given what we know now. We use our current estimates of speeds at each altitude. Without additional information, our best estimate of the speed in the future is the same as it is now, for each altitude, which means that $\theta^n_{xy} = \theta^{n'}_{x'y}$ for $n' \geq n$ and $x' \geq x$. Using the assumption that speeds are 30 percent slower going up and 20 percent faster going down, we can construct a speed graph over the entire distance, similar to what is depicted in Figure 13.2. If we convert speeds to travel times, we now have a simple shortest path problem to find the fastest path from the current location to the end (presumably at an altitude of 0).

This problem can be formulated as a linear program by defining the decision variable

$$u_{xy,x'y'} = \begin{cases} 1 & \text{if we decide to make the move from } (x,y) \text{ to } (x',y') \\ 0 & \text{otherwise.} \end{cases}$$

This entire vector is chosen at time n, and can be thought of as the current planned trajectory given what we know now. From a particular point (x, y), we assume that there are only three points that can be reached, by using the decisions up, down or stay. Let $c_{xy,x'y'}(\bar{\mu}^n)$ be the time required to move from (x, y) to (x', y'). Our linear program would be given by

$$F^n(\bar{\mu}^n) = \min_u \sum_{x,y} \sum_{x',y'} c_{xy,x'y'}(\bar{\mu}^n) u_{xy,x'y'} \tag{13.1}$$

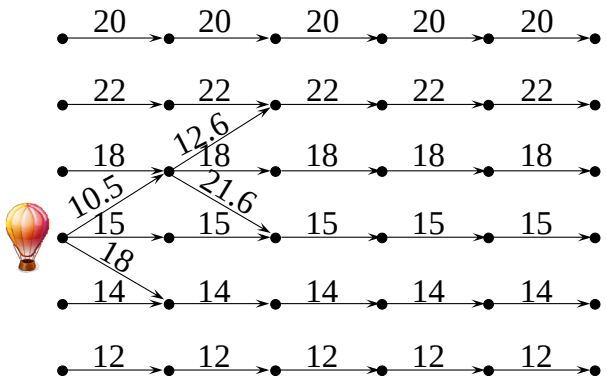


Figure 13.2 Speeds for each decision at each (x, y) location given the current belief state.

subject to

$$\sum_{x'y'} u_{xy,x'y'} = R_{0y}, \tag{13.2}$$

$$\sum_{x'y'} u_{xy,x'y'} - \sum_{xy} u_{xy,x'y'} = 0, \tag{13.3}$$

$$u_{xy,x'y'} \geq 0. \tag{13.4}$$

Here, $R_{0y} = 1$ if the balloon starts the process at an altitude y. Equation (13.3) requires that if the balloon arrives to (x, y) then it must also leave. Equation (13.4) requires that the flows be nonnegative. We would also like the decision u_t to be 0 or 1, but we will get this as a natural result of the structure of the problem.

We are assuming that we are solving the problem deterministically, using our current estimates of the speeds. We could put probability distributions around the speeds and solve a stochastic optimization problem, but this would not contribute to our understanding of how to handle the learning aspects of the problem.

Learning while flying Imagine that the wind at your altitude has dropped significantly. You are, of course, wondering if the wind is better at a different altitude. If we move to (x', y'), we are going to observe $W^{n+1}_{x'y'}$. Because of the covariance structure, this observation will allow us to update our estimate of the entire vector θ^n giving us $\theta^{n+1}(W^{n+1}_{x'y'})$. This would allow us to solve the problem $F^{n+1}(\theta^{n+1}(W^{n+1}_{x'y'}))$ using our new belief state. We have three possible decisions (up, down or stay). If we want to use pure exploitation, we would solve the problem $F^n(\theta^n)$, which would tell us which action to take. But this ignores the potential value of learning. As an alternative, we could choose the best value of $(x'y')$ (corresponding to one of the three decisions) to solve

$$\max_{(x'y')} \mathbb{E}^n \left\{ F^{n+1}(\theta^{n+1}(W^{n+1}_{x'y'})) - F^n(\theta^n) \right\}. \tag{13.5}$$

We have written this function in the form of the knowledge gradient, recognizing that $F^n(\theta^n)$ is a constant (given what we know at time n) which does not affect the solution.

When we first introduced the knowledge gradient, we presented it in the context of ranking and selection where we had to choose from among a small, discrete set of choices. Now, we have to solve a linear program for each possible experiment, which means we can no longer compute the expectation exactly. But we can take advantage of the fact that we have a small number of potential experimental decisions (up, down or stay), which means we effectively face a ranking and selection problem. We cannot compute the expectation exactly, but we can use Monte Carlo methods. Let $W^{n+1}_{x'y'}(\omega)$ be a sample observation of the wind if we were to choose to go to location $(x'y')$. This is not a true observation - it is a simulated observation for the purpose of approximating the expectation.

Given this observation, we can update the mean and covariance matrix and solve the linear program. Note that this observation does not just change the speed out of node $(x'y')$ (take another look at Figure 13.2); it changes estimates of speeds over the entire network, partly because of the presence of correlations, and partly because we solve our linear program by assuming that as we step forward in time, we do not collect additional information (which means that speeds at the same altitude for larger values of x' are the same). Let $F^{n+1}(x'y'|\omega)$ be the resulting "sample observation" of $\mathbb{E}^n F^{n+1}(\theta^{n+1}(W^{n+1}_{x'y'}))$. We could then repeat this, say, 100 times, for each possible decision and take an average. We could then make the decision that has the highest estimated value.

Sound familiar? This is precisely the ranking and selection problem. Solving the linear program 100 times and taking an average is a bit clumsy (famously known as the "brute force" solution). A more elegant approach would be to use (drum roll please) the knowledge gradient algorithm to choose the best decision. This would allow us to avoid running 100 experiments of a choice that does not look promising. We can build a prior by initially evaluating each choice assuming that we do not learn anything (the wind does not change from the prior), so that we start with a solution built around the pure exploitation policy.

Optimal learning versus stochastic optimization It is useful to contrast our learning algorithm versus what we might have done if we were solving this as a stochastic optimization problem. In a stochastic optimization model, we would acknowledge that we do not know what wind we will face when we arrive to a location $(x'y')$ (above, we solved the problem deterministically). For example, a deterministic solution might take us closer to the ground if the wind seems to be fastest there. However, a stochastic solution might recognize that the wind near the ground might be much lower than expected, and if we are close to the ground, the only place to go is up (at a high cost). A stochastic solution to this problem is relatively difficult, and probably requires the techniques of approximate dynamic programming.

The issue of whether to use a deterministic or stochastic algorithmic strategy did not arise with the simpler problems such as ranking and selection, because the optimal solution was to pick the index $j*$ with the highest expected value of θ^N_j. Our stochastic shortest path problem, however, is different because it involves finding a

path over a sequence of steps with uncertain costs. Even if we fix our distribution of belief, it is still a stochastic optimization problem.

The essential difference between a stochastic optimization algorithm and a learning algorithm is that with a learning algorithm, we accept that observations of the wind (the external experiment) come from an exogenous distribution that may be different from our true distribution of belief. As a result, we use these observations to update our distribution of belief. With a classical stochastic optimization formulation, we acknowledge that we do not know the actual wind, but we do know the probability distribution, which means that observations do not change our distribution of belief about the distribution.

13.1.2 Optimizing a portfolio

Imagine that you are looking at investing in a group of companies. The companies are organized by industry segment (retailing, financial services, transportation), so we will refer to the j^{th} company within the i^{th} industry segment. We start by assuming that θ^0_{ij} is the expected return (over some period of time) for company j in segment i. We also let Σ^0 be the matrix that captures the covariances in our beliefs about μ_{ij}, where the diagonal elements $\Sigma_{ij,ij}$ represent the variance.

In general, we assume that the covariance between two companies has the structure

$$\Sigma_{ij,i'j'} = \sigma_0^2 + \sigma_i^2 1_{\{i=i'\}}.$$

Thus, σ_0^2 captures the common covariance (e.g. the extent to which companies respond to the general economy), and σ_i^2 captures the common covariance for companies within industry segment i.

Before we invest in a company, we have the ability to visit the company and learn more about the management team, marketing plans and facilities. We might assume that our observation W of each company is independent with variance λ_{ij}. Let $x_{ij} = 1$ if we visit company (i, j). If we choose to visit company (i, j), we can update the mean and variance using our standard formulas, giving us an updated $\theta^1_{ij}(x)$. If $x_{ij} = 0$, then $\theta^1_{ij}(x) = \theta^0_{ij}$, and we let $\Sigma^1(x)$ be the updated covariance matrix.

Given the vector x of visits (presumably chosen to a constraint on how many visits we can make), we then have to find an optimal portfolio. Let y_{ij} be the amount that we are going to invest in company (ij). We do this by solving the standard quadratic programming problem

$$F(x) = \mathbb{E}F(x, W) = \min_y \left(\theta^1(x)y + \theta y^T \Sigma^1(x) y\right) \tag{13.6}$$

subject to

$$\sum_{ij} y_{ij} = R^I, \tag{13.7}$$

$$y_{ij} \geq 0. \tag{13.8}$$

Here, x is our experimental decision, W captures what we observe if we choose to run the experiment, and y is our implementation decision. R^I is the amount of money

we have to invest. The learning problem is given by

$$\min_x \mathbb{E}F(x, W) \tag{13.9}$$

subject to

$$\sum_{ij} c_{ij}x_{ij} = R^M, \tag{13.10}$$

$$x_{ij} \geq 0 \text{ and integer.} \tag{13.11}$$

Here, R^M is our experimental budget. The learning problem is hard to solve, both because it requires integer solutions but primarily because of the problems computing the expectation in (13.9). We suggest simply finding the marginal value of making each visit, and then assigning visits in order of decreasing marginal value until the budget is exhausted. This is a type of batch ranking and selection problem, where instead of finding the best choice, we find the best set of choices.

Even this simple heuristic is hard to implement. Imagine that there are 100 companies that we are considering, and we have the budget to make 20 visits. What are the best 20 companies to visit? We are not able to compute the expectation $\mathbb{E}F(x, W)$ in (13.6) either. Instead, we can choose a company to visit, then randomly sample what we might learn $(W_{ij}(\omega))$. We can perform 100 samples of each company and take an average to obtain an estimate of the value of visiting each company, but this may be computationally demanding (and 100 samples may not be enough). Or, we can use the knowledge gradient algorithm to allocate our computational budget across the 100 companies.

13.1.3 Network problems

Optimization on a graph is a well-known class of mathematical programming problems, with applications in transportation, project management, telecommunication and other areas. A graph is described by a set $\mathcal{V}$ of nodes (or "vertices") and a set $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ of edges, where each edge connects two nodes. Figure 13.3 provides a visual illustration. It is easiest to think of a graph as a map of physical locations: for example, nodes could represent cities and edges could stand for major highways connecting those cities. An edge $(i, j) \in \mathcal{E}$ between nodes i and j carries a cost (or "length") c_{ij}. The cost can be expressed in terms of money or time. For example, we might focus on the economic cost of transporting a shipment of goods from one city to another, or on the time needed for the shipment to reach its destination.

Implementation decisions in networks The shipment problem is a well-studied application of *minimum-cost flows*, a model for optimizing the route of resources across a network. A number b_i represents either the supply or demand for a commodity at node i. If $b_i > 0$, then i is a supply node (perhaps we have a production facility in city i, or we purchase from a manufacturer located in this city). If $b_i < 0$, then i is a demand node (perhaps retailers in this city are selling our products). If $b_i = 0$, the node is a transit node and our product can be transported through it en route to a demand node.

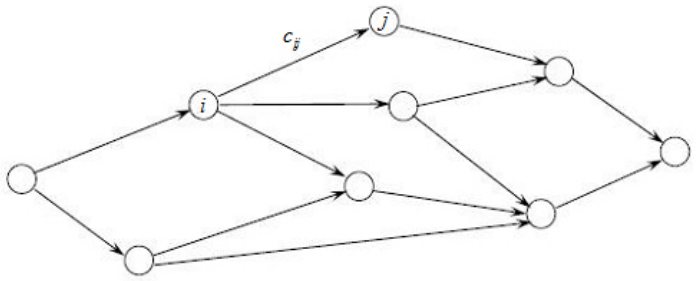


Figure 13.3 A generic network, with nodes i and j connected by an edge of length c_{ij}.

The cost of transporting a single unit of product from node i to node j is given by c_{ij}. The objective function can be written as

$$\min_x \sum_{(i,j)\in\mathcal{E}} c_{ij}x_{ij},$$

where x_{ij} represents the quantity of product that we wish to ship from i to j. Obviously we require $x_{ij} \geq 0$ for all i, j.

Each node is subject to a flow conservation constraint. Product can only enter the network through a supply node and leave through a demand node. Thus, the total product leaving node j has to be equal to the amount that entered node j, plus the amount that was supplied (or, minus the amount that was used to satisfy demand). We can write this constraint as

$$b_j + \sum_{i:(i,j)\in\mathcal{E}} x_{ij} = \sum_{i:(j,i)\in\mathcal{E}} x_{ji}, \qquad j \in \mathcal{V}.$$

The implementation decision in this problem is the flow schedule, the values of x_{ij} that we choose for all edges (i, j). The objective function and constraints are both linear in x_{ij}, making this a classic application of linear programming. There are many variations of the problem. For example, we might add capacity constraints $x_{ij} \leq d_{ij}$ if there is a cap on the amount that we can ship along a particular route (perhaps only a limited number of trucks is available). We may also add more flexibility to our model by giving ourselves the option to use only part of the available supply, in a situation where we have a choice of suppliers. In this case, for $b_j > 0$, we can add new decision variables $0 \leq y_j \leq b_j$ denoting the amount of supply we used, and replacing b_j with y_j in the corresponding flow constraints. We would also incorporate additional purchase costs into our objective function.

In some applications, we may not always know the costs c_{ij} exactly. In a supply chain, costs may depend on numerous factors. They are affected by purchase and production costs, but also by more vague factors like the reliability of a supplier. Perhaps a particular supplier has a high chance of experiencing shortages, or a particular transit route is more likely to encounter delays. Eventually, the shortages

and delays will be passed down to us, increasing our own costs. The distribution of the costs, or even their expected value, may be difficult to quantify, and has to be learned by doing business with a particular supplier or shipping along a particular transit route. We will need to model our beliefs about the costs and think about which suppliers we would like to experiment with in order to arrive at the best shipment schedule.

A special case of minimum-cost flows is the well-known shortest path problem. In this setting, there is a single supplier s with $b_s = 1$, and a single demand node t with $b_t = -1$. All other nodes are transit nodes, and the cost c_{ij} represents the "length" of edge (i, j). Our goal is to find a route from s to t with the shortest possible length. Like all the other network problems discussed here, the shortest-path problem can be formulated and solved as a linear program. There are also many optimization algorithms that are specifically tailored to shortest-path problems. We give one well-known example, the Bellman-Ford algorithm. Given a cost vector c and specified source and destination nodes s, t, the algorithm calculates the distance $V(i; c)$ from any node i to t in the following way:

1) Let $V(s; c) = 0$. For all other nodes $i \neq s$, let $V(i; c) = \infty$.
2) For each edge $(i, j) \in \mathcal{E}$, if $V(i; c) + c_{ij} < V(j; c)$, let $V(j; c) = V(i; c) + c_{ij}$.
3) Repeat step 2) a total of $|\mathcal{V}| - 1$ times.
4) If $V(i; c) + c_{ij} < V(j; c)$ for any $(i, j) \in \mathcal{E}$, the graph contains a negative-cost cycle and the shortest path length is $-\infty$ (corresponding to an unbounded LP). Otherwise $V(t; c)$ gives the length of the shortest path from s to t.

Optimal learning becomes an issue when the lengths c_{ij} are unknown. Length is often interpreted as travel time. For example, a GPS navigation system finds the quickest way to get from a specified origin to a specified destination. However, in reality, travel times are highly variable. GPS relies on certain estimates of travel time, but they may be outdated or inaccurate when a particular query is received. However, in the limited time available to make a decision, we may have an opportunity to collect a very small amount of information to help us choose a travel route. Some drivers located in different regions of the traffic network may be signed up to participate in a program where their smartphone can send information on traffic congestion (using the ability of the phone to tell how fast it is moving) to the GPS. We can choose to query a small number of smartphones, receive real-time observations of traffic congestion, and use them to update our estimates and improve our implementation decision (the route we choose).

Other network representations The graph does not have to represent a physical structure. Graphs are used in project management to represent precedence relations between different tasks or assignments. Consider the problem of managing a software project that consists of a series of tasks. Some tasks must be completed in a specific order, whereas others can be tackled in parallel. Figure 13.4 displays precedence relationships for the following set of tasks:

Each node in the graph represents a task, whereas an edge between tasks i and j indicates that i must be completed before j. In addition, each task has a duration d_i.

A	Build preliminary demo	F	Compatibility testing/QA
B	Alpha testing, user feedback	G	Train support staff
C	Preliminary market study	H	Final marketing campaign
D	Finalize design	I	Begin production
E	Plan production/distribution	J	Ship to retailers and launch

Given T tasks, we can schedule the tasks in the following way. Let x_i be the starting time of task i and solve

$$\min_x x_T$$

subject to

$$\begin{aligned} t_j &\geq t_i + d_i \qquad \text{for all } (i,j) \in \mathcal{E} \\ t_i &\geq 0 \qquad \text{for } i = 1, ..., T. \end{aligned}$$

We minimize the starting time of the last job. Another problem that often arises in project scheduling is known as the critical path problem. This is essentially the same as finding the longest path (the path with the largest total duration) from the first to the last job. The critical path is of interest to managers because it shows which jobs can be delayed without impacting key deadlines.

The duration of a particular task is rarely known exactly. Preliminary estimates are often inaccurate, leading to unplanned delays. Before beginning the project, we may wish to improve our estimates by going into our archives and analyzing data from earlier projects that involved similar tasks. The data may be cumbersome to access (many such archives have yet to be digitized), so our time is limited. It then becomes important to use our learning budget effectively by singling out the tasks that seem most important, and focusing on creating good estimates of their duration.

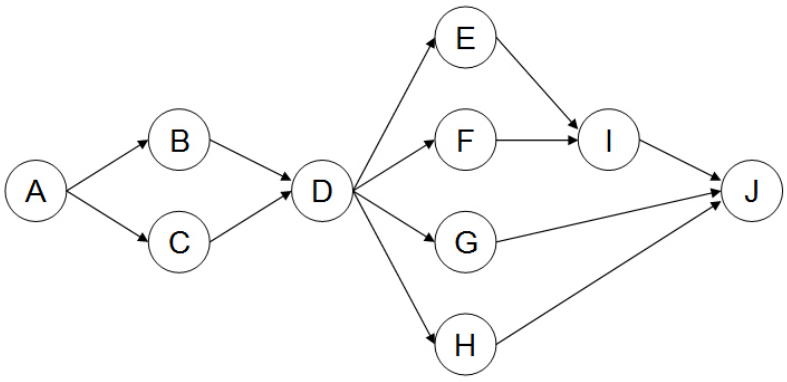


Figure 13.4 Graph of precedence relations for tasks in software development.

13.1.4 Discussion

We have just seen several problems where the decision problem (we sometimes call this the implementation problem) is a linear or nonlinear program. Finding the expected value of an experimental outcome can be computationally difficult. But we have seen that we can apply our optimal learning ideas within a learning algorithm to help us find a good learning policy.

Both of these problems could be formulated in the general framework of finding an experimental decision x to minimize $\mathbb{E}F(x, W)$. This is a very classical optimization problem with a rich algorithmic history. There are many algorithms which depend on the structure of the problem. An excellent reference for this problem class is Spall (2003) and the references cited there (the literature is extremely large).

13.2 LEARNING ON GRAPHS

We show how optimal learning can be incorporated into network optimization using, the shortest-path problem from Section 13.1.3 as an example. To make our analysis cleaner, we assume that the graph has no cycles, as in Figure 13.3.

Optimal learning begins to play a role when we no longer know the cost vector c. Let us apply our standard modeling assumptions from Chapter 4. Suppose that, for each edge $(i, j) \in \mathcal{E}$, we have $c_{ij} \sim \mathcal{N}\left(\theta^0_{ij}, \beta^0_{ij}\right)$, where β^0 denotes precision. We put a Gaussian prior on the length of each edge, and assume that the edges are independent. This is a strong assumption: for example, if the graph represents a traffic network, as in some of the examples discussed in Section 13.1.3, one would expect the travel times on neighbouring streets to be heavily correlated. We will allow correlations in Section 13.4. For now, we focus on the insights that can be obtained if we make the independence assumption.

As in ranking and selection, the belief state in this problem can be written as $S = (\theta, \beta)$, a vector of means and variances. A set of beliefs about the edges induces a belief about the best path. Define $V^n = V\left(s; \theta^n\right)$ to be the estimated length of the shortest path, given our time-n beliefs about the edge lengths. In other words, we simply run the Bellman-Ford algorithm using the estimates θ^n as the costs, and use the quantity V^n as our time-n estimate of the shortest path. Bellman-Ford can also provide us with the path that achieves this estimated length. We refer to this path as p^n.

Now suppose that, before we commit to a path through the network, we have N opportunities to collect information about individual edges. If we choose to learn about (i^n, j^n) at time n, we collect an observation $W^{n+1} \sim \mathcal{N}\left(c_{ij}, \beta^W_{ij}\right)$ and our beliefs evolve according to the familiar updating equations

$$\theta^{n+1}_{ij} = \begin{cases} \frac{\beta^n_{ij}\theta^n_{ij} + \beta^W_{ij} W^{n+1}_{ij}}{\beta^n_{ij} + \beta^W_{ij}} & \text{if } (i,j) = (i^n, j^n) \\ \theta^n_{ij} & \text{otherwise,} \end{cases} \tag{13.12}$$

$$\beta^{n+1}_x = \begin{cases} \beta^n_{ij} + \beta^W_{ij} & \text{if } (i,j) = (i^n, j^n) \\ \beta^n_{ij} & \text{otherwise.} \end{cases} \tag{13.13}$$

Just as in ranking and selection, we change our beliefs about one edge at a time. However, a small change in our beliefs about a single *edge* can have a profound effect on our beliefs about the shortest *path*. Figure 13.5(a) shows the graph from Figure 13.3, with the addition of prior beliefs about the edge lengths, represented by bell curves. The solid line in Figure 13.5(a) represents the path from node 1 to node 9 that we currently believe is the shortest. When we measure the edge $(6, 8)$, the outcome of our observation is smaller than expected, leading us to shift that particular bell curve to the left. Because this edge now seems shorter than before, we prefer to route our path through it, leading to the new solution in Figure 13.5(b). We only changed our beliefs about one edge, but the shortest path now looks very different from before.

Our objective is to choose a policy π for allocating the N experiments in a way that minimizes

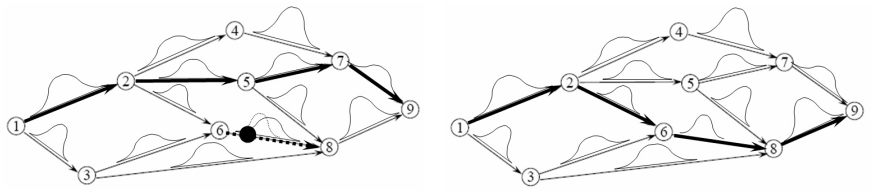
$$\min_{\pi} \mathbb{E} F^{\pi}(c, W) \tag{13.14}$$

where $F^{\pi}(c, W) = V^N$, the final time-N estimate of the length of the shortest path. Equation (13.14) reflects the key difference between learning on a graph and ranking and selection: we measure individual edges ("alternatives") just as before, but now we are no longer interested in finding the single alternative with the best value. We seek to measure alternatives that help us improve our solution to the shortest-path problem.

To see how this can be done, let us examine the impact of a single observation on our solution, that is, the difference between V^n and V^{n+1} brought about by measuring (i, j). For a fixed edge (i, j) define two paths as follows. The path p_{ij}^n is defined to be the shortest path *containing* (i, j), based on the time-n beliefs, whereas $\bar{p}_{ij}^n$ is the shortest path *not containing* (i, j), again according to the time-n beliefs. Both paths are easy to find. For p_{ij}^n, we can run Bellman-Ford to find the shortest path from s to i, and again for the shortest path from i to t. We then use (i, j) to connect these two paths. To find $\bar{p}_{ij}^n$, we merely run Bellman-Ford once on a modified graph with the edge (i, j) removed. Figure 13.6 gives a visual illustration.

Keep in mind that these two paths also depend on our beliefs at time n. When we run Bellman-Ford to find p_{ij}^n and $\bar{p}_{ij}^n$, we use θ^n for the edge costs. The algorithm will also provide us with V_{ij}^n and $\bar{V}_{ij}^n$, the estimated lengths of both paths. A length of a path is simply the sum of the estimates θ_{ij}^n on every edge (i, j) in the path.

The key insight of this section is the following. If we measure (i, j) at time-n, the path p^{n+1} that will be optimal under the time-$(n + 1)$ beliefs – that is, the path that we



(a) (b)

Figure 13.5 The effect of an observation of a single edge (dotted line) on our beliefs about the shortest path (solid line).

will get by running Bellman-Ford with the costs c^{n+1} at time $n+1$ – will necessarily be either p_{ij}^n or $\bar{p}_{ij}^n$. To understand this result, it helps to think in the following way. Suppose that (i,j) is already part of the current optimal path p^n. Then, if we measure it, our beliefs about it will either improve (in which case p^n will become even better than before) or get worse to the point where it is no longer advisable to route a path through (i,j). In the latter case, the best choice among the remaining paths is $\bar{p}_{ij}^n$. Similarly, if (i,j) is not part of p^n, our beliefs about it will either get worse (in which case we will keep the current optimal solution), or get better to the point where we will choose to route our path through (i,j). In the latter case, p_{ij}^n will be the best such route.

This argument allows us to calculate

$$\nu_{ij}^{KG,n} = \mathbb{E}\left[V^n - V^{n+1} \mid S^n, (i^n, j^n) = (i,j)\right], \tag{13.15}$$

the expected improvement contributed to our estimated length of the shortest path by a single observation of (i,j). We reprise the knowledge gradient notation of Chapter 4 because $\nu_{ij}^{KG,n}$ is the exact analog of (4.3), the marginal value of a single observation. We merely replace $\max_{x'} \theta_{x'}^{n+1} - \max_{x'} \theta_{x'}^n$, the improvement in the estimated value of the best alternative, with $V^n - V^{n+1}$, the improvement in the estimated value of our current implementation decision, which is the solution to a shortest-path problem. Our experimental decision can then be written as

$$(i^n, j^n) = \arg\max_{(i,j)} \nu_{ij}^{KG,n},$$

the edge with the highest value of information.

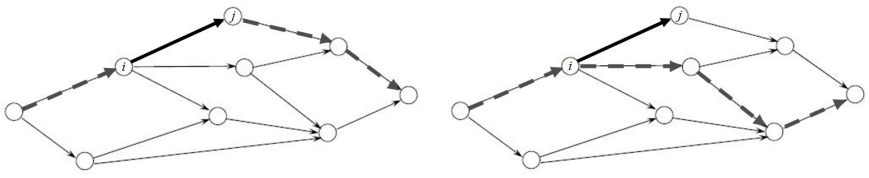
In fact, (13.15) has a closed-form solution

$$\nu_{ij}^{KG,n} = \tilde{\sigma}_{ij}^n f\left(-\frac{\left|V_{ij}^n - \bar{V}_{ij}^n\right|}{\tilde{\sigma}_{ij}^n}\right), \tag{13.16}$$

which is remarkably similar to (4.13), the KG formula for ranking and selection. As before, the quantity

$$\tilde{\sigma}_{ij}^n = \sqrt{\frac{1}{\beta_{ij}^n} - \frac{1}{\beta_{ij}^n + \beta_{ij}^W}}$$

represents the reduction in our uncertainty about (i,j) brought about by a single observation of this edge. The function $f(z) = z\Phi(z) + \phi(z)$ is exactly the same as



(a) (b)

Figure 13.6 Illustration of (a) the best path containing (i,j) and (b) the best path not containing (i,j).

before. The main difference is in the computation of the normalized influence. Where (4.11) uses the quantity $|\theta_x^n - \max_{x' \neq x} \theta_{x'}^n|$, (13.16) uses $\left|V_{ij}^n - \bar{V}_{ij}^n\right|$. This can be thought of as a generalization of (4.11). In ranking and selection, the implementation decision was the same as the experimental decision. We measured individual alternatives in order to find a good alternative, and the influence was determined by the distance between the alternative we chose and the best of the others. In this setting, the influence depends on the distance between the best implementation decision containing the chosen edge (alternative), and the best implementation decision not containing that edge.

13.3 ALTERNATIVE EDGE SELECTION POLICIES

The value of the KG concept, aside from giving us an algorithm for making decisions, is that it allows us to think about new problems. It is not immediately clear how to carry over ideas like Gittins indices, interval estimation, or UCB methods to a problem where the experimental decision is distinct from the implementation decision. On the other hand, the concept of the marginal value of information can be easily extended to such problems. In (13.15), we defined KG as the expected improvement made in our estimate of the value of the optimal implementation decision. We will see in Section 13.4 that this same idea can be used together with more general optimization models.

Still, there are always alternatives. For example, it is possible to shoehorn the graph problem into the framework of ranking and selection, using the concepts we developed in Chapter **??** for subset selection. A shortest-path problem can be viewed as a type of subset selection, because a path is just a subset of the edge set $\mathcal{E}$. We can convert the problem of learning on a graph into a ranking and selection problem where each alternative is a path. The only issue is that a graph is typically described by a set of nodes and edges. The set of all possible paths is typically very large and difficult to enumerate. Fortunately, we can use Monte Carlo simulation to create a small choice set of paths, as in Section 9.2.4.

Given the current belief state (θ^n, β^n), fix an integer K. For every $(i,j) \in \mathcal{E}$, generate samples $\bar{c}_{ij}^n(\omega_k)$, $k = 1, ..., K$ from the prior distribution $\mathcal{N}\left(\theta_{ij}^n, \beta_{ij}^n\right)$. Now, for every $k = 1, ..., K$, run the Bellman-Ford algorithm using the sampled costs $\bar{c}^n(\omega_k)$. The resulting path $\bar{p}_k^n$ is the optimal solution for the kth sample.

Our beliefs about the edges now induce a set of beliefs $\left(\theta^{paths,n}, \Sigma^{paths,n}\right)$ on the K paths we generated, according to the equations

$$\begin{aligned}
\theta_k^{paths,n} &= \sum_{(i,j) \in \bar{p}_k^n} \theta_{ij}^n, \qquad k = 1, ..., K, \\
\Sigma_{k,l}^{paths,n} &= \sum_{(i,j) \in \bar{p}_k^n \cap \bar{p}_l^n} \frac{1}{\beta_{ij}^n}, \qquad k, k' = 1, ..., K.
\end{aligned}$$

We can now apply any standard ranking and selection algorithm (including KG with correlated beliefs!) to this prior. The result will be some path $\bar{p}^*$. We can then use a simple heuristic to pick a single edge to measure from this path. One sensible choice

is

$$(i^n, j^n) = \arg \min_{(i,j)\in\bar{p}^*} \beta_{ij}^n,$$

the edge with the lowest precision on the path of interest. This approach allows us to avoid having to re-derive a KG formula every time we work on a new problem class. Instead, we can simply use the old framework of ranking and selection a little more flexibly.

13.4 LEARNING COSTS FOR LINEAR PROGRAMS*

We now extend the learning problem described in Section 13.2 to the broader setting of a general LP. This section is intended for readers with an interest in more advanced topics. Some familiarity with LP geometry and duality theory will be very helpful for understanding this material; Chvátal (1983) or Vanderbei (2008) are useful LP references.

We return to the optimization problem

$$V(c) = \max_x \; c^T x \tag{13.17}$$

subject to:

$$\begin{aligned} Ax &= b, \\ x &\geq 0, \end{aligned}$$

with which we started this chapter. Recall that, by the strong duality property of linear programming,

$$V(c) = \min_{y,s} \; b^T y \tag{13.18}$$

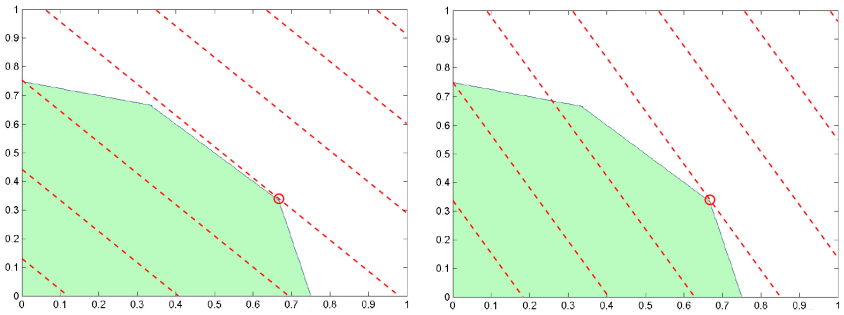
subject to

$$\begin{aligned} A^T y - s &= c, \\ s &\geq 0. \end{aligned}$$

That is, the dual LP has the same optimal value. We can let $x(c)$, $y(c)$ and $s(c)$ represent the optimal primal and dual solutions, that is, the choices of x, y, s in (13.17) and (13.18) that achieve the value $V(c)$.

Now, we assume that c is unknown (the other parameters A and b will still be known), and begin with a multivariate Gaussian prior $c \sim \mathcal{N}\left(\theta^0, \Sigma^0\right)$. As in Section 13.2, we have a budget of N experiments of individual objective coefficients which we can use to improve our solution. Measuring the jth coefficient will give us an observation from the distribution $\mathcal{N}(c_j, \lambda_j)$ where λ_j denotes a known variance. Measuring j^n at time n will cause our beliefs to change according to the equations

$$\begin{aligned} \theta^{n+1} &= \theta^n + \frac{W_{j^n}^{n+1} - \theta_{j^n}^n}{\lambda_{j^n} + \Sigma_{j^n j^n}^n} \Sigma^n e_{j^n}, \\ \Sigma^{n+1} &= \Sigma^n - \frac{\Sigma^n e_{j^n} e_{j^n}^T \Sigma^n}{\lambda_{j^n} + \Sigma_{j^n j^n}^n}, \end{aligned}$$



(a) (b)

Figure 13.7 The optimal solution $V\left(\theta^n + \tilde{\sigma}\left(j\right)z\right)$ remains unchanged over a range of values of z.

exactly as in (2.18) and (2.19). We can see that the graph problem in Section 13.2 is just a special case of this more general problem. The graph structure can be encoded in the matrix A, and we no longer require independent beliefs. Our belief state $S^n = (\theta^n, \Sigma^n)$ consists of the parameters of the multivariate Gaussian distribution.

At time n, our beliefs about the optimal value of the LP are given by $V\left(\theta^n\right)$. We simply solve the deterministic LP (using the simplex method or any standard software package such as Excel) using the vector θ^n for the objective coefficients. Our objective is to choose a learning policy π to maximize

$$\max_{\pi} \mathbb{E} F^{\pi}\left(c, W\right)$$

with $F^{\pi}\left(c, W\right) = V\left(c^N\right)$. The KG factor in this problem can be defined as

$$\nu_j^{KG,n} = \mathbb{E}\left[V\left(\theta^{n+1}\right) - V\left(\theta^n\right) \mid S^n, j^n = j\right]. \tag{13.19}$$

Next, recall from (4.22) that the conditional distribution of θ^{n+1}, given that we measure j at time n, can be written as

$$\theta^{n+1} = \theta^n + \tilde{\sigma}\left(j\right) Z^{n+1},$$

where $Z^{n+1} \sim \mathcal{N}\left(0,1\right)$ and $\tilde{\sigma}\left(j\right) = \frac{\Sigma^n e_j}{\sqrt{\lambda_j + \Sigma^n_{jj}}}$ is a vector of variance reductions. Thus, the challenge of computing the expectation in (13.19) reduces to solving

$$\mathbb{E}\left[V\left(\theta^{n+1}\right) \mid S^n, j^n = j\right] = \mathbb{E} V\left(\theta^n + \tilde{\sigma}\left(j\right) Z^{n+1}\right), \tag{13.20}$$

which is the expected value of a linear program with a stochastic objective function. In general, this problem is computationally intractable. However, in this specific case, the randomness in the objective function depends on a single scalar random variable. The expression $V\left(\theta^n + \tilde{\sigma}\left(j\right) Z^{n+1}\right)$ can be interpreted as the optimal value of an LP whose objective function θ^n is perturbed by a single stochastic parameter Z^{n+1}. If Z^{n+1} were to be replaced by a fixed quantity z, this would be a standard problem in LP sensitivity analysis. In our case, we have to analyze the sensitivity of the LP over the entire distribution of Z^{n+1}.

Let us recall two facts from linear programming. First, the optimal solution always occurs at a corner point (or "extreme point") of the feasible region. Figure 13.7 shows an example in two dimensions where the shaded region represents the set of feasible solutions x. For some fixed value z, the dashed lines in Figure 13.7(a) represent the level curves of the objective function, or points where $\left(\theta^n + \tilde{\sigma}\left(j\right)z\right)^T x = a$ for different values of a. The value of a that makes the level curve tangent to the feasible region is precisely $V\left(\theta^n + \tilde{\sigma}\left(j\right)z\right)$, and the corner point where the level curve touches the feasible region is the optimal solution $x\left(c^n + \tilde{\sigma}\left(j\right)z\right)$.

The second fact is that, for small enough changes in the sensitivity parameter z, the optimal solution $x\left(c^n + \tilde{\sigma}\left(j\right)z\right)$ does not change. In Figure 13.7(b), we change z slightly, rotating all of the level curves. However, the optimal level curve is tangent to the feasible region at the same point as before. The range of values of z for which the optimal solution does not change is sometimes called the *invariant support set*. We can rewrite $\mathbb{E}V\left(\theta^n + \tilde{\sigma}\left(j\right)Z^{n+1}\right)$ as a sum of integrals over the different possible invariant support sets. Suppose that $z_1, ..., z_I$ are a finite set of points, arranged in increasing order, such that $x\left(\theta^n + \tilde{\sigma}\left(j\right)z\right)$ is constant for all $z_i < z < z_{i+1}$. Then, for every i, there is a single corner point x_i satisfying $x_i = x\left(\theta^n + \tilde{\sigma}\left(j\right)Z^{n+1}\right)$ for $z_i < Z^{n+1} < z_{i+1}$. Consequently, taking the expectation over the distribution of Z^{n+1}, we obtain

$$\begin{aligned}\mathbb{E}V\left(\theta^n + \tilde{\sigma}\left(j\right)Z^{n+1}\right) &= \sum_i \int_{z_i}^{z_{i+1}} \left(\theta^n + \tilde{\sigma}\left(j\right)z\right)^T x_i \phi\left(z\right) dz \\ &= \sum_i a_i\left(\Phi\left(z_{i+1}\right) - \Phi\left(z_i\right)\right) + b_i\left(\phi\left(z_i\right) - \phi\left(z_{i+1}\right)\right),\end{aligned}$$

where $a_i = \left(\theta^n\right)^T x_i$ and $b_i = \tilde{\sigma}\left(j\right)^T x_i$. It turns out that this expression is equivalent to

$$\mathbb{E}V\left(\theta^n + \tilde{\sigma}\left(j\right)Z^{n+1}\right) = \left(\max_i a_i\right) + \sum_i \left(b_{i+1} - b_i\right) f\left(-\left|z_i\right|\right),$$

where f is once again the familiar function $f\left(z\right) = z\Phi\left(z\right) + \phi\left(z\right)$. We now make the observation that

$$\max_i a_i = \max_i \left(\theta^n\right)^T x_i = V\left(\theta^n\right),$$

because $\max_i a_i$ is the largest time-n objective value at all of the extreme points that could ever be optimal for any perturbation z, including $z = 0$. Therefore, $x\left(\theta^n\right)$ is one of the points x_i and its value is precisely $V\left(\theta^n\right)$. Therefore,

$$\mathbb{E}V\left(\theta^n + \tilde{\sigma}\left(j\right)Z^{n+1}\right) = V\left(\theta^n\right) + \sum_i \left(b_{i+1} - b_i\right) f\left(-\left|z_i\right|\right). \tag{13.21}$$

Combining (13.19), (13.20) and (13.21) gives us

$$\nu_j^{KG,n} = \sum_i \left(b_{i+1} - b_i\right) f\left(-\left|z_i\right|\right), \tag{13.22}$$

which looks identical to the computation of KG in ranking and selection with correlated beliefs (Section 4.5), with the important difference that the breakpoints z_i now represent the values of Z^{n+1} that change our implementation decision, which in this case is a corner point of the LP.

In the remainder of this section, we explain how the breakpoints z_i and the corresponding corner points x_i can be computed. After this is done, computing the KG factor is a simple matter of applying (13.22). We can find the breakpoints by starting with $x(\theta^n)$, which is the optimal solution when $z = 0$, finding the breakpoints that are nearest to $z = 0$, and expanding outward until all the breakpoints have been found.

Our first order of business is to determine whether $z = 0$ is itself a breakpoint. We can do this by calculating the smallest and largest values of z for which $x(\theta^n)$ is an optimal solution of the perturbed LP. These quantities, which we denote as z_- and z_+, are the optimal values of the LPs

$$z_- = \min_{y,s,z} \quad z \tag{13.23}$$

subject to

$$\begin{aligned} A^T y - s - z\tilde{\sigma}(j) &= \theta^n, \\ x(\theta^n)^T s &= 0, \\ s &\geq 0, \end{aligned}$$

and

$$z_+ = \max_{y,s,z} \quad z \tag{13.24}$$

subject to

$$\begin{aligned} A^T y - s - z\tilde{\sigma}(j) &= \theta^n, \\ x(\theta^n)^T s &= 0, \\ s &\geq 0. \end{aligned}$$

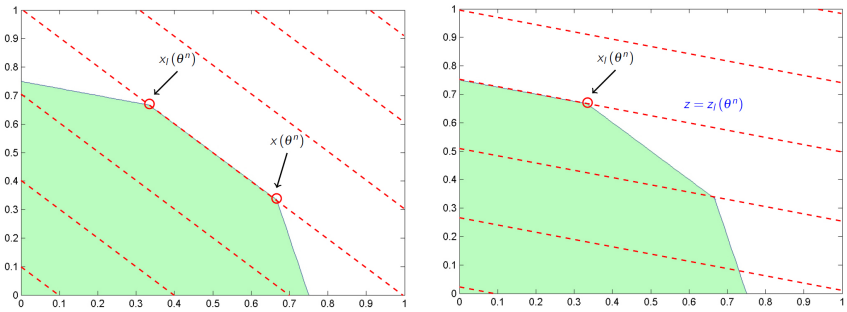
The feasible region, which is the same in both (13.23) and (13.24), merely ensures that $x(\theta^n)$ continues to be an optimal solution of the perturbed LP. Recall that there are three conditions for x to be an optimal solution to the primal LP (13.17) and y, s to be an optimal solution to the dual LP (13.18). First, x has to be primal-feasible. Second, y and s have to be dual-feasible. Third, we must have $x^T s = 0$, known as the complementary slackness property. In our case, $x(\theta^n)$ is automatically feasible for the perturbed LP, because it is optimal for the unperturbed LP

$$V(\theta^n) = \max_x \quad (\theta^n)^T x \tag{13.25}$$

subject to

$$\begin{aligned} Ax &= b, \\ x &\geq 0, \end{aligned}$$

which has the same feasible region as the perturbed LP. The constraints $A^T y - s - z\tilde{\sigma}(j) = \theta^n$ and $x(\theta^n)^T s = 0$ in (13.23) and (13.24) simply ensure that we are able to find a solution to the dual of the perturbed LP that maintains complementary slackness with $x(\theta^n)$. We are looking for the smallest and largest values of the perturbation z for which this is still possible. We now analyze four possible outcomes of this procedure.



(a) (b)

Figure 13.8 When $z_- = 0$, the optimal level curve is tangent to a face of the feasible region. The endpoints of this face are $x(\theta^n)$ and $x_l(\theta^n)$.

Case 1: $\boldsymbol{z_- < 0 < z_+}$ In this case, $z = 0$ is not a breakpoint. This is the case illustrated in Figure 13.7(a). The level curve of θ^n is tangent to a single corner point. However, both z_- and z_+ are breakpoints.

Case 2: $\boldsymbol{z_- = 0, z_+ > 0}$ When zero is a breakpoint, the optimal level curve is tangent to a face of the feasible region, rather than to a single corner point, and the solution $x(\theta^n)$ is located at one corner of the face. Figure 13.8(a) provides a visual illustration. If we apply a positive perturbation $0 < z < z_+$, the level curves will rotate clockwise, but $x(\theta^n)$ will still be the optimal solution. However, any negative perturbation will rotate the level curves counter-clockwise and change the optimal solution.

To find the next breakpoint $z_l(\theta^n) < 0$, we first need to locate the other corner $x_l(\theta^n)$ of the face. This solution will still be optimal for the unperturbed LP, but it will also be optimal for the perturbed LP with $z = z_l(\theta^n)$.

We can do this by solving two LPs. First, $x_l(\theta^n)$ is the optimal solution to the problem

$$V_l(\theta^n) = \min_x \tilde{\sigma}(j)^T x \tag{13.26}$$

subject to

$$\begin{aligned} Ax &= b, \\ s(\theta^n)^T x &= 0, \\ x &\geq 0. \end{aligned}$$

Any feasible solution to this problem continues to be optimal for the unperturbed LP in (13.25). In Figure 13.8(a), such a solution would be located somewhere on the face of the feasible region. Of all such points, the x with the lowest $\tilde{\sigma}(j)^T x$ corresponds to the opposite corner of the face. The optimal value of the LP

$$z_l(\theta^n) = \min_{y,s,z} z \tag{13.27}$$

subject to

$$\begin{aligned} A^T y - s - z\tilde{\sigma}(j) &= \theta^n, \\ x_l(\theta^n)^T s &= 0, \\ s &\geq 0, \end{aligned}$$

corresponds to the next breakpoint, the most negative perturbation for which the perturbed LP still has $x_l(\theta^n)$ as an optimal solution. The relationship between $x(\theta^n)$, $x_l(\theta^n)$, and $z_l(\theta^n)$ can be seen in Figure 13.8(b).

Case 3: $z_- < 0, z_+ = 0$ Zero is still a breakpoint, but $x(\theta^n)$ is now at the opposite corner of the face. We can now rotate the level curves counter-clockwise without changing the optimum, but not clockwise. The next corner point $x_u(\theta^n)$ is the optimal solution to the problem

$$V_u(\theta^n) = \max_x \quad \tilde{\sigma}(j)^T x \tag{13.28}$$

subject to

$$\begin{aligned} Ax &= b, \\ s(\theta^n)^T x &= 0, \\ x &\geq 0. \end{aligned}$$

The next breakpoint $z_u(\theta^n) > 0$ is the optimal value of the LP

$$z_u(\theta^n) = \quad \max_{y,s,z} \quad z$$

subject to

$$\begin{aligned} A^T y - s - z\tilde{\sigma}(j) &= \theta^n, \\ x_u(\theta^n)^T s &= 0, \\ s &\geq 0. \end{aligned}$$

Case 4: $z_- = z_+ = 0$ Zero is a breakpoint, so the optimal level curve for the vector θ^n is still tangent to a face of the feasible region. However, $x(\theta^n)$ is no longer a corner point. Rather, it is somewhere in the middle of the face. We can sometimes discover optimal solutions like this if we use an interior-point algorithm to solve (13.25) instead of the simplex method. Fortunately, this case is easy to deal with: we simply solve (13.26)-(13.29) all together to find both corner points of the face at the same time.

Putting it all together We can repeat this analysis to find sequences of positive and negative breakpoints. For example, if $z_- = 0$ and $z_+ = 0$, as in Case 2, we can find $z_l(\theta^n)$ and then repeat the analysis of Case 2 again, this time with $\theta^n + \tilde{\sigma}(j) z_l(\theta^n)$ as the "unperturbed" objective function, in place of θ^n. This will give us the next negative breakpoint after $z_l(\theta^n)$, and so on, until (13.27) becomes unbounded, which means that the same solution will continue to be optimal regardless of how much we perturb the objective function. The procedure to find a vector $\bar{z}$

of breakpoints and a vector $\bar{x}$ of corresponding corner points can be summarized as follows.

1) Solve (13.25) to obtain $x\left(\theta^n\right)$, $y\left(\theta^n\right)$ and $s\left(\theta^n\right)$.

2) Solve (13.23) and (13.24) to obtain z_-, z_+.

   2a) If $z_- < 0 < z_+$, let $\bar{z} = (z_-, z_+)$ and $\bar{x} = \left(x\left(\theta^n\right)\right)$.

   2b) Otherwise, let $\bar{z} = (0)$ and let $\bar{x}$ be an empty vector.

3) While $\min \bar{z} > -\infty$, do the following:

   3a) Let $\theta' = \theta^n + \tilde{\sigma}(j) \min \bar{z}$.

   3b) Find $x_l\left(\theta'\right)$ and $z_l\left(\theta'\right)$ using (13.26) and (13.27).

   3c) Update $\bar{z} \leftarrow \left(\min \bar{z} + z_l\left(\theta'\right), \bar{z}\right)$ and $\bar{x} \leftarrow \left(x_l\left(\theta'\right), \bar{x}\right)$.

4) While $\max \bar{z} < \infty$, do the following:

   4a) Let $\theta' = \theta^n + \tilde{\sigma}(j) \max \bar{z}$.

   4b) Find $x_u\left(\theta'\right)$ and $z_u\left(\theta'\right)$ using (13.28) and (13.29).

   4c) Update $\bar{z} \leftarrow \left(\bar{z}, \max \bar{z} + z_u\left(\theta'\right)\right)$ and $\bar{x} \leftarrow \left(\bar{x}, x_u\left(\theta'\right)\right)$.

13.5 BIBLIOGRAPHIC NOTES

Section 13.1 - For other important mathematical programming applications (with a focus on the network and LP models considered in this chapter), see e.g. Vanderbei (2008).

Sections 13.2 - 13.3 - These sections are based on Ryzhov & Powell (2011*b*). Prior to this paper, several authors in the computer science community addressed the "bandit shortest path" problem, which would be the online version of the shortest path problem that we consider. For example, Abernethy et al. (2008) describes an algorithm that requires enumerating all the paths, ignoring the structure that arises from overlapping paths. Takimoto & Warmuth (2003) describes the use of a hedging algorithm for the online shortest path problems. Stochastic shortest path problems have been studied even earlier (Kulkarni 1986, Snyder & Steele 1995, Peer & Sharma 2007), but usually with the assumption of a known distribution for the rewards.

Section 13.4 - This material is due to Ryzhov & Powell (2011*a*). The procedure for finding breakpoints comes from Hadigheh & Terlaky (2006).

CHAPTER 14

OPTIMIZING OVER CONTINUOUS ALTERNATIVES*

There are many applications where we have to choose the best value of a continuous, multidimensional vector x. One example arises in the electricity sector, where utilities have to deal with the high variability of electricity spot prices. One strategy is to use a battery to store energy when prices are low, and release energy when prices are high. The idea is illustrated in Figure 14.2, where we store energy when the price goes below x^{store}, and we release energy when the price goes above $x^{withdraw}$. We now face the problem of deciding how to choose $x = (x^{store}, x^{withdraw})$. Let $\mu(x)$ be the expected profits per day that we obtain from using a policy fixed by x, where we assume we can only obtain noisy estimates of $\mu(x)$. Our challenge, as with elsewhere in this volume, is finding the best value of x as quickly as possible.

While we do not know the true function $\mu(x)$, we imagine it might look like the surface shown in Figure 14.2. We are going to assume it is smooth, but not necessarily concave (or convex, if we were solving a minimization problem). If x had only one or two dimensions, we could discretize it and use the techniques presented in the earlier chapters, although even two dimensions starts to become problematic if, for example, we would like to discretize each dimension into 100 intervals. If we have three or more dimensions, discretization quickly becomes cumbersome.

There are a number of problems which require tuning continuous but relatively low-dimensional parameter vectors. Some examples include:

Optimal Learning. By Warren B. Powell and Ilya O. Ryzhov

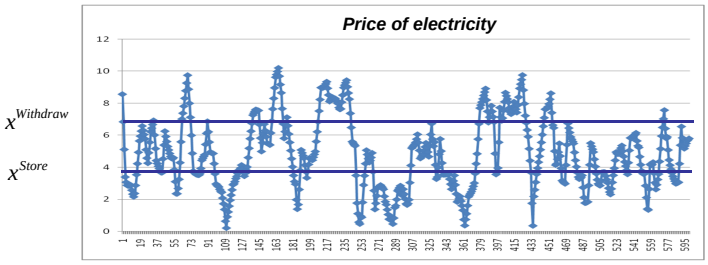


Figure 14.1 A battery charging problem, where we storage energy when the price goes below x^{store}, and withdraw it when the price goes above $x^{withdraw}$.

- Finding the best parameters that govern a business simulator - We might have a model that simulates the use of ambulances, or a manufacturing system that depends on the speed of different machines. A model of freight transportation may require assumptions on the time that drivers have to rest or the maximum time that a customer may be served early or late.
- Tuning the concentrations of different chemicals - We may be trying to maximize the yield of a chemical process that depends on the concentrations of different additives (we may also have to vary the temperature of different steps of the process).
- Design of devices - The design of an aerosol spray can requires tuning the diameter of the feed-in tube, the spacing between the feed-in tube and the aerosol spray tube, the diameter of the aerosol spray tube, and the pressure in the can.

In some cases, these parameters have to be tuned using field experiments or physical lab experiments, while in other cases they can be done using computer simulations. Computer simulations may run in a few minutes, but in some instances they can require hours or even days to complete a single run. Laboratory and field experiments can be even more time consuming, and are generally much more expensive. For this reason, we want to choose our experiments carefully so that we learn as much as possible from each function evaluation.

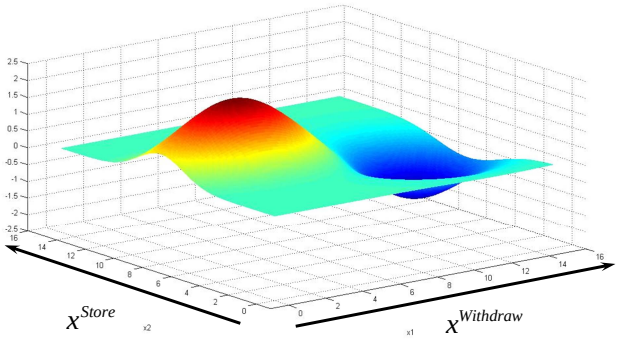


Figure 14.2 Illustration of the surface generated by varying x^{store} and $x^{withdraw}$.

We are going to continue to use our notation that $\mu(x)$ is the true value of the function at x. The only difference is that x is now a continuous vector. We are going to make a series of experiments $x^0, x^1, \ldots, x^n$, from which we make noisy observations of our function of the form

$$\hat{y}^{n+1} = \mu(x^n) + \varepsilon^{n+1}.$$

We assume that ε is a random variable that is normally distributed with mean 0 and variance λ. We wish to design a sequential search policy π that guides the selection of x^i, giving us an estimate of the value of our function which we represent as

$$F^\pi = \mu(x^N),$$

where $x^N = \arg\max_{x \in \mathcal{X}} \bar{\mu}^N(x)$, and $\bar{\mu}^N(x)$ is our estimate of $\mu(x)$ after N experiments obtained by following policy π. Our challenge is to find the policy π^* that solves

$$F^* = \max_\pi F^\pi.$$

This chapter addresses the problem of tuning these multidimensional parameter vectors for problems where we do not have access to derivatives. As always, we assume that observing $\mu(x)$ is time consuming and/or expensive, and noisy. Also, while we focus on multidimensional parameter vectors, our experimental work as of this writing is for vectors where the number of dimensions is less than 10.

14.1 THE BELIEF MODEL

In previous chapters, we have considered different ways of representing our belief in the function. We started with a lookup table, where we assumed that x was discrete, and there was an estimate θ_x^n for each x (see, for example, Chapters 3, 4 and 5). We have also considered models where we used linear regression to approximate $\mu(x)$ (as in Chapter 7). In this chapter, we need a different belief model since we have almost no idea about the structure of our function, but we do know it is continuous. For this reason, we are going to use an approach known as *Gaussian process regression* which is a class of nonparametric models which creates an approximation based on all prior observations.

We begin with a prior $\theta^0(x)$, which is now in the form of a continuous function. We are going to use our policy to generate a sequence of experiments $x^0, \ldots, x^{n-1}$, and we are going to use these points to approximate our function. Thus, after these n experiments $\hat{y}^1, \ldots, \hat{y}^n$, we will have a belief $\theta^n(x)$ at each of these points. We also need to capture the covariance in our beliefs about the function at each of these points. We define the $n \times n$ matrix Σ^n, where $\Sigma^n_{ij} = Cov^n(\mu(x^i), \mu(x^j))$ is the covariance in our belief about μ at x^i and x^j after n iterations. Note that each time we run a new experiment, θ^n grows by one element, while the matrix Σ^n adds a row and a column.

We are going to take advantage of the continuous structure of the problem and assume that we can write the covariance in our belief between any two points using

the function

$$Cov^0(\mu(x^i), \mu(x^j)) = \beta \exp(-\sum_{m=1}^{p} \alpha_i (x_m^i - x_m^j)^2), \quad \alpha > 0, \beta > 0. \tag{14.1}$$

It is common to refer to the p-dimensional vector α as the activity of μ, while the scalar parameter β captures the uncertainty of our belief about μ. The activity parameter α controls the degree of smoothness, where the function becomes smoother as α becomes smaller. For larger values of α, the covariance between two points shrinks with α, which means that we learn less about $\mu(x^i)$ and $\mu(x^j)$ as x^i and x^j become farther apart.

14.1.1 Updating equations

We have to learn how to update our beliefs about $\mu(x^i),\ i = 1, \ldots, n$ as we make more observations. We are going to use the property that the vector θ^n follows a multivariate normal distribution, where $\mathbb{E}^n \mu(x^i) = \theta^n(x^i)$ and the covariance matrix is given by Σ^n. Throughout this chapter $\mathbb{E}^n$ refers to the conditional information given the history of observations $\hat{y}^1, \ldots, \hat{y}^n$.

We start by calculating a vector $\tilde{y}^n$ which we call *experimental residuals* using

$$\tilde{y}^n = \begin{bmatrix} \hat{y}^1 \\ \vdots \\ \hat{y}^n \end{bmatrix} - \begin{bmatrix} \theta^0(x^0) \\ \vdots \\ \theta^0(x^{n-1}) \end{bmatrix}. \tag{14.2}$$

This is the difference between our observations $\hat{y}^i,\ i = 1, \ldots, n$, and our original belief $\theta^0(x^i),\ i = 1, \ldots, n$. We also define the *residual covariance* S^n which is updated using

$$S^n = \Sigma^0 + Diagonal([\lambda(x^0), ..., \lambda(x^{n-1})]). \tag{14.3}$$

Expanded into matrices, this is the same as

$$\begin{bmatrix} S_{11}^n & \cdots & S_{1i}^n & \cdots & S_{1n}^n \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ S_{i1}^n & \cdots & S_{ii}^n & \cdots & S_{in}^n \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ S_{n1}^n & \cdots & S_{ni}^n & \cdots & S_{nn}^n \end{bmatrix} = \begin{bmatrix} \Sigma_{11}^0 & \cdots & \Sigma_{1i}^0 & \cdots & \Sigma_{1n}^0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \Sigma_{i1}^0 & \cdots & \Sigma_{ii}^0 & \cdots & \Sigma_{in}^0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \Sigma_{n1}^0 & \cdots & \Sigma_{ni}^0 & \cdots & \Sigma_{nn}^0 \end{bmatrix} + \begin{bmatrix} \lambda(x^0) & 0 & \cdots & \cdots & 0 \\ 0 & \ddots & 0 & \cdots & \vdots \\ \vdots & 0 & \lambda(x^i) & 0 & \vdots \\ \vdots & \vdots & 0 & \ddots & 0 \\ 0 & \cdots & \cdots & 0 & \lambda(x^{n-1}) \end{bmatrix}$$

Finally, we are going to compute the *gain matrix*, denoted G^n using

$$G^n = \Sigma^0 [S^n]^{-1}. \tag{14.4}$$

As we see shortly below, the gain matrix is used to weigh new information, which plays the same role as a stepsize.

The updating equations for the mean vector θ^n and the covariance matrix S^n are now given by

$$\begin{bmatrix} \theta^n(x^0) \\ \vdots \\ \theta^n(x^{n-1}) \end{bmatrix} = \begin{bmatrix} \theta^0(x^0) \\ \vdots \\ \theta^0(x^{n-1}) \end{bmatrix} + G^n\tilde{y}^n, \tag{14.5}$$

$$\Sigma^n = (I_n - G^n)\Sigma^0. \tag{14.6}$$

where I_n is the $n \times n$ identity matrix.

Equations (14.5) and (14.6) update our beliefs around the points $x^0, \ldots, x^n$ that we have already measured, but we also need a belief for an arbitrary point x, since we need to search over all possible points to determine the next point that we *might* wish to evaluate. We begin by defining

$$\bar{\Sigma}^0 = \Sigma^0([x^0, ..., x^{n-1}, x]),$$
$$\bar{\Sigma}^n = \Sigma^n([x^0, ..., x^{n-1}, x]).$$

Let $\vec{0}$ be a column vector of zeroes. Our new gain matrix is given by

$$\bar{K}^n = \bar{\Sigma}^0 \begin{bmatrix} I_n \\ - \\ \vec{0}^T \end{bmatrix} [S^n]^{-1}. \tag{14.7}$$

We can now find θ^0 and $\bar{\Sigma}^0$ for an arbitrary $(n+1)$st point x using

$$\begin{bmatrix} \theta^n(x^0) \\ \vdots \\ \theta^n(x^{n-1}) \\ \theta^n(x) \end{bmatrix} = \begin{bmatrix} \theta^0(x^0) \\ \vdots \\ \theta^0(x^{n-1}) \\ \theta^0(x) \end{bmatrix} + \bar{K}^n\tilde{y}^n, \tag{14.8}$$

$$\bar{\Sigma}^n = (I_{n+1} - \bar{K}^n \begin{bmatrix} I_n & | & \vec{0} \end{bmatrix})\bar{\Sigma}^0. \tag{14.9}$$

If we want the distribution of $\mu(x)$ given our n observations at some arbitrary decision x, we can use (14.8) and (14.9) to obtain

$$\theta^n(x) = \theta^0(x) + \begin{bmatrix} \Sigma^0(x^0, x) & , \cdots , & \Sigma^0(x^{n-1}, x) \end{bmatrix} [S^n]^{-1}\tilde{y}^n, \tag{14.10}$$

$$\Sigma^n(x, x) = \Sigma^0(x, x) - \begin{bmatrix} \Sigma^0(x^0, x) & , \cdots , & \Sigma^0(x^{n-1}, x) \end{bmatrix} [S^n]^{-1} \begin{bmatrix} \Sigma^0(x^0, x) \\ \vdots \\ \Sigma^0(x^{n-1}, x) \end{bmatrix}. \tag{14.11}$$

Equation (14.10) is known as *Gaussian process regression* (GPR) in some communities, and regression kriging in others.

We have to choose the point x^n to measure before we have observed the outcome $\hat{y}^{n+1}$. The updated regression function, given the observations $\hat{y}^1, \ldots, \hat{y}^n$, is nor-

mally distributed with distribution

$$\begin{bmatrix} \theta^{n+1}(x^0) \\ \vdots \\ \theta^{n+1}(x^{n-1}) \\ \theta^{n+1}(x^n) \end{bmatrix} = \begin{bmatrix} \theta^{n}(x^0) \\ \vdots \\ \theta^{n}(x^{n-1}) \\ \theta^{n}(x^n) \end{bmatrix} + \tilde{\sigma}(\bar{\Sigma}^n, x^n)Z^{n+1}, \tag{14.12}$$

where $Z^{n+1} = \left(\hat{y}^{n+1} - \theta^n(x^n)\right) / \sqrt{\lambda(x^n) + \Sigma^n(x^n, x^n)}$, with

$$\tilde{\sigma}(\Sigma, x) = \frac{\Sigma e_x}{\sqrt{\lambda(x) + e_x^T \Sigma e_x}}. \tag{14.13}$$

Here e_x is a column vector of zeroes with a 1 at the row corresponding to decision x. It can be shown that Z^{n+1} is a standard normal random variate (mean 0, variance 1) because $\text{Var}(\hat{y}^{n+1} - \theta^n(x^n)|\mathcal{F}^n) = \lambda(x^n) + \Sigma^n(x^n, x^n)$.

14.1.2 Parameter estimation

Our model is characterized by the p-dimensional vector α, the scalar β and the noise λ, as well as our prior $\theta^0(x)$. There may be problems where we feel we can assume these are known, but in practice we are going to have to estimate them from data. As is so often the case with statistical estimation, there is more than one way to solve this problem, but a popular method that we have found works quite well is maximum likelihood estimation (MLE).

MLE starts by creating a *likelihood function* which is the product of the density (using the normal distribution) of all the prior observations given the unknown parameters. We take the log of this product (giving us the log-likelihood), and then find the values of the parameters to maximize the resulting sum.

We form the likelihood function using

$$\begin{aligned} &L_{\hat{y}}\left(\alpha, \beta, \lambda(x^0), ..., \lambda(x^{n-1}), \theta^0(x^0), ..., \theta^0(x^{n-1})\right) \\ &\quad = \quad (2\pi)^{-n/2} \left|S^n\right|^{-1/2} \exp\left(-\frac{1}{2}\begin{bmatrix} \hat{y}^1 - \theta^0(x^0) \\ \vdots \\ \hat{y}^n - \theta^0(x^{n-1}) \end{bmatrix}^T (S^n)^{-1} \begin{bmatrix} \hat{y}^1 - \theta^0(x^0) \\ \vdots \\ \hat{y}^n - \theta^0(x^{n-1}) \end{bmatrix}\right). \end{aligned}$$

Now, if we assume that the variance of the observation noise, $\lambda(\cdot)$, is a constant λ and $\theta^0(\cdot)$ is a constant θ^0, we can write the likelihood function as

$$L_{\hat{y}}(\alpha, \beta, \lambda, \theta^0) = (2\pi)^{-n/2}|\Sigma^0 + \lambda I_n|^{-1/2} \exp\left(-\frac{1}{2}(\hat{y} - \theta^0\mathbf{1})^T(\Sigma^0 + \lambda I_n)^{-1}(\hat{y} - \theta^0\mathbf{1})\right),$$

where $\mathbf{1}$ is a $n \times 1$ column vector of ones and $\hat{y} = \begin{bmatrix} \hat{y}^1 & \cdots & \hat{y}^n \end{bmatrix}^T$. Note that in this case we are estimating $p + 3$ parameters using n observations. We can write the log-likelihood function as:

$$\ell_{\hat{y}}(\alpha, \beta, \lambda, \theta^0) = -\frac{n}{2}\ln(2\pi) - \frac{1}{2}\ln(|\Sigma^0 + \lambda I_n|) - \frac{1}{2}(\hat{y} - \theta^0\mathbf{1})^T(\Sigma^0 + \lambda I_n)^{-1}(\hat{y} - \theta^0\mathbf{1}). \tag{14.14}$$

We can approximately maximize the likelihood over the parameters by using the function `patternsearch` in MATLAB started at multiple points chosen by a Latin hypercube sampling (LHS) design using the command `lhsdesign`. Also, in the above log-likelihood we can easily solve for θ^0 in terms of α, β, and λ, giving us the estimate

$$\hat{\theta^0} = \frac{\hat{y}^T(\Sigma^0 + \lambda I_n)^{-1}\mathbf{1}}{\mathbf{1}^T(\Sigma^0 + \lambda I_n)^{-1}\mathbf{1}}.$$

Finally, to prevent numerical issues, if $|\Sigma^0 + \lambda I_n|$ is very small in (14.14), a useful equivalent expression to $\ln(|\Sigma^0 + \lambda I_n|)$ is $trace(logm(\Sigma^0 + \lambda I_n))$ where $logm$ is the matrix logarithm.

We typically will have to perform an initial sample of experiments x to obtain a starting estimate of α, β and λ. If d is the dimensionality of our parameter vector (the sum of the dimensions of α, β and λ), a common rule of thumb is to perform a Latin hypercube design with $2d$ plus 2 points. In MATLAB, we can use `lhsdesign` to choose the points $x^1, \ldots, x^{2d+2}$. However, as the number of dimensions grows, the value of a LHS design diminishes, and it is better to simply choose the starting experiments at random.

14.2 SEQUENTIAL KRIGING OPTIMIZATION

Sequential kriging uses a form of meta-modeling that assumes the surface is represented by a linear model, a bias model and a noise term, which we can write using

$$\mu(x) = \sum_{f \in \mathcal{F}} \theta_f \phi_f(x) + Z(x) + \varepsilon.$$

Here, $\mathcal{F}$ is a set of features, $\phi_f(x)$ is a basis function (also known as an independent variable) corresponding to feature f, and $Z(x)$ is a term that represents the systematic bias due to the limitations of the basis functions.

The kriging meta-model is based on the idea that the bias function $Z(x)$ is a realization of a stationary Gaussian stochastic process. If x is a d-dimensional vector, we assume that the covariance between $Z(x)$ and $Z(x')$ can be written

$$Cov(Z(x), Z(x')) = \beta \exp\left[-\sum_{i=1}^{d} \alpha_i (x_i - x_i')^2\right],$$

where, as before, β is the variance of the stochastic process while α_i scales the covariance function for each dimension.

Samples of one-dimensional Gaussian surfaces are illustrated in Figure 14.3, which shows curves generated from different values of the activity variable α. Smaller values of α produces slower, undulating surfaces, while larger values of α (which reduces the correlations between nearby points) produces surfaces that undulate with higher frequencies. If this is the bias, it means that the true surface is likely to be reasonably smooth. It is important to realize that $Z(x)$ is not a random noise

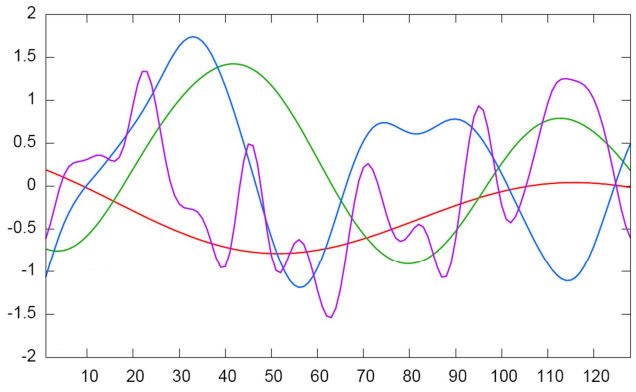


Figure 14.3 Illustration of sample realizations of one-dimensional Gaussian surfaces.

term with zero mean (which would imply that the approximation is unbiased). A sample realization of $Z(x)$ is the bias for a specific function $\mu(x)$ and a specific approximation represented by the set of basis functions ϕ and regression vector θ.

The best linear predictor $\bar{Y}^n(x)$ of $\mu(x)$ after n experiments is given by

$$\begin{aligned}\bar{Y}^n(x) &= \sum_{f\in\mathcal{F}} \theta_f^n \phi_f(x) + \\ &\quad \sum_{i=1}^{n} Cov(Z(x_i), Z(x)) \sum_{j=1}^{n} Cov(Z(x_i), Z(x_j))(\hat{y}_i - \sum_{f\in\mathcal{F}} \theta_f^n \phi_f(x)),\end{aligned}$$

where θ^n is the least squares estimator of the regression parameters after n observations.

The idea of sequential kriging is to use the expected improvement $I(x)$ resulting from measuring the function at x. We start by illustrating this for a function $f(x)$ that can be observed deterministically. Let x^n be the current best solution given our current approximation. After n experiments, let $F(x)$ represent our belief about $f(x)$ which is normally distributed with mean $\bar{Y}^n(x)$ and variance $\sigma^{2,n}(x)$. The expected improvement, given the history of n observations, can be written as

$$\mathbb{E}^n[I(x)] = \mathbb{E}^n \max\big(f(x^*) - F(x), 0\big). \tag{14.15}$$

We need to build on this idea to handle the issue of noisy experiments. Sequential kriging optimization (SKO) uses the following expression for the expected improvement

$$\mathbb{E}^n I(x) = \mathbb{E}^n\left[\max\left(\bar{Y}^n(x^{**}) - \mu(x), 0\right)\right]\left(1 - \frac{\sigma_\varepsilon}{\sqrt{\sigma^{2,n}(x) + \sigma_\varepsilon^2}}\right). \tag{14.16}$$

The first term on the right hand side of (14.16) mimics the expected improvement term used if we could measure the function deterministically in equation (14.15). The second term in (14.16) represents a heuristic adjustment term that rewards uncertainty.

If $\sigma^{2,n}(x) = 0$, then this adjustment term is zero and we would not place any value in measuring that point.

The expectation can be calculated analytically using

$$\begin{aligned}\mathbb{E}^n\left[\max\left(\bar{Y}^n(x^{**}) - \mu(x)\right)\right] &= \left(\bar{Y}^n(x^{**}) - \bar{Y}^n(x)\right)\Phi\left(\frac{\bar{Y}^n(x^{**}) - \bar{Y}^n(x)}{\sigma^n(x)}\right) \\ &\quad + \sigma^n(x)\phi\left(\frac{\bar{Y}^n(x^{**}) - \bar{Y}^n(x)}{\sigma^n(x)}\right),\end{aligned}$$

where ϕ and Φ are the standard normal density and cumulative distribution functions.

Define a utility function that captures the uncertainty associated with a point x after n observations. For example, we might choose

$$u^n(x) = -\left(\bar{Y}^n(x) + \sigma^n(x)\right).$$

We choose our effective best solution, x^{**}, by maximizing the utility. We could try to search over the entire region $x \in \mathcal{X}$, but a common shortcut is to limit the search over previously sampled locations $x^1, \ldots, x^n$, giving us the rule

$$x^{**} = \arg\max_{x^1,\ldots,x^n}[u^n(x)].$$

14.3 EFFICIENT GLOBAL OPTIMIZATION

While our interest is in solving problems where we can only obtain noisy observations of our function $\mu(x)$, an important contribution is a procedure known as *efficient global optimization* (EGO) which was developed for the case where there is no noise. EGO is similar to the knowledge gradient in that it uses the same idea of choosing to measure the point that maximizes the expected value of information. We have already seen this concept in Section **??**, where it was known as the expected improvement (EI) procedure. In the literature, the names EGO and EI are used interchangeably to refer to the same technique.

Following our standard notation, we assume that we choose an experiment x^n and then observe $\hat{y}^{n+1} = \mu(x^n)$. Our indexing system follows the style of the rest of the book, and is designed to handle uncertainty. Below, $\hat{y}^{n+1}$ will be a random variable when we choose x^n, but in this section it is deterministic.

We start by defining

$$I^{n+1}(x) = \max\left(\theta^{n+1}(x) - \max_{i=1,..,n}\hat{y}^i, 0\right). \tag{14.17}$$

If we have no observation noise, $\bar{\nu}^{KG,n}(x) \leq \mathbb{E}^n[I^{n+1}(x)]$. Furthermore, $\mathbb{E}^n[I^{n+1}(x)] = \mathbb{E}^n\left[\max_{i=0,..,n}\theta^{n+1}(x^i)|x^n = x\right] - \max_{i=0,..,n-1}\theta^n(x^i)$. We show this using

$$\begin{aligned}
\bar{\nu}^{KG,n}(x) &= \mathbb{E}^n\left[\max_{i=0,..,n}\theta^{n+1}(x^i)|x^n = x\right] - \max_{i=0,..,n}\theta^n(x^i)\Big|_{x=x^n} \\
&\leq \mathbb{E}^n\left[\max_{i=0,..,n}\theta^{n+1}(x^i)\Big|x = x^n\right] - \max_{i=0,..,n-1}\theta^n(x^i) \\
&= \mathbb{E}^n\left[\max\left(\theta^{n+1}(x^n), \max_{i=0,..,n-1}\theta^n(x^i)\right)\Big|x = x^n\right] - \max_{i=0,..,n-1}\theta^n(x^i) \\
&= \mathbb{E}^n\left[\max\left(\theta^{n+1}(x^n), \max_{i=1,..,n}\hat{y}^i\right)\Big|x = x^n\right] - \max_{i=1,..,n}\hat{y}^i \\
&= \mathbb{E}^n\left[\max\left(\theta^{n+1}(x^n) - \max_{i=1,..,n}\hat{y}^i, 0\right)\Big|x = x^n\right] \\
&= \mathbb{E}^n[I^{n+1}(x)]. \qquad (14.18)
\end{aligned}$$

In the third line, we used the fact that, given what we know at time n, $\hat{y}^{i+1} = \theta^n(x^i) = \theta^{n+1}(x^i)$ for $i = 0, ..., n-1$ since there is no observation noise. The EGO algorithm maximizes the expected improvement given by equation (14.18) at each iteration. If we assume that there is no experimental noise, the expectation of (14.17) has a closed-form solution

$$\nu^{EGO,n}(x) = \tilde{\sigma}_x\left(\Sigma^n, x\right) f\left(\frac{\theta^n(x) - \max_{i=1,...,n}\hat{y}^i}{\tilde{\sigma}_x\left(\Sigma^n, x\right)}\right),$$

much like its analog in (4.34). This policy closely parallels the knowledge gradient for the case where there is no observation noise, and does not account for correlations in the belief structure.

14.4 EXPERIMENTS

It will always be hard to draw firm conclusions about the performance of algorithms in real experiments. In this section, we draw on a series of experiments reported in Scott et al. (2011), where SKO and KGCP were compared using a series of standard test problems as well as new test problems that were generated directly from the Gaussian process model. The standard test problems are known as Ackley, Branin, Hartman3 and the Six Hump Camelback. The Gaussian process datasets generated scalar functions using $\alpha = .1, 1.0, 10.0$, which captures the structural deviation between the true surface and the approximation. Recall that for smaller values of α, these deviations are described by smooth undulations, while larger values of α produce high frequency ripples. All of the test problems were run with three different levels of experimental noise, where we used $\lambda = .1, 1.0$ and 10.0.

Table 14.1 summarizes seven test problems, along with the results of comparisons of SKO and KGCP using the expected opportunity cost as the performance metric. These results suggest that the knowledge gradient algorithm consistently outperforms SKO, sometimes dramatically so. There was only one dataset where SKO outperformed KGCP in a statistically significant way (for the Six Hump Camelback, with $\lambda = 10$), but the expected opportunity cost for KGCP was 1.0264 versus .8488 for SKO, which

is a small difference, especially compared to the relative performance for some of the other problems. As a general pattern, the relative improvement of KGCP over SKO was largest for problems where the experimental noise was the smallest. We suspect that as a general rule, differences in algorithms will become less noticeable as the experimental noise becomes larger.

		KGCP		SKO	
Test Function	$\sqrt{\lambda}$	$\mathbb{E}(OC)$	$\sigma(OC)$	$\mathbb{E}(OC)$	$\sigma(OC)$
Ackley 5 ($\mathcal{X} = [-15, 30]^5$)	$\sqrt{.1}$	$\mathbf{5.7304}$	.1874	7.8130	.1802
	$\sqrt{1.0}$	$\mathbf{10.8315}$	.2413	12.6346	.2088
$p = 5, \sigma = 1.126$	$\sqrt{10.0}$	$\mathbf{17.3670}$	.1477	18.1126	.1156
Branin	$\sqrt{.1}$	$\mathbf{.0141}$	.0044	.0460	.0023
	$\sqrt{1.0}$	$\mathbf{.0462}$	.0039	.1284	.0218
$p = 2, \sigma = 51.885$	$\sqrt{10.0}$	$\mathbf{.2827}$	.0186	.4396	.0248
Hartman3	$\sqrt{.1}$	$\mathbf{.0690}$	.0063	.1079	.0075
	$\sqrt{1.0}$	.5336	.0296	.5012	.0216
$p = 3, \sigma = .938$	$\sqrt{10.0}$	1.8200	.0541	1.8370	.0510
Six Hump Camelback	$\sqrt{.1}$	$\mathbf{.0714}$	.0087	.1112	.0059
	$\sqrt{1.0}$	.3208	.0192	.3597	.0156
$p = 2, \sigma = 3.181$	$\sqrt{10.0}$	1.0264	.0391	$\mathbf{.8488}$	.0370
GP ($\alpha = .1, \beta = 100$)	$\sqrt{.1}$	.0076	.0057	.0195	.0041
	$\sqrt{1.0}$	.0454	.0243	.0888	.0226
$p = 1, \sigma = 8.417$	$\sqrt{10.0}$	.3518	.0587	.2426	.0216
GP ($\alpha = 1, \beta = 100$)	$\sqrt{.1}$	$\mathbf{.0077}$	.0022	.0765	.0311
	$\sqrt{1.0}$	$\mathbf{.0270}$	.0045	.1993	.0486
$p = 1, \sigma = 9.909$	$\sqrt{10.0}$	$\mathbf{.4605}$	.1028	.6225	.0669
GP ($\alpha = 10, \beta = 100$)	$\sqrt{.1}$	$\mathbf{.1074}$	.0259	.5302	.0799
	$\sqrt{1.0}$	$\mathbf{.1846}$	.0286	.6638	.0839
$p = 1, \sigma = 10.269$	$\sqrt{10.0}$	$\mathbf{1.0239}$	.1021	1.8273	.1450

Table 14.1 Comparison of the knowledge gradient algorithm for continuous parameters to sequential kriging optimization for noisy experiments, from experiments reported in Scott et al. (2011).

14.5 EXTENSION TO HIGHER DIMENSIONAL PROBLEMS

The algorithms we have presented in this chapter can be used for multidimensional parameter vectors, but care has to be used when optimizing over higher dimensional parameter spaces. It is not unusual to see algorithms which work in multiple dimensions being tested on problems with one or two dimensions. Transitioning to as few as five dimensions can actually be quite difficult for certain algorithms. Searching a parameter space with 10 or 20 dimensions is dramatically harder than five dimensions without making suitable approximations.

There are several strategies that can be used to search higher dimensional parameter spaces. Some of these include

- Make a random sample of the parameter space $\mathcal{X}$, producing a set of possible parameters $x_1, x_2, \ldots, x_M$. Now, use traditional optimal learning policies for discrete alternatives.
- Assume that the function $\mu(x)$ is approximately separable in x, and use this property to search each dimension separately (possibly using the techniques in this chapter for continuous parameters).
- Stitch together a solution by optimizing over small subsets of dimensions. This can be done in parallel for different subsets of dimensions, after which new, overlapping subsets can be chosen.

The challenge of dimensionality exists even with classical stochastic search algorithms, where efficient learning may not be an issue. For example, imagine that we have a stochastic function $F(x, W)$ that depends on a controllable vector x and a random vector W. Let $W(\omega)$ be a sample realization of $F(x, W)$, and assume that we are able to compute the gradient $\nabla_x F(x, W(\omega))$. We might be able to solve the optimization problem

$$\max_x \mathbb{E} F(x, W)$$

using a classical stochastic gradient algorithm of the form

$$x^n = x^{n-1} + \alpha_{n-1} \nabla_x F(x^{n-1}, W(\omega^n)). \tag{14.19}$$

Stochastic gradient algorithms such as (14.19) can handle problems with thousands of dimensions, but can exhibit very slow convergence, even when we assume that we can compute the gradient $\nabla_x F(x^{n-1}, W(\omega^n))$. These algorithms work by basically linearizing the function and dealing with each dimension individually. Although we have not seen this in the research literature, it is possible to envision the use of optimal learning techniques applied in each dimension individually. However, for high dimensional problems, we would be limited to very simple policies. Needless to say, optimal learning for high dimensional problems is an interesting area of research.

14.6 BIBLIOGRAPHIC NOTES

There is an extensive literature on stochastic search for continuous variables which we have not covered in this chapter because they do not explicitly address the issue of value of information. An excellent review of the literature as of 2003 is given by Spall (2003). This literature can be largely divided between algorithms that assume that we have access to gradient information (or at least stochastic gradient), which is our focus.

There is a number of papers tackling the problem of optimizing noisy functions without derivative information. These algorithms primarily depend on methods for

Monte Carlo sampling of the search region $\mathcal{X}$, and while everyone is looking for algorithms with fast convergence, most of the theory focuses on asymptotic convergence analysis (assuming an off-line application) while convergence rates are studied empirically. Benveniste et al. (1990), Kushner & Yin (1997) and Kushner & Yin (2003) are important references for the theory behind stochastic search algorithms. Some recent contributions include adaptive search with resampling (Andradóttir & Prudius (2010)) and model reference adaptive search (Hu et al. (2007)).

Sections 14.1 - Our belief model is based on material presented in Sacks et al. (1989) and Frazier et al. (2009). Our presentation of Gaussian process regression is based on Rasmussen & Williams (2006); the material on regression kriging in Forrester et al. (2008). The recursive updating equations for the means and variances are based on Frazier et al. (2009). The maximum likelihood estimation method for the Gaussian process regression model is based on Rasmussen & Williams (2006).

Sections 14.2 - Sequential kriging optimization was proposed by Huang et al. (2006). Stein (1999) provides a thorough introduction to the field of kriging, which evolved from the field of spatial statistics.

Sections **??** - The adaptation of the knowledge gradient for continuous alternatives was developed by Scott et al. (2011).

Sections 14.3 - Efficient global optimization was proposed by Jones et al. (1998*b*).

Sections 14.4 - We draw on a series of experiments reported in Scott et al. (2011). Our test functions are culled from Frazier et al. (2009), Huang et al. (2006) and Jones et al. (1998*b*). See also Scott et al. (2010) for additional empirical work on calibration of an airline business simulator.

CHAPTER 15

LEARNING WITH A PHYSICAL STATE

All of the problems that we have considered in earlier chapters have one characteristic in common. In each of these problems, the decision we make depends only on our belief state about the problem. The decision itself can be discrete (as in Chapter 3), a subset (Chapter **??**), a scalar (Chapter 10), or continuous (Chapter 14), and we have considered many different types of objective functions (online, offline, or linked to a complex implementation decision) and different belief structures (lookup table, parametric and nonparametric). However, within each specific problem class, the set of possible decisions has always stayed the same over time, and every policy we have looked at has made decisions based purely on the available information.

There are many problems where this is not the case, and our decision also depends on a *physical state* as well as a belief state. A simple example arises when we are trying to learn the travel times on links in a transportation network. We can traverse from node i to node j to learn more about the time τ_{ij} to get from i to j, but doing so puts us at node j, which changes the decisions we can consider next. If we decide to go from i to j, we have to balance the information we gain about τ_{ij} against the impact of now being at location j.

Another application arises in medical treatments. Consider the problem of treating a diabetes patient. The doctor has certain beliefs about the effectiveness of different treatments. At the same time, the doctor also observes the results of blood sugar tests that are regularly undergone by the patient. The decision to recommend a

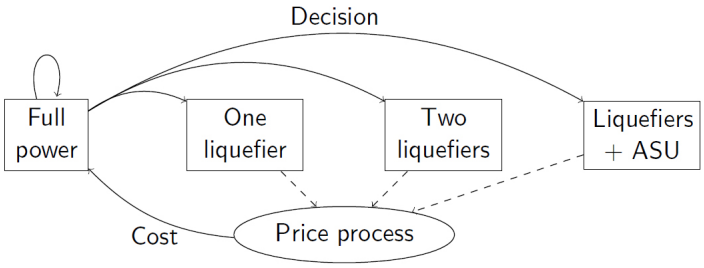


Figure 15.1 An operational flexibility problem, where the state of the power plant affects the energy consumed and the cost of resuming operations.

particular type of treatment is based on the doctor's beliefs about the effectiveness of that treatment, but also on the patient's physical condition. For example, a class of diabetes drugs known as "secretagogues" works by stimulating pancreatic cells to release insulin. Secretagogues can work well in reducing blood sugar, but may cause weight gain as a side effect. The decision to try a treatment can change the physical state of the patient (captured by his weight), which can then change the decisions we can make in the future.

A second example involves the management of a sensor moving around to collect information. This might be a robot sensing radiation near an accident, or a medical technician collecting information about the prevalence of a viral outbreak in the population. The information can be used to help us design response strategies. If we collect information at one location, it reduces the cost of collecting information at nearby locations. Our decision of what information to collect depends on the physical location of the sensor.

We use the term "physical state" somewhat loosely, but we always mean a state variable that changes the decisions that we are allowed to make. For example, imagine that we are selling a product where we can change the price and then learn how the market might respond to the price. We may feel that the market will respond badly to rapid changes in the price, and as a result we impose a restriction that we cannot change the price by more than one dollar. If we charge $20 during one week, we cannot then try charging $28 or $15 the next week. While we may not think of price as a physical state, this falls within the problem class that we wish to consider.

There are many practical problems where learning arises which also have a physical state. Below are some additional examples of problems that have a physical state:

- Operational flexibility - A manufacturer of industrial gases buys energy on the real-time market to run its cryogenic plants. The spot price of energy is subject to very sudden spikes, known as "coincident peaks" in the energy literature. Each plant runs an air separation unit (ASU) which breaks up air into component gases, as well as one or two recycle compressors or liquefiers. Shutting down all of these components allows us to save more money during a coincident peak, but if the peak never arrives, we will waste a lot of energy ramping back up to normal operations (which can take upwards of 24 hours). We have a belief state that helps us predict coincident peaks, but the physical state of the power plant also affects the costs incurred.

- Competitive games - Imagine a game of chess. Each player has a certain belief about the opponent's strategy, and uses this belief to decide on the next move. However, the decisions a player can make depends on the state of the board.
- Pricing with finite inventory - A store sets a price for a product with the goal of maximizing revenue. As in Chapters 7 and **??**, we may wish to experiment with different prices and observe the effect on revenues. However, if we have finite inventory, our decision will depend on the amount we have in stock as well as on our beliefs about the revenue curve. Experimentation is less useful when there is less inventory remaining.
- Mutual fund cash balance - A mutual fund needs cash to meet shareholder redemptions. At the same time, keeping more cash on hand means having less money to invest, leading to lower profits. Over time, we gradually learn about shareholder behavior and the rate at which redemptions arise. Our ability to meet redemptions at any given time, however, is constrained by the amount of cash on hand.

The physical state introduces a whole new level of challenge. The question of how to make decisions based on both physical and belief states is at the very frontier of optimal learning. In this chapter, we give a framework for where to begin thinking about this problem. We also suggest a few approaches that integrate the optimal learning concepts we have developed throughout this book into the world of physical states, and close with a discussion of how some of these concepts might be taken further.

15.1 INTRODUCTION TO DYNAMIC PROGRAMMING

Sequential, stochastic decision problems have long been approached using dynamic programming. Assume that we have a process that may be in a physical state $S_t = s$ at time t. (Throughout this book, we have denoted time by n; the reason why we switch to t here will become clear shortly.) If we choose a (discrete) action x, we may land in a state $S_{t+1} = s'$ with probability $p(s'|s, x)$. Assume that we earn a contribution $C(S_t, x)$ from taking action x when we are in state S_t, and that contributions are discounted by $0 < \gamma < 1$ with each passing time period. If $V_t(S_t)$ is the value of being in state S_t at time t, then we would like to choose our action x_t by solving

$$x_t = \arg\max_x \left(C(S_t, x) + \gamma \sum_{s'} p(s'|S_t, x) V_{t+1}(s') \right). \tag{15.1}$$

The problem is that we do not know $V_{t+1}(s')$. If we have a finite horizon problem with T as the last time period, we might assume $V_T(s) = 0$ then compute the remaining value functions using

$$V_t(S_t) = \max_x \left(C(S_t, x) + \gamma \sum_{s'} p(s'|S_t, x) V_{t+1}(s') \right). \tag{15.2}$$

Equation (15.2) needs to be computed by looping over all states S_t. If S_t is discrete and very low dimensional (as in, three or less), this approach can work quite well.

But there are many problems where the state S_t has more than three dimensions, or it may be continuous. For these problems (which are quite common), we encounter what is widely known as the "curse of dimensionality."

15.1.1 Approximate dynamic programming

The dynamic programming community has developed an algorithmic strategy known broadly as *approximate dynamic programming* (ADP) that is designed to circumvent the curse of dimensionality. ADP is actually an umbrella for a variety of algorithms, but its most popular form works as follows. Let's begin by assuming that the state S_t is discrete. Further assume that we are given an approximation $\overline{V}_t^0(s)$ for all states s and times $t = 1, \ldots, T$ (for example, we might start with $\overline{V}_t^0(s) = 0$).

If we use equation (15.2), we are solving the problem by proceeding backward in time. It is this process that requires that we loop over all possible states, because we do not know in advance which states that we might actually visit. With approximate dynamic programming, we are going to progress *forward* in time, starting with a given initial state S_0. If we are in state S_t at time t and choose action x_t (according to some rule that we discuss below), we are then going to observe a sample realization of a random variable W_{t+1} (which was unknown at time t). Finally, we are going to use a *transition function* (also known as the state transition model), which gives us the next state using

$$S_{t+1} = S^M(S_t, x_t, W_{t+1}).$$

We are going to simulate our way from time 0 to time T iteratively. To firm up our notation, let n the iteration index. In iteration n, let S_t^n be the state that we visit at time t. Also let ω^n index the sample path of observations of W_t, which means that $W_{t+1}(\omega^n)$ is the sample realization of what we observe between t and $t+1$ while following sample path ω^n.

We have to figure out how we are going to make decisions. The most natural policy is to mimic equation (15.1) and solve

$$x_t^n = \arg\max_x \left(C(S_t^n, x) + \gamma \sum_{s'} p(s'|S_t^n, x) \overline{V}_{t+1}^{n-1}(s') \right). \tag{15.3}$$

Not surprisingly, we are going to refer to this as a pure exploitation policy, because it means to choose the best action based on our current belief about the downstream values captured by $\overline{V}_{t+1}^{n-1}(s')$. Before we start criticizing this policy (as we will), we need to close the loop by mentioning how we are coming up with these value function approximations. First compute

$$\hat{v}_t^n = C(S_t^n, x_t^n) + \gamma \sum_{s'} p(s'|S_t^n, x_t^n) \overline{V}_{t+1}^{n-1}(s'). \tag{15.4}$$

We can think of $\hat{v}_t^n$ as a sample estimate of the value of being in state S_t^n. This is the value that we experienced while we were following our sample path through the states $S_0^n, S_1^n, \ldots, S_T^n$. It would be nice to mimic (15.2) and just set $V_t(S_t^n) = \hat{v}_t^n$,

but $\hat{v}_t^n$ is a noisy sample estimate. For this reason, we have to do smoothing, which we can do using

$$\overline{V}_t^n(S_t^n) = (1-\alpha_{n-1})\overline{V}_t^{n-1}(S_t^n) + \alpha_{n-1}\hat{v}_t^n.$$

Here, α_{n-1} is known as a stepsize, which is a parameter between 0 and 1. It might be a constant (such as $\alpha_{n-1} = .001$), but more often it is assigned a declining sequence such as $\alpha_n = a/(a+n-1)$, where a is a tunable parameter.

Notice that, in the above presentation, we have two time indices, n and t. This is because we considered a finite-horizon problem ending at time T. The index t denotes a stage of the problem, whereas the index n denotes the number of times we have updated our approximation $\bar{V}$ of the value function. The algorithm works by making multiple passes through the problem, so that we solve (15.3) backward for $t = T-1, ..., 1$ for each fixed value of n.

We can also accommodate infinite-horizon problems ($T \rightarrow \infty$), for which the optimality equation (15.2) becomes

$$V(S) = \max_x \big(C(S,x) + \gamma \sum_{s'} p(s'|S,x)V(s')\big). \tag{15.5}$$

for all states S. In this case, the ADP algorithm merges the two time indices into one, such that (15.3) becomes

$$x^n = \arg\max_x \left(C(S^n,x) + \gamma \sum_{s'} p(s'|S^n,x)\overline{V}^{n-1}(s') \right). \tag{15.6}$$

and (15.4) becomes

$$\hat{v}^n = C(S^n,x^n) + \gamma \sum_{s'} p(s'|S^n,x^n)\overline{V}^{n-1}(s'). \tag{15.7}$$

The approximation is updated using

$$\overline{V}^n(S^n) = (1-\alpha_{n-1})\overline{V}^{n-1}(S^n) + \alpha_{n-1}\hat{v}^n. \tag{15.8}$$

We still run the algorithm by progressing forward in time starting from S^0. Now, a single iteration consists of a visit to a single state S^n, and n is our only time index, as in the rest of the book. We hope that, after a large number of iterations, $\overline{V}^n$ will be close to the true solution of (15.5). To simplify our presentation, we will focus on infinite-horizon problems when developing our algorithms in this chapter, though it is important to keep in mind the wealth of finite-horizon applications.

We now have the beginnings of an algorithm that seems to have considerable appeal. We can handle arbitrarily complex state variables S_t, and even the random information W_t can be quite complex (problems have been solved with thousands of dimensions), because at any point in time we are working with a single state and a single sample realization of W_t. It almost seems like magic, so you know that there has to be something wrong. Indeed, the problem is known as the exploration vs. exploitation problem, something that should be now be quite familiar to readers of this book.

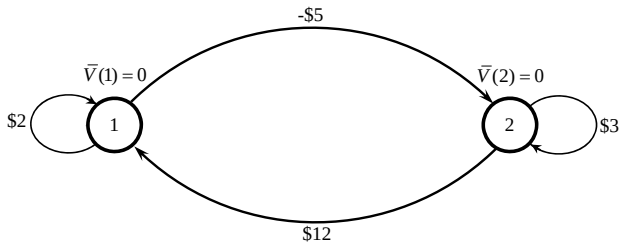


Figure 15.2 A two state dynamic program, where the estimated value of being in each state is currently zero.

15.1.2 The exploration vs. exploitation problem

Unfortunately, the problem with our elementary approximate dynamic programming algorithm is not just that it may not work very well, but that it can, and generally, will, work terribly. To see why, we only need the very simple two-state dynamic program illustrated in Figure 15.2. Assume that we are initially in state 1, and we have to choose an action. We can choose to stay in state 1, in which case we earn a $2 contribution plus the approximate value of being in state 1, where our initial approximation is $\overline{V}(1) = 0$. Alternatively, we can choose to go to state 2, where we incur a loss of $5, but then we receive the value of being in state 2, where again we have an approximate value of $\overline{V}(2) = 0$. Looking at these two alternatives, we would naturally decide to stay in state 1. As a result, we never discover that the transition from state 2 back to state 1 would earn us $12.

This is hardly an artificial example. A more real-world setting is illustrated with an interstate truck driver who arrives in a city and is offered a set of loads of freight. He has to choose one of the loads, which has him moving from his current location to the destination of the load. Let S_t be the current location of the driver (typically a city or three-digit zip code), let x_t be his choice of load from a set of offered loads (these arrive randomly and only become known when he arrives at a location), and let $C(S_t, x)$ be the net profit he receives from accepting the load and moving to its destination. For this simple problem, $S_{t+1} = S^M(S_t, x_t, W_{t+1}) = S^M(S_t, x_t)$ captures the destination of the load (which is deterministic).

A new truck driver might not have any information about the value of being in a city, and as a result might reasonably start with $\overline{V}^0(s) = 1$ for each state s. This means that he would initially take the load paying the most. Over time, he will learn about the value of being in states that he visits, but this will lead to situations where he is choosing between a load that returns him to a city which he has already visited (in which case $\overline{V}^{n-1}(s') > 0$), and a load that takes him to a city he has never visited (which means $\overline{V}^{n-1}(s') = 0$). As a result, there is a natural bias to take loads to cities he has visited before. Sound familiar? This is like returning to restaurants that you know, rather than trying new restaurants. Needless to say, this is fairly intuitive behavior.

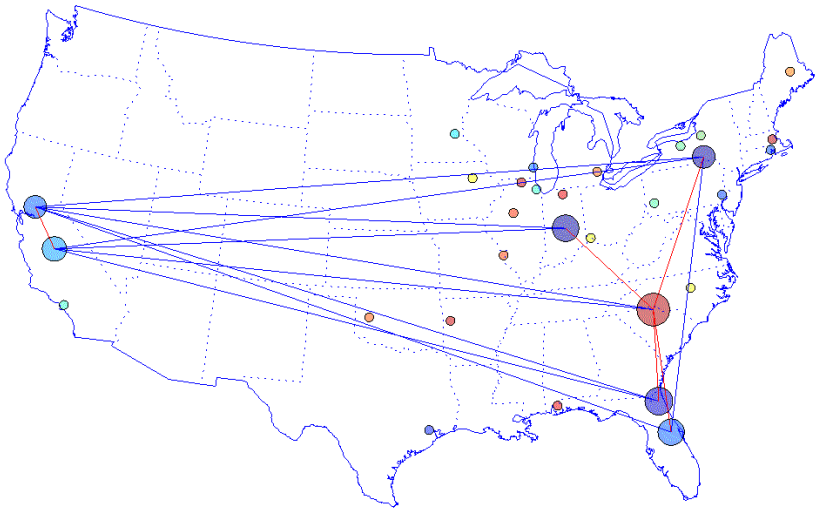


Figure 15.3 The effect of a pure exploitation policy for our nomadic trucker.

The problem with this policy is that, as we would expect, there is a tendency to get caught visiting a small number of cities. Figure 15.3 shows the results of a simulation of this policy, and out of a set of 30 possible locations, the driver finds that he is constantly returning to the same seven cities. This might seem familiar, but it is hardly optimal.

15.1.3 Discussion

Our ADP algorithm has not really solved the curse of dimensionality. While we have eliminated the need to loop over all states to calculate the value of being in a state, we still need good approximations of the value of each state that we *might* visit, as shown in equation (15.3). While this may be much smaller than the size of the full state space, this is still a large number of states.

There are two changes we have to consider to overcome this problem. The first is that we have to introduce *exploration* steps, where we visit a state just to collect information about the state. We have seen this idea throughout this volume, so there should be no surprise that it arises in this setting as well. The second change is that we have to use some form of *generalized learning*. That is, we need to learn more than just the value of being in a single state. Again, this has been a common theme that we have seen before, first through the mechanism of correlated beliefs, and later when we made the transition to parametric belief models as we did in Chapter 7.

First, we present some simple heuristics for exploration. Then, we progress to more formal methods that address both of the issues raised in this discussion. As a rule, the exploration strategies in this chapter are designed for infinite-horizon problems, so we use a single time index n.

15.2 SOME HEURISTIC LEARNING POLICIES

One very simple policy that is frequently used in practice is our old acquaintance epsilon-greedy: at time n, we make a random decision with probability ε, or follow the pure exploitation policy (15.6) with probability $1-\varepsilon$. We face the usual problem of tuning ε, but otherwise, it will be possible to achieve good performance as long as $\mathcal{X}$ is fairly small.

Two more sophisticated exploration strategies developed within the computer science community are known as R-max and E^3. Both of these policies are based on the idea of categorizing the states according to whether we have "enough" or "not enough" information about them.

The R-max policy makes decisions according to the rule

$$X^{Rmax,n}\left(S^n;\alpha^n\right)=\arg\max_x\left(C\left(S^n,x\right)+\gamma\sum_s\rho_s^n\left(S^n,x\right)F\left(s\right)\right)$$

where

$$F\left(s\right)=\begin{cases}R^{\max} & \text{if } s \text{ has been visited fewer than } m \text{ times,}\\ V^n\left(s\right) & \text{if } s \text{ has been visited at least } m \text{ times.}\end{cases}$$

The integer m is a tunable parameter representing the number of times we need to visit a state in order to obtain "enough" information. The value R^{max} is deliberately chosen to be very large, such that we are more likely to choose an action x if it is more likely to lead us to states for which we have little information. Eventually, once we visit every state enough times, R^{max} reduces to the pure exploitation policy . We are still, however, left with the issue of tuning m and R^{max}.

The E^3 policy ("Explicit Explore or Exploit") can be viewed as a modification of epsilon-greedy. As in R-max, we have a parameter m representing the amount of information that is "sufficient" for us to know the value of being in a state. Upon reaching state S^n, our decision is made as follows. If S^n is a state that we have never visited before, we make a random decision. If we have visited S^n before, but fewer than m times, we make the decision that we have tried the fewest number of times among all our previous visits to the state. Finally, if we have visited S^n more than m times, we follow the pure exploitation policy and make our decision according to (15.6). Once again, the policy reduces to pure exploitation once we have sufficiently explored the state space. In the early stages, the policy encourages us to make decisions with which we are unfamiliar, in order to learn about them.

15.3 THE LOCAL BANDIT APPROXIMATION

Our first formal technique, known as the local bandit approximation or LBA, attempts to convert the dynamic program into a multi-armed bandit problem at each time step, and then uses a Gittins-like calculation to make the next decision. The multi-armed bandit problem, which we covered in Chapter 5, has no physical state. We make our decision based purely on our knowledge about the rewards obtainable from M

different arms. To apply bandit-style thinking to our current problem, we need a way to remove the physical state.

Let us start by defining a policy π_n that always makes decisions according to (15.6) for $\bar{V}^{n-1}$ fixed at the current value of n. At first glance, this sounds like the pure exploitation policy. However, pure exploitation updates the approximation $\bar{V}^{n-1}$ using (15.8) after $\hat{v}^n$ is observed. That is, pure exploitation assumes that the approximation is fixed when we make a decision, but we continue to update the approximation in every iteration. This is exactly the concept of experiential learning discussed back in Section **??**.

By contrast, the policy π_n *always* makes decisions according to the *fixed* approximation $\bar{V}^{n-1}$. It is analogous to the "Stop" policy discussed in Section 5.4. If we use this policy, we will commit to our current value function approximation and stop learning altogether. The dynamic program is thus reduced to a Markov chain (Y_n) whose transition probabilities are given by

$$P\left(Y_{n+1} = s' \,|\, Y_n = s\right) = p\left(s' \,|\, s, X^{\pi,n}\left(s\right)\right)$$

where $X^{\pi,n}\left(s\right)$ is the decision that solves (15.6) for $S^n = s$. Our reward for visiting state s is thus $\tilde{C}\left(s\right) = C\left(s, X^{\pi,n}\left(s\right)\right)$.

We use this policy much the same way that we did in Section **??**: we assume that we will make only one more decision before switching to this naive policy. Suppose that S^n is our state at time n. Let us adopt a perspective that is centered around the state S^n. In this time step, we will make a decision x^n and transition to a different state. Suppose that, from that point onward, we will only make decisions according to the policy π_n. This policy will lead us to visit many different states, before we eventually return to S^n. We use the word *sojourn* to describe the period of time in which we travel around states other than S^n. The sojourn ends as soon as we return to this state. Figure 15.4 provides a visual illustration, with states represented by circles and decisions represented by squares.

As long as we use the policy π_n to make decisions, we can study the sojourn via the Markov chain (Y_n). Let

$$\tau_{s,s'} = \min\left\{n \geq 0 \,|\, Y_n = s', Y_0 = s\right\}$$

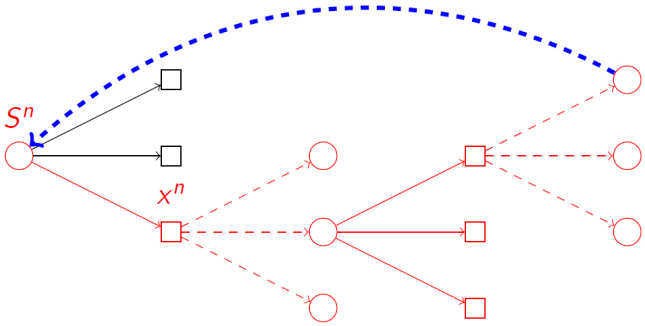


Figure 15.4 If we visit a state S^n and make decision x^n, we enter a sojourn that takes us to other states and actions. The sojourn ends as soon as we return to S^n.

be the number of transitions needed for the Markov chain (Y_n) to reach state s' for the first time, provided that it started in state s. The quantities

$$\begin{aligned} R(s,s') &= \mathbb{E}\left(\sum_{n=0}^{\tau_{s,s'}-1} \gamma^n f(Y_n) \mid Y_0 = s\right) \\ T(s,s') &= \mathbb{E}\left(\sum_{n=0}^{\tau_{s,s'}-1} \gamma^n \mid Y_0 = s\right) \end{aligned}$$

represent the average discount reward collected and the average discounted time elapsed, respectively, before our first visit to state s'. We can compute these quantities using first-transition analysis on Y_n. First,

$$R(s,s') = \begin{cases} f(s) + \gamma \sum_{s'' \in \mathcal{S}} P(Y_{n+1} = s'' \mid Y_n = s) R(s'',s') & \text{if } s \neq s' \\ 0 & \text{if } s = s' \end{cases}.$$

Similarly,

$$T(s,s') = \begin{cases} 1 + \gamma \sum_{s'' \in \mathcal{S}} P(Y_{n+1} = s'' \mid Y_n = s) T(s'',s') & \text{if } s \neq s' \\ 0 & \text{if } s = s' \end{cases}.$$

It follows that, if we visit state S^n and choose action x, the expected reward collected during the ensuing sojourn, and the expected length of the sojourn, are given by

$$\begin{aligned} R^{sojourn}(S^n,x) &= C(S^n,x) + \gamma \sum_{s' \in \mathcal{S}} p(s' \mid S^n,x) R(s',S^n), \\ T^{sojourn}(S^n,x) &= 1 + \gamma \sum_{s' \in \mathcal{S}} p(s' \mid S^n,x) T(s',S^n). \end{aligned}$$

The quantity

$$\nu^{LBA,n}(S^n,x) = \frac{R^{sojourn}(S^n,x)}{T^{sojourn}(S^n,x)} \tag{15.9}$$

represents the expected reward per expected unit time that we receive during the sojourn.

It turns out that the ratio in (15.9) resembles a Gittins index. In Chapter 5, we defined the Gittins index of a bandit arm as a risk-free reward that would make us indifferent between playing the arm repeatedly or just collecting the reward. An equivalent definition describes the Gittins index as the long-term expected reward per play that we can obtain from playing the arm over and over.

Thus, in a sense, this analysis is converting the dynamic program into a bandit problem. When we visit state S^n, we behave as if S^n were the only state in the dynamic program. All other states are grouped together into the sojourn. The different decisions available in state S^n are viewed as "arms" of a bandit, and the expected reward that we collect during the sojourn after pulling the "arm" x is treated as the "reward" of that arm. Thus, (15.9) is viewed as the Gittins index of decision x. We then make decisions according to the simple rule

$$X^{LBA,n}(S^n) = \arg\max_x \nu_x^{LBA,n},$$

analogous to the Gittins index policy for multi-armed bandits.

Unlike the original Gittins index policy, LBA is not optimal, because we are using the policy π_n instead of the optimal policy in the analysis of the sojourn. However, when this approach was first introduced, it was a pioneering example of an attempt to bring optimal learning ideas over to learning with a physical state. It also illustrates some of the difficulties inherent in carrying optimal learning concepts over to the physical state setting.

15.4 THE KNOWLEDGE GRADIENT IN DYNAMIC PROGRAMMING

We are going to develop the concept of the knowledge gradient in the context of dynamic programs with a physical state. We do this in two steps. First, we are going to introduce the idea of approximating a value function using a linear regression model. This makes it possible to generalize what we learn from a visit to a single state to what we know about all states. Then, we show how to compute the knowledge gradient in this setting.

15.4.1 Generalized learning using basis functions

A limitation of all the policies we have reviewed up to now is that they are limited by the use of a lookup table belief model for value functions. A much more powerful way of learning uses some form of generalization. While there are several ways of doing this, by far the most popular involves using linear regression to approximate the value of being in a state.

Before we do this, we have to solve a small technical difficulty. Our pure exploitation policy would have chosen an action using

$$x_t^n = \arg\max_x \left(C(S_t^n, x) + \gamma \sum_{s'} p(s'|S_t^n, x) \overline{V}_{t+1}^{n-1}(s') \right).$$

If we have a large state space (which is the only time where learning is an issue), then we would never be able to compute the one-step transition matrix $p(s'|S_t^n, x_t^n)$. A more natural way to write this equation is using its expectation form, where we would write

$$x_t^n = \arg\max_x \left(C(S_t^n, x) + \gamma \mathbb{E}\{\overline{V}_{t+1}^{n-1}(S'|S_t^n, x\} \right)$$

with S' denoting the randomly determined next state. We have not solved anything yet, because we can only compute the expectation for special cases. We can get around this by using the concept of the *post-decision state variable* which we designate S_t^x. The post-decision state is the state immediately after we have made a decision, but before we have learned any new information. For example, imagine we have a simple inventory problem where at time t we need to serve a random demand $\hat{D}_t$ with some resource, where R_t is the amount of resource (medicine, money, vaccines, water, power plants...) currently available. The (pre-decision) state is

given by $S_t = (R_t, \hat{D}_t)$. Unsatisfied demands are lost. The resource variable evolves according to

$$R_{t+1} = R_t - x_t + \hat{R}_{t+1},$$

where $0 \leq x_t \leq R_t$ is our decision of how much to withdraw to satisfy demands, and $\hat{R}_{t+1}$ represents random additions (blood donation, cash deposits). The post decision state $S_t^x = R_t^x$, where

$$R_t^x = R_t - x_t.$$

The key idea with the post-decision state is that it is a deterministic function of the state S_t and the action x_t. More generally, we assume we have a function $S^{M,x}(S_t, x_t)$ that returns the post-decision state S_t^x.

Using this concept, our exploitation policy would be computed using

$$x_t^n = \arg\max_x \left(C(S_t^n, x) + \gamma \overline{V}_t^{x,n-1}(S^{M,x}(S_t^n, x)) \right), \tag{15.10}$$

where $\overline{V}_t^{x,n-1}(S_t^x)$ is our value function approximation around the post-decision state S_t^x. Note that this is a deterministic optimization problem, which is much easier to solve. Below, we will again focus on the infinite-horizon formulation, in which (15.10) becomes

$$x^n = \arg\max_x \left(C(S^n, x) + \gamma \overline{V}^{x,n-1}(S^{M,x}(S^n, x)) \right), \tag{15.11}$$

We now just have to resolve the problem of approximating the value function. For this purpose, we propose using a simple linear model of the form

$$V(S^{x,n}) = \sum_{f \in \mathcal{F}} \theta_f \phi_f \left(S^{x,n} \right) = (\phi(S^{x,n}))^T \theta.$$

Here, $\phi_f(S)$ is known as a *feature* or *basis function*. These are functions that capture what an expert feels are the important characteristics of the state variable. Instead of having to estimate the value of being in each state, we now have to find the regression vector θ.

In Chapter 7, we showed how we could use the knowledge gradient to estimate the parameters of a linear regression model. We are going to do the same here, but now we have to deal with the issue of a physical state. We are also going to adopt another convention that we have used widely in our presentation which is the idea that we have a prior for our regression model. That is, we are going to assume that we have an initial estimate θ^0 as a starting point that is an unbiased estimate of the true value. This is like saying that we have a rough idea of what the value function looks like. This assumption allows us to claim that $\hat{v}^n$ is an unbiased observation of the true value $V(S^{x,n-1})$ of being in the post-decision state $S^{x,n-1}$. We are going to further assume that the error $\epsilon = \hat{v}^n - V(S^{x,n-1})$, in addition to having mean zero (since we assume that our prior is unbiased) also has a known error with variance σ_ϵ^2.

With this foundation, we can quickly describe the equations needed to update our regression vector θ. We start by defining the matrix

$$X^n = \begin{bmatrix} \phi_1\left(S^{x,0}\right) & \cdots & \phi_F\left(S^{x,0}\right) \\ \vdots & \ddots & \vdots \\ \phi_1\left(S^{x,n-1}\right) & \cdots & \phi_F\left(S^{x,n-1}\right) \end{bmatrix}.$$

This matrix consists of the value of each basis function evaluated at every state that we have visited. Also define the vector of observations

$$y^n = \begin{bmatrix} \hat{v}^1 \\ \vdots \\ \hat{v}^n \end{bmatrix}.$$

If we were using traditional linear regression, we could compute the best regression vector θ^n using

$$\theta^n = \left[\left(X^n\right)^T X^n\right]^{-1} \left(X^n\right)^T y^n.$$

We can compute θ^n recursively without having to perform matrix inversion. First let $B^n = \left[\left(X^n\right)^T X^n\right]^{-1}$. The updating equation for θ^n is given by

$$\theta^n = \theta^{n-1} + \frac{\hat{v}^n - \phi\left(S^{x,n-1}\right)^T \theta^{n-1}}{1 + \phi\left(S^{x,n-1}\right)^T B^{n-1} \phi\left(S^{x,n-1}\right)} B^{n-1} \phi\left(S^{x,n-1}\right). \quad (15.12)$$

We can also compute the matrix B^n recursively using

$$B^n = B^{n-1} - \frac{B^{n-1}\phi\left(S^{x,n-1}\right)\phi\left(S^{x,n-1}\right)^T B^{n-1}}{1 + \phi\left(S^{x,n-1}\right)^T B^{n-1}\phi\left(S^{x,n-1}\right)}. \quad (15.13)$$

We are going to make the somewhat heroic assumption that the observations $\hat{v}^n$ are independent and identically distributed with variance σ_ϵ^2. This means that the covariance matrix for the observation vector Y^n is $I\sigma_\epsilon^2$, where I is the identity matrix. Our estimate of the regression vector is given by

$$\theta^n = B^n (X^n)^T Y^n.$$

That is, our approximation of the value of state $S^{x,n}$ at time n is $\bar{V}^n\left(S^n\right) = \phi\left(S^{x,n}\right)^T \theta^n$. It is also possible to show (as we did in Chapter 7) that the covariance matrix for θ is given by

$$\Sigma^{\theta,n} = \sigma_\epsilon^2 B^n.$$

We can now rewrite equation (15.12) as

$$\theta^n = \theta^{n-1} - \frac{\hat{v}^n - \phi\left(S^{x,n-1}\right)^T \theta^{n-1}}{\sigma_\varepsilon^2 + \phi\left(S^{x,n-1}\right)^T \Sigma^{\theta,n-1}\phi\left(S^{x,n-1}\right)} \Sigma^{\theta,n-1}\phi\left(S^{x,n-1}\right) \quad (15.14)$$

and

$$\Sigma^{\theta,n} = \Sigma^{\theta,n-1} - \frac{\Sigma^{\theta,n-1}\phi\left(S^{x,n-1}\right)\phi\left(S^{x,n-1}\right)^T \Sigma^{\theta,n-1}}{\sigma_\varepsilon^2 + \phi\left(S^{x,n-1}\right)^T \Sigma^{\theta,n-1}\phi\left(S^{x,n-1}\right)}. \quad (15.15)$$

15.4.2 The knowledge gradient

We are now ready to compute a knowledge gradient for each potential action in a dynamic program. We are going to build on the principles we have developed earlier in this volume, but care has to be used in the presence of a physical state.

We start by writing our exploitation policy (doing the best given what we know now) in the form

$$X^{Exp,n}\left(S^n\right) = \arg\max_x Q^n(S^n, x) \tag{15.16}$$

where

$$Q^n(S^n, x) = C(S^n, x) + \gamma\phi\left(S^{x,n}\right)^T \theta^n.$$

When we use the knowledge gradient, we capture the fact that our decision will generate information (presumably from an exogenous distribution) that would allow us to update our regression vector, giving us an updated estimate θ^{n+1} (which is random at time n, when we choose our action). We can write the knowledge gradient policy as

$$X^{KG,n}\left(S^n\right) = \arg\max_x Q^{KG,n}\left(S^n, x\right) \tag{15.17}$$

where

$$Q^{KG,n}(S^n, x) = C(S^n, x) + \gamma\mathbb{E}_x^n \overline{V}^{n+1}(S^{x,n}). \tag{15.18}$$

To understand this, it is useful to talk through the steps. If we are in state S^n and choose action x, we would then observe a random quantity W^{n+1} that would determine the next pre-decision state S^{n+1}. Once in state S^{n+1}, we would observe $\hat{v}^{n+1}$, which would then be used to update our parameter vector θ^n to give us an updated estimate θ^{n+1}. However, we are still in state S^n and W^{n+1} (and therefore S^{n+1} and $\hat{v}^{n+1}$) is a random variable. For this reason, we have to take the expectation $\mathbb{E}_x^n$ of $\overline{V}^{n+1}$ given what we know now (given by S^n, which captures both our physical state as well as our belief state), and the action x that we are considering.

Note that while we are in state S^n, the information we gain from a transition from S^n to S^{n+1} from taking action x updates our belief about the value function that may benefit us even though our next decision is from state S^{n+1} (which is random when we are still in state S^n). We did not enjoy this property when we used a lookup table representation. If we learned something about state s' from a decision to move from s to s', this new information was of little or no value while we were in state s'. The information did not add value until we were in some other state where we might consider a transition back to state s. We could apply the same principle using a lookup table representation if we exploited the idea of correlated beliefs.

We start by expanding equation (15.18) using

$$\begin{aligned} Q^{KG,n}(S^n, x) = \; & C(S^n, x) \\ & +\gamma \sum_{S^{n+1}} p(S^{n+1}|S^n, x)\mathbb{E}_x^n \max_{x'} Q^{n+1}(S^{n+1}, x') \end{aligned} \tag{15.19}$$

To compute the expectation in (15.19) we build on ideas we first presented in Chapter 4 and write θ^{n+1} using

$$\theta^{n+1} \sim \theta^n + \frac{\Sigma^{\theta,n+1}\phi(S^{x,n})}{\sqrt{\sigma_\varepsilon^2 + \phi(S^{x,n})^T\Sigma^{\theta,n}\phi(S^{x,n})}}Z, \tag{15.20}$$

where Z is a standard normal random variable with mean 0 and variance 1. Multiplying both sides in (15.20) by $\phi(S^y)$, for some decision y not necessarily equal to x, gives us the relationship between $\bar{V}^n$ and $\bar{V}^{n+1}$ which we can write as

$$\bar{V}^{n+1}(S^y) \sim \bar{V}^n(S^y) + \frac{\phi(S^y)^T\Sigma^{\theta,n-1}\phi(S^{x,n})}{\sqrt{\sigma^2_\varepsilon + \phi(S^{x,n})^T\Sigma^{\theta,n}\phi(S^{x,n})}}Z. \tag{15.21}$$

The quantity $\phi(S^y)^T\Sigma^{\theta,n-1}\phi(S^{x,n})$ is precisely $Cov^n(S^y, S^{x,n})$.

Using equation (15.21), the last term in (15.19) can be written as

$$\mathbb{E}^n_x \max_{x'} Q^{n+1}(S^{n+1}, x') = \mathbb{E} \max_{x'} (a^n_{x'} + b^n_{x'} Z) \tag{15.22}$$

where

$$\begin{aligned} a^n_{x'} &= C(S^{n+1}, x') + \gamma \bar{V}^n(S^{M,x}(S^{n+1}, x')), \\ b^n_{x'} &= \gamma \frac{\phi(S^{M,x}(S^{n+1}, x'))^T\Sigma^{\theta,n-1}\phi(S^{x,n})}{\sqrt{\sigma^2_\varepsilon + \phi(S^{x,n})^T\Sigma^{\theta,n}\phi(S^{x,n})}}. \end{aligned}$$

Now we have to use the methods for computing the knowledge gradient with correlated beliefs (see Section 4.5) to compute (15.22) using

$$\mathbb{E} \max_{x'}(a^n_{x'} + b^n_{x'} Z) = \left(\max_{x'} a^n_{x'}\right) + \sum_{x'} (b^n_{x'+1} - b^n_{x'}) f(-|c_{x'}|),$$

where f is defined as $f(z) = z\Phi(z) + \phi(z)$, where ϕ and Φ denote the standard Gaussian pdf and cdf. Readers familiar with Section 4.5 will remember that it is important that the actions be sorted so that the slopes b^n_y are sorted in increasing order, and that dominated lines have been eliminated.

We can now compute the knowledge gradient using

$$\begin{aligned} \nu^{KG,n}(S^{x,n}, S^{n+1}) &= \mathbb{E} \max_{x'} (a^n_{x'} + b^n_{x'} Z) - \max_{x'} a^n_{x'} \\ &= \sum_{x'} \left(b^n_{x'+1} - b^n_{x'}\right) f(-|c_{x'}|). \end{aligned}$$

Since $a^n_{x'} = Q^n(S^{n+1}, x')$, the quantity $\nu^{KG,n}(S^{x,n}, S^{n+1})$ can be viewed as the expected improvement in our estimate of the value of being in state S^{n+1}, obtained as a result of making the random transition from $S^{x,n}$ (which depends on our action) to S^{n+1}. We can substitute the definition of the knowledge gradient back into (15.19) to obtain

$$\begin{aligned} &\sum_{S^{n+1}} p\left(S^{n+1}|S^n, x\right) \mathbb{E}^n_x \max_{x'} Q^{n+1}\left(S^{n+1}, x'\right) \\ &\quad = \sum_{S^{n+1}} p\left(S^{n+1}|S^n, x\right) \max_{x'} Q^n\left(S^{n+1}, x'\right) \\ &\qquad + \sum_{S^{n+1}} p\left(S^{n+1}|S^n, x\right) \nu^{KG,n}\left(S^{x,n}, S^{n+1}\right). \end{aligned}$$

The value of being in the post-decision state is the expected value of being in the next pre-decision state. We can write this expectation using

$$\sum_{S^{n+1}} p(S^{n+1}|S^n, x) \max_{x'} Q^n\left(S^{n+1}, x'\right) \approx \bar{V}^n\left(S^{x,n}\right)$$

and (15.17) reduces to

$$X^{KG,n}(S^n) = \max_x C(S^n,x) + \gamma \bar{V}^n(S^{x,n}) + \sum_{S^{n+1}} p(S^{n+1}|S^n,x)\nu^{KG,n}(S^{x,n},S^{n+1}). \quad (15.23)$$

We see that our knowledge gradient policy looks very similar to the pure exploitation policy in equation (15.16), with the exception that we now have one more term that captures the value of information.

One potential problem with equation (15.23) is that we may not be able to compute the conditional probability $p(S^{n+1}|S^n,x)$, especially if the random variables are continuous. In this case, a quick work-around is to to approximate the expectation in the value of information term using Monte Carlo simulation. In this case, we would use

$$\sum_{S^{n+1}} p\left(S^{n+1}|S^n,x\right)\nu^{KG,n}\left(S^{x,n},S^{n+1}\right) \approx \frac{1}{K}\sum_{k=1}^{K}\nu^{KG,n}\left(S^{x,n},S_k^{n+1}\right),$$

where $S_k^{n+1} = S^{M,W}\left(S^{x,n},W^{n+1}(\omega_k)\right)$ for the kth sample path generated. This technique works well for relatively low values of K, e.g. $K \approx 20$, as long as we have some sort of simulation model from which we can generate transitions.

The structure of the policy in (15.23) is suited for online learning, as we saw in Section 5.4. If we are trying to learn the value functions as quickly as possible in an offline setting, we would focus purely on the value of information, giving us a policy of the form

$$X^{Off,n}(S^n) = \arg\max_x \sum_{S^{n+1}} p\left(S^{n+1}|S^n,x\right)\nu^{KG,n}\left(S^{x,n},S^{n+1}\right). \quad (15.24)$$

This policy will look odd to readers familiar with approximate dynamic programming, since it appears to ignore the contribution from an action and the downstream value. However, in an offline setting, our goal is to just learn the value function approximation, with the hope that if we can learn this function quickly, then a pure exploitation policy (holding the value function fixed) will return good results.

15.4.3 Experiments

Experience with this use of the knowledge gradient is quite limited, but we report on experimental work that is available as this book went to press. We are going to use these ideas to optimize the performance of an energy storage device such as a battery. Let R^n be the amount of energy in the battery as a percentage of total capacity, which is taken to be 35 megawatt-hours. We allow R^n to take integer values between 0 and 100. We can buy energy on the spot market at a price P^n which evolves according to the model

$$\log\frac{P^{n+1}}{P^n} = -\alpha\log\frac{P^n}{P^0} + \sigma Z^{n+1} \quad (15.25)$$

where Z, as before, is a standard normal random variable with mean 0 and variance 1. We use the values $\alpha = 0.0633$ and $\sigma = 0.2$, along with an initial price $P^0 = 30$.

The decision x^n is an integer from -50 to 50 representing how much to charge ($x^n \geq 0$) or discharge ($x^n < 0$) the storage device. The decisions x^n are constrained so that R^n does not go negative or above the maximum capacity of the battery. The post-decision state is given by

$$\begin{aligned} R^{x,n} &= R^n + x^n, \\ P^{x,n} &= P^n. \end{aligned}$$

The next pre-decision state is obtained by letting $R^{n+1} = R^{x,n}$ (we assume no exogenous changes to the amount stored in the battery) and generating P^{n+1} using equation (15.25). The single-period reward is given by $C(S^n, x^n) = -P^n x^n$, the cost incurred or revenue obtained as a result of our decision. Our goal is to determine a storage and withdrawal policy that maximizes total discounted profits over time, using a discount factor of $\gamma = .99$.

The spot price P^n is continuous, and thus we cannot solve the problem exactly. We use a parametric value function approximation with six polynomial basis functions given by $\phi\left(S^{x,n}\right) = \left(1, R^{x.n}, \left(R^{x,n}\right)^2, P^{x,n}, \left(P^{x,n}\right)^2, R^{x,n}P^{x,n}\right)$. We run the KG policy with the Monte Carlo approximation from equation (15.24), with $K = 20$.

For each sample path, the algorithm started with a prior $\theta_1^0 = 15000$, $\theta_2^0, ..., \theta_6^0 = 0$ and a diagonal covariance matrix $\Sigma^{\theta,0}$ with all diagonal elements equal to 500^2. An optimistic value for θ^0 was chosen heuristically to reduce the likelihood of getting stuck in a subset of the state space. The observation noise was chosen to be $\sigma_\varepsilon^2 = 2000^2$.

Most experiments with approximate dynamic programming use a series of training iterations to estimate the value function approximation, after which θ is fixed and a series of simulations are run to determine how well the policy works. This is classical off-line learning, and yet it is most common to use the exploitation policy given in equation (15.10) which has more of an online structure. For this reason, we report the results of four comparisons: we are going to use the online policy $X^{KG,n}(S^n)$ in equation (15.23) and the offline policy $X^{Off,n}(S^n)$ in equation (15.24), tested in two different objective functions. The first is an online objective function, where we assume that we are accumulating contributions in a real setting as we are learning the value function. The second is a more traditional offline setting, where we use a budget to fit the value function approximation and then evaluate the policy using a series of testing iterations. Naturally, we expect the online policy $X^{KG,n}(S^n)$ to do better on the online objective function, and we expect the offline policy $X^{Off,n}(S^n)$ to do better on the offline objective function.

Table 15.1 compares our two learning policies using basis functions against an algorithm that uses a lookup table representation, and an epsilon-greedy learning policy that we introduced in Section 3.2.3. The average performance of each algorithm, averaged over several hundred problems, shows that as we hoped, the offline KG policy (using either basis functions or a lookup table representation) does better on the offline objective function, and the online KG policy does better on the online objective function. Interestingly, the basis functions worked best for the offline objective function while the lookup table worked best for the online objective function. The epsilon-greedy policy performed poorly on both objective functions.

Table 15.1 Means and standard errors for the storage problem.

	Offline objective		Online objective	
	Mean	Avg. SE	Mean	Avg. SE
Offline KG (basis functions)	1136.20	3.54	-342.23	19.96
Online KG (basis functions)	871.13	3.15	44.58	27.71
Offline KG (lookup)	210.43	0.33	-277.38	15.90
Online KG (lookup)	79.36	0.23	160.28	5.43
Epsilon-greedy (param.)	-475.54	2.30	-329.03	25.31

A separate but important issue involves the computational effort to support a policy. For this problem, we can safely say that the epsilon-greedy policy requires virtually no computational effort, and for this reason could perhaps be run for more iterations. The knowledge gradient policy when using a lookup table representation required 0.025 seconds for each calculation. The KG policy when using basis functions required .205 seconds to determine the best action which is small, but not negligible. In a true online problem, these execution times are probably negligible (perhaps we are making decisions once a day or once an hour). However, in many approximate dynamic programming applications, this may be considered a fairly significant amount of overhead. The value of the policy would have to be judged based on the degree to which it accelerated convergence.

One final note of caution has to do with the basis function model itself. If the vector ϕ of basis functions is poorly chosen – for instance, if the features do not adequately describe the value function, or if V is nonlinear in the features – then ADP can perform very poorly. The literature has found cases in which the parameters θ^n will never converge, regardless of how long we run the algorithm. This issue has nothing to do with our particular choice of exploration strategy, but rather is intrinsic to the linear model. Notwithstanding, basis functions continue to be a very popular algorithmic strategy in ADP, due to their remarkable ease of use. In the above model, performance can often be vastly improved by tuning σ_ε^2 or the starting covariance matrix $\Sigma^{\theta,0}$. Unfortunately, we are not able to avoid tunable parameters altogether in our ADP algorithm, the way we did in Chapter 4. It is important to bear in mind, however, that other policies such as epsilon-greedy would add even more tunable parameters if we were to use them together with basis functions.

15.5 AN EXPECTED IMPROVEMENT POLICY

As in Sections **??** and 14.3, our version of KG for learning with a physical state also has a close relative in the form of an expected improvement policy. This time, EI appears under the name "value of perfect information" or VPI. The difference in name is due to the fact that this method was developed independently in the computer science community. Recalling that

$$Q^n(S^n, x) = C(S^n, x) + \gamma\phi\left(S^{x,n}\right)^T \theta^n,$$

and defining

$$\sigma_x^{2,n} = \phi\left(S^{x,n}\right)^T \Sigma^{\theta,n} \phi\left(S^{x,n}\right),$$

we calculate

$$\nu^{VPI,n}\left(S^{x,n}\right) = \sigma_x^{2,n} f\left(-\frac{\left|\phi\left(S^{x,n}\right)^T \theta^n - \max_{x' \neq x} \phi\left(S^{x',n}\right)^T \theta^n\right|}{\sigma_x^{2,n}}\right).$$

We then make our decision according to the formula

$$X^{VPI,n}\left(S^n\right) = \arg\max_x C(S^n, x) + \gamma V^n(S^{x,n}) + \nu^{VPI,n}(S^{x,n}).$$

Just as the other variants of EI, this policy implicitly assumes that we will learn the true value of $S^{x,n}$ immediately after choosing the decision x. Furthermore, just like the KG policy, VPI makes a decision by solving Bellman's equation, plus a value of information term.

15.6 BIBLIOGRAPHIC NOTES

In addition to the models discussed in this chapter, there is a separate stream of literature within the computer science community on learning unknown transition probabilities in a Markov decision process. This particular learning problem uses the Dirichlet distribution that we briefly touched on in Section 2.6 to model the unknown probabilities; the resulting algorithms tend to have a high degree of computational complexity. The LBA method of Duff & Barto (1996) also grew out of this literature. An example of an early approach is Silver (1963). See e.g. Dearden et al. (1999), Steele (2000) and Duff (2003) for additional work on this topic.

Section 15.1 - We give a very streamlined introduction to ADP based on Chapters 3-4 of Powell (2011). Other good references include Puterman (1994), Bertsekas & Tsitsiklis (1996), and Si et al. (2005).

Section 3.2.3 - The R-max method was developed by Brafman & Tennenholtz (2003), whereas E^3 is due to Kearns & Singh (2002). Our heuristics from Chapter 3 can also be used in this setting; implementation is described e.g. in Sutton & Barto (1998) and Kaelbling (1993).

Section 15.3 - The LBA policy was originally proposed by Duff & Barto (1996) as a conceptual algorithm. The full implementation was worked out by Ryzhov et al. (2010), where some experimental results are given. The expression of the Gittins index as expected reward per expected unit time is due to Katehakis & Veinott (1987); see also Dupacova (1995).

Section 15.4 - Basis functions are a standard model for generalized learning; see e.g. Tesauro (1992) for an early treatment. The model continues to see widespread use; see Sutton et al. (2009) for some recent advances. The development of the knowledge gradient for dynamic programming is based on Ryzhov & Powell (2011*b*). The first

derivation of the knowledge gradient in a dynamic programming setting (that is, in the presence of a physical state) was given in Ryzhov & Powell (2011*b*).

Section 15.5 - The VPI policy was proposed by Dearden et al. (1998).

Bibliography

Abernethy, J., Hazan, E. & Rakhlin, A. (2008), Competing in the dark: An efficient algorithm for bandit linear optimization, *in* 'Proceedings of the 21st Annual Conference on Learning Theory', pp. 263–274.

Agrawal, R. (1995), 'Sample mean based index policies with O (log n) regret for the multi-armed bandit problem', *Advances in Applied Probability* **27**(4), 1054–1078.

Andradóttir, S. & Prudius, A. A. (2010), 'Adaptive random search for continuous simulation optimization', *Naval Research Logistics*.

Audibert, J. & Bubeck, S. (2010), Best arm identification in multi-armed bandits, *in* 'Proceedings of the 23rd Annual Conference on Learning Theory (COLT)', pp. 1–14.

Auer, P., Cesa-bianchi, N. & Fischer, P. (2002), 'Finite-time analysis of the multiarmed bandit problem', *Machine Learning* **47**(2), 235–256.

Auer, P., Jaksch, T. & Ortner, R. (2008), Near-optimal regret bounds for reinforcement learning, *in* D. Koller, Y. Bengio, D. Schuurmans, L. Bottou & A. Culotta, eds, 'Advances in Neural Information Processing Systems', Vol. 21, pp. 89–96.

Avriel, M. & Wilde, D. (1966), 'Optimality proof for the symmetric Fibonacci search technique', *Fibonacci Q.* **4**, 265–269.

Banks, J., Nelson, B. L. & J. S. Carson, I. I. (1996), *Discrete-Event System Simulation*, Prentice-Hall, Inc., Englewood Cliffs, N.J.

Bartlett, P., Dani, V., Hayes, T., Kakade, S., Rakhlin, A. & Tewari, A. (2008), High-probability regret bounds for bandit online linear optimization, *in* 'Proceedings of the 21st Annual Conference on Learning Theory', pp. 335–342.

Bechhofer, R. E., Kiefer, J. & Sobel, M. (1968), *Sequential Identification and Ranking Procedures*, University of Chicago Press, Chicago.

Bechhofer, R., Santner, T. & Goldsman, D. (1995), *Design and Analysis of Experiments for Statistical Selection, Screening and Multiple Comparisons*, J.Wiley & Sons, New York.

Bellman, R. & Kalaba, R. (1959), 'On adaptive control processes', *OIRE Trans.* **4**, 1–9.

Benveniste, A., Metivier, M. & Priouret, P. (1990), *Adaptive Algorithms and Stochastic Approximations*, Springer-Verlag, New York.

Berry, D. A. & Fristedt, B. (1985), *Bandit Problems*, Chapman and Hall, London.

Bertsekas, D. P. & Tsitsiklis, J. N. (1996), *Neuro-dynamic programming*, Athena Scientific, Belmont, MA.

Bertsimas, D. J. & Nino-Mora, J. (2000), 'Restless bandits, linear programming relaxations, and a primal-dual index heuristic', *Operations Research* **48**(1), 80–90.

Brafman, R. I. & Tennenholtz, M. (2003), 'R-max – a general polynomial time algorithm for near-optimal reinforcement learning', *Journal of Machine Learning Research* **3**, 213–231.

Branke, J., Chick, S. E. & Schmidt, C. (2005), New developments in ranking and selection: an empirical comparison of the three main approaches, *in* M. Kuhl, N. Steiger, F. Argstrong & J. Joines, eds, 'Proc. 2005 Winter Simulation Conference', IEEE, Inc., Piscataway, NJ, pp. 708–717.

Brezzi, M. & Lai, T. (2002), 'Optimal learning and experimentation in bandit problems', *Journal of Economic Dynamics and Control* **27**(1), 87—-108.

Bruss, F. (1984), 'A unified approach to a class of best choice problems with an unknown number of options', *The Annals of Probability* **12**(3), 882–889.

Bull, A. D. (2011), 'Convergence rates of efficient global optimization algorithms', *Submitted for publication*.

Cayley, A. (1875), 'Mathematical questions with their solutions, No. 4528', *Educational Times*.

Chhabra, M. & Das, S. (2011), Learning the Demand Curve in Posted-Price Digital Goods Auctions, *in* 'Proceedings of the 10th International Conference on Autonomous Agents and Multi-Agent Systems', pp. 63–70.

Chick, S. E. (2003), Expected opportunity cost guarantees and indifference zone selection procedures, *in* S. E. Chick, P. J. Sánchez, D. Ferrin & D. J. Morrice, eds, 'Proceedings of the 2003 Winter Simulation Conference', pp. 465–473.

Chick, S. E. (2006), Subjective Probability and Bayesian Methodology, *in* S. Henderson & B. Nelson, eds, 'Handbooks of Operations Research and Management Science, vol. 13: Simulation', North-Holland Publishing, Amsterdam, pp. 225–258.

Chick, S. E. & Gans, N. (2009), 'Economic analysis of simulation selection problems', *Management Science* **55**(3), 421–437.

Chick, S. E. & Inoue, K. (2001), 'New two-stage and sequential procedures for selecting the best simulated system', *Operations Research* **49**(5), 732—-743.

Chick, S. E. & Wu, Y. (2005), 'Selection procedures with frequentist expected opportunity cost bounds', *Operations Research* **53**(5), 867–878.

Chick, S. E., Branke, J. & Schmidt, C. (2010), 'Sequential Sampling to Myopically Maximize the Expected Value of Information', *INFORMS Journal on Computing* **22**(1), 71–80.

Chvátal, V. (1983), *Linear programming*, WH Freeman.

Cinlar, E. (2011), *Probability and Stochastics*, Springer.

Cohn, D., Atlas, L. & Ladner, R. (1994), 'Improving generalization with active learning', *Machine Learning* **5**(2201), 221.

Cohn, D., Ghahramani, Z. & Jordan, M. (1996), 'Active learning with statistical models', *Journal of Artificial Intelligence Research* **4**, 129–145.

Cozzolino, J., Gonzalez-Zubieta, R. & Miller, R. (1965), Markov decision processes with uncertain transition probabilities, Technical Report 11, Operations Research Center, MIT.

D. R. Barr, M. H. R. (1966), 'An introduction to ranking and selection procedures', *J. Amer. Statist. Assoc.* **61**(315), 640– 646.

Dearden, R., Friedman, N. & Andre, D. (1999), Model-based bayesian exploration, *in* 'Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence', pp. 150–159.

Dearden, R., Friedman, N. & Russell, S. (1998), Bayesian Q-learning, *in* 'Proceedings of the 15th National Conference on Artificial Intelligence', pp. 761–768.

DeGroot, M. H. (1970), *Optimal Statistical Decisions*, John Wiley and Sons.

Duff, M. (2003), Design for an optimal probe, *in* 'Proceedings of the 20th International Conference on Machine Learning', pp. 131–138.

Duff, M. & Barto, A. (1996), Local bandit approximation for optimal learning problems, *in* M. Mozer, M. Jordan & T. Pesche, eds, 'Advances in Neural Information Processing Systems', Vol. 9, Cambridge, MA: MIT Press, pp. 1019–1025.

Dupacova, J. (1995), 'Multistage Stochastic Programs - The State of the Art and Selected Bibliography', *Kybernetica* **31**, 151–174.

Forrester, A. I. J., Sobester, A. & Keane, A. J. (2008), *Engineering design via surrogate modelling: a practical guide*, John Wiley and Sons.

Frazier, P. I. & Powell, W. B. (2010), 'Paradoxes in Learning and the Marginal Value of Information', *Decision Analysis* **7**(4), 378–403.

Frazier, P. I., Powell, W. B. & Dayanik, S. (2008), 'A Knowledge Gradient Policy for Sequential Information Collection', *SIAM Journal on Control and Optimization* **47**(5), 2410—-2439.

Frazier, P. I., Powell, W. B. & Dayanik, S. (2009), 'The Knowledge-Gradient Policy for Correlated Normal Beliefs', *INFORMS Journal on Computing* **21**(4), 599–613.

Fu, M. C. (2002), 'Optimization for simulation: Theory vs. practice', *INFORMS Journal on Computing* **14**(3), 192–215.

Gelman, A., Carlin, J. B., Stern, H. S. & Rubin, D. B. (2004), 'Bayesian Data Analysis, 2nd ed', *Chapman & Hall, New York* p. 63.

Geman, S., Bienenstock, E. & Doursat, R. (1992), 'Neural networks and the bias/variance dilemma', *Neural computation* **4**(1), 1–58.

Gittins, J. (1979), 'Bandit processes and dynamic allocation indices', *Journal of the Royal Statistical Society. Series B (Methodological)* **41**(2), 148–177.

Gittins, J. (1989), 'Multi-armed Bandit Allocation Indices', *Wiley and Sons: New York*.

Gittins, J. & Jones, D. (1974), A dynamic allocation index for the sequential design of experiments, *in* J. Gani, ed., 'Progress in statistics', North Holland, Amsterdam, pp. 241—-266.

Gittins, J., Glazebrook, K. & Weber, R. R. (2011), *Multi-Armed Bandit Allocation Indices*, John Wiley & Sons, New York.

Glasserman, P. (2004), *Monte Carlo Methods in Financial Engineering*, Springer-Verlag, New York.

Glazebrook, K. (1982), 'On the evaluation of suboptimal strategies for families of alternative bandit processes', *Journal of Applied Probability* **19**(3), 716–722.

Glazebrook, K. & Minty, R. (2009), 'A Generalized Gittins Index for a Class of Multiarmed Bandits with General Resource Requirements', *Mathematics of Operations Research*.

Gramacy, R. B. & Lee, H. K. H. (2011), 'Optimization under unknown constraints', *Valencia discussion paper (to appear)*. Arxiv preprint arXiv:1004.4027.

Gupta, S. S. & Miescke, K. J. (1994), 'Bayesian look ahead one stage sampling allocations for selecting the largest normal mean', *Statistical Papers* **35**, 169–177.

Gupta, S. S. & Miescke, K. J. (1996), 'Bayesian look ahead one-stage sampling allocations for selection of the best population', *Journal of statistical planning and inference* **54**, 229–244.

Hadigheh, A. & Terlaky, T. (2006), 'Sensitivity analysis in linear optimization: Invariant support set intervals', *European Journal of Operational Research* **169**(3), 1158–1175.

Hastie, T., Tibshirani, R. & Friedman, J. (2009), *The elements of statistical learning: data mining, inference and prediction*, Springer, New York.

Hastie, T., Tibshirani, R., Friedman, J. & Franklin, J. (2005), *The elements of statistical learning: data mining, inference and prediction*, Vol. 27, Springer, New York.

Horrace, W., Marchand, J. & Smeeding, T. (2008), 'Ranking inequality: Applications of multivariate subset selection', *Journal of Economic Inequality* **6**(1), 5–32.

Hu, J., Fu, M. & Marcus, S. (2007), 'A model reference adaptive search method for global optimization', *Operations Research* **55**(3), 549–568.

Huang, D., Allen, T. T., Notz, W. I. & Zeng, N. (2006), 'Global Optimization of Stochastic Black-Box Systems via Sequential Kriging Meta-Models', *Journal of Global Optimization* **34**(3), 441–466.

Inoue, K., Chick, S. E. & Chen, C. (1999), 'An empirical evaluation of several methods to select the best system', *ACM Transactions on Modeling and Computer Simulation (TOMACS)* **9**, 381–407.

Jedynak, B., Frazier, P. I. & Sznitman, R. (2011), 'Questions with noise: Bayes optimal policies for entropy loss', *Journal of Applied Probability (to appear)*.

Jones, D., Schonlau, M. & Welch, W. (1998*a*), 'Efficient global optimization of expensive black-box functions', *Journal of Global Optimization* **13**(4), 455—-492.

Jones, D., Schonlau, M. & Welch, W. (1998*b*), 'Efficient global optimization of expensive black-box functions', *Journal of Global Optimization* **13**(4), 455–492.

Kaelbling, L. P. (1993), *Learning in embedded systems*, MIT Press, Cambridge, MA.

Kaminski, P., Bryson Jr, A. & Schmidt, S. (1971), 'Discrete square root filtering: A survey of current techniques', *IEEE Transactions on Automatic Control* **16**(6), 727–736.

Katehakis, M. & Veinott, A. (1987), 'The Multi-Armed Bandit Problem: Decomposition and Computation', *Mathematics of Operations Research* **12**(2), 262–268.

Katz, R. & Ionescu, F. (1977), 'Application of the Free-Wilson Technique to Structurally Related Series of Homologues. Quantitative S truct ure-Activity Relations hip Studies of Narcotic Analgetics', **20**(11), 1413–1419.

Kearns, M. & Singh, S. (2002), 'Near-optimal reinforcement learning in polynomial time.', *Machine Learning* **49**, 209–232.

Kleinberg, R., Niculescu-Mizil, A. & Sharma, Y. (2010), 'Regret bounds for sleeping experts and bandits', *Machine Learning* **80**(2), 245–272.

Kulkarni, V. (1986), 'Shortest paths in networks with exponentially distributed arc lengths', *Networks* **16**, 255–274.

Kushner, H. J. & Yin, G. G. (1997), *Stochastic Approximation Algorithms and Applications*, Springer-Verlag, New York.

Kushner, H. J. & Yin, G. G. (2003), *Stochastic Approximation and Recursive Algorithms and Applications*, Springer.

Lai, T. (1987), 'Adaptive treatment allocation and the multi-armed bandit problem', *The Annals of Statistics* **15**(3), 1091–1114.

Lai, T. L. & Robbins, H. (1985), 'Asymptotically Efficient Adaptive Allocation Rules', *Advances in Applied Mathematics* **6**, 4–22.

Martin, J. (1967), *Bayesian Decision Problems and Markov Chains*, John Wiley and Sons.

Miller, A. (2002), *Subset selection in regression*, CRC Press.

Montgomery, D. C. (2008), *Design and analysis of experiments (7th ed.)*, John Wiley and Sons.

Negoescu, D. M., Frazier, P. I. & Powell, W. B. (2010), 'The Knowledge-Gradient Algorithm for Sequencing Experiments in Drug Discovery', *Informs Journal on Computing* pp. 1–29.

Negoescu, D. M., Frazier, P. I. & Powell, W. B. (2011), 'The Knowledge-Gradient Algorithm for Sequencing Experiments in Drug Discovery', *INFORMS Journal on Computing* **23**(3), 346–363.

Peer, S. & Sharma, D. (2007), 'Finding the shortest path in stochastic networks', *Computers and Mathematics with Applications* **53**, 729–740.

Powell, W. B. (2011), *Approximate Dynamic Programming: Solving the curses of dimensionality*, 2nd. edn, John Wiley & Sons, Hoboken, NJ.

Puterman, M. L. (1994), *Markov Decision Processes*, 1st edn, Johnn Wiley and Sons, Hoboken.

Rasmussen, C. E. & Williams, C. K. I. (2006), 'Gaussian Processes for Machine Learning', *The MIT Press*.

Robbins, H. & Monro, S. (1951), 'A stochastic approximation method', *The Annals of Mathematical Statistics* **22**(3), 400–407.

Roberts, C. P. & Casella, G. (2004), *Monte Carlo Statistical Methods*, Springer-Verlag, New York.

Rubinstein, R. Y. & Kroese, D. P. (2008), *Simulation and the Monte Carlo Method*, Wiley-Interscience, New York.

Ryzhov, I. O. & Powell, W. B. (2009*a*), A Monte Carlo Knowledge Gradient Method for Learning Abatement Potential of Emissions Reduction Technologies, *in* M. Rossetti, R. R. Hill, A. Dunkin & R. G. Ingals, eds, 'Proceedings of the 2009 Winter Simulation Conference', pp. 1492–1502.

Ryzhov, I. O. & Powell, W. B. (2009*b*), A Monte-Carlo Knowledge Gradient Method For Learning Abatement Potential Of Emissions Reduction Technologies, *in* M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin & R. G. Ingalls, eds, 'Proceedings of the 2009 Winter Simulation Conference'.

Ryzhov, I. O. & Powell, W. B. (2009*c*), The knowledge gradient algorithm for online subset selection, *in* 'Proceedings of the 2009 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning', pp. 137–144.

Ryzhov, I. O. & Powell, W. B. (2011*a*), 'Information collection for linear programs with unknown objective coefficients', *Submitted for publication*.

Ryzhov, I. O. & Powell, W. B. (2011*b*), 'Information Collection on a Graph', *Operations Research* **59**(1), 188–201.

Ryzhov, I. O. & Powell, W. B. (2011*c*), The value of information in multi-armed bandits with exponentially distributed rewards, *in* 'Proceedings of the 2011 International Conference on Computational Science', pp. 1363–1372.

Ryzhov, I. O., Powell, W. B. & Frazier, P. I. (2011), 'The knowledge gradient algorithm for a general class of online learning problems', *Operations Research (to appear)*.

Ryzhov, I. O., Valdez-Vivas, M. R. & Powell, W. B. (2010), Optimal Learning of Transition Probabilities in the Two-Agent Newsvendor Problem, *in* B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan & E. Yücesan, eds, 'Proceedings of the 2010 Winter Simulation Conference', pp. 1088–1098.

Sacks, J., Welch, W., Mitchell, T. J. & Wynn, H. P. (1989), 'Design and analysis of computer experiments', *Statistical Science* **4**(4), 409–423.

Satia, J. & Lave, R. (1973), 'Markov decision processes with imprecise transition probabilities', *Operations Research* **21**(3), 755–763.

Scott, W., Frazier, P. I. & Powell, W. B. (2011), 'The Correlated Knowledge Gradient for Simulation Optimization of Continuous Parameters using Gaussian Process Regression', *SIAM J. Control Optim.*

Scott, W. R., Powell, W. B. & Simão, H. P. (2010), Calibrating simulation models using the knowledge gradient with continuous parameters, *in* B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan & E. Yücesan, eds, 'Proceedings of the 2010 Winter Simulation Conference', pp. 1099–1109.

Settles, B. (2009), Active learning literature survey, Computer Sciences Technical Report 1648, University of Wisconsin–Madison.

Si, J., Barto, A. G., Powell, W. B. & Wunsch, D. (2005), *Learning and Approximate Dynamic Programming*.

Silver, E. (1963), Markovian decision processes with uncertain transition probabilities or rewards, Technical Report 1, Operations Research Center, MIT.

Singh, S. P., Jaakkola, T., Szepesvari, C. & Littman, M. (2000), 'Convergence results for single-step on-policy reinforcement-learning algorithms', *Machine Learning* **38**(3), 287—-308.

Snyder, T. & Steele, J. (1995), Probabilistic networks and network algorithms, *in* M. Ball, T. Magnanti & C. Monma, eds, 'Handbooks of Operations Research and Management Science, vol. 7: Networks', North-Holland Publishing, Amsterdam, pp. 401–424.

Spall, J. C. (2003), *Introduction to Stochastic Search and Optimization: Estimation, Simulation and Control*, John Wiley & Sons, Hoboken, NJ.

Srinivas, N., Krause, A., Kakade, S. M. & Seeger, M. (2010), Gaussian process optimization in the bandit setting: No regret and experimental design, *in* 'Proceedings of the 27th International Conference on Machine Learning', pp. 1015–1022.

Steele, J. M. (2000), *Stochastic Calculus and Financial Applications*, Springer, New York.

Stein, M. (1999), *Interpolation of Spatial Data: Some theory for kriging*, Springer Verlag.

Streeter, M. & Smith, S. (2006), A simple distribution-free approach to the max k-armed bandit problem, *in* 'Principles and Practice of Constraint Programming', Vol. 4204 of *Lecture Notes in Computer Science*, pp. 560–574.

Sutton, R., Maei, H., Precup, D., Bhatnagar, S., Silver, D., Szepesvári, C. & Wiewiora, E. (2009), Fast gradient-descent methods for temporal-difference learning with linear function approximation, *in* 'Proceedings of the 26th International Conference on Machine Learning', pp. 993–1000.

Sutton, R. S. & Barto, A. G. (1998), *Reinforcement Learning*, Vol. 35, MIT Press, Cambridge, MA.

Takimoto, E. & Warmuth, M. K. (2003), 'Path kernels and multiplicative updates', *Journal of Machine Learning Research* **4**, 773–818.

Tesauro, G. (1992), 'Practical Issues in Temporal Difference Learning', *Machine Learning* **8**(3-4), 257–277.

Vanderbei, R. (1980), 'The optimal choice of a subset of a population', *Mathematics of Operations Research* **5**(4), 481–486.

Vanderbei, R. J. (2008), *Linear programming: foundations and extensions (3rd ed.)*, Springer.

Vermorel, J. & Mohri, M. (2005), 'Multi-armed bandit algorithms and empirical evaluation', *Proceedings of the 16th European Conference on Machine Learning* pp. 437–448.

Wald, A. & Wolfowitz, J. (1948), 'Optimum Character of the Sequential Probability Ratio Test', *The Annals of Mathematical Statistics* **19**, 326–339.

Weibull, J. W., Mattsson, L.-G. & Voorneveld, M. (2007), 'Better May be Worse: Some Monotonicity Results and Paradoxes in Discrete Choice Under Uncertainty', *Theory and Decision* **63**(2), 121–151.

Wetherill, G. B. & Glazebrook, K. D. (1986), *Sequential methods in statistics*, Chapman & Hall.