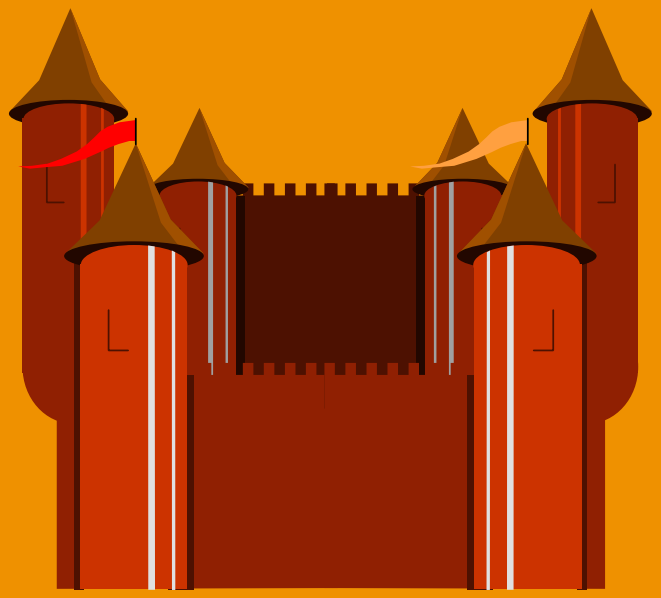
ORF 544

Stochastic Optimization and Learning

Spring, 2019



Warren Powell
Princeton University
http://www.castlelab.princeton.edu

Week 3

Chapter 4: Introduction to stochastic optimization

Chapter 5: Derivative-based stochastic optimization

Chapter 6: Stepsizes

Overview

Classes of algorithmic strategies

Classes of algorithmic strategies

- Exact methods
  - » If we can compute the expectation in

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

    … then this is a deterministic problem.
- Sampled problems
  - » If the only problem is that we cannot compute the expectation, we might solve a sampled approximation

$$\max_x \hat{\mathbb{E}}F(x,W)$$

- Adaptive learning algorithms
  - » This is what we have to turn to for most problems, and is the focus of this tutorial.

Classes of algorithmic strategies

- Exact solution methods
  - » Consider our classical stochastic optimization problem:

$$\max_x \mathbb{E}_W F(x, W)$$

  - » If we can compute the expectation, then we have a function $f(x) = \mathbb{E}_W F(x, W)$, which means we can write our optimization problem as

$$\max_x f(x)$$

  - » In other words, if we can compute the expectation, then we have a deterministic optimization problem.
  - » This means that "stochastic optimization" is, fundamentally, related to computation.

Classes of algorithmic strategies

- A stochastic shortest path problem
  - » Assume that the cost of traversing from i to j is a random variable given by $\hat{c}_{ij}$.
  - » Let x_{ij} be the flow from i to j. We wish to solve the linear optimization problem

$$\min_x \mathbb{E} \sum_{i,j} \hat{c}_{ij} x_{ij}$$

  - » Subject to flow conservation constraints. Assume that we have to choose x_{ij} before we get to see the costs $\hat{c}_{ij}$. This means we can write

$$\min_x \mathbb{E} \sum_{i,j} \hat{c}_{ij} x_{ij} = \sum_{i,j} x_{ij} \mathbb{E}\hat{c}_{ij} = \sum_{i,j} x_{ij} \bar{c}_{ij}$$

Classes of algorithmic strategies

- Exact solution methods

4.2.2 A newsvendor problem with known distribution

We next consider one of the oldest stochastic optimization problems, known as the newsvendor problem, which is given by

$$\max_x EF(x,W) = \mathbb{E}\big(p \min x, W - cx\big). \tag{4.5}$$

Assume that we know the cumulative distribution $F^W(w) = \mathbb{P}[W \leq w]$ of the demand W. We begin by computing the stochastic gradient, given by

$$\nabla_x F(x,W) = \begin{cases} p-c & \text{If } x \leq W \\ -c & \text{If } x > W. \end{cases} \tag{4.6}$$

We next observe that if $x = x^*$, the optimal solution, then the expectation of the gradient should be zero. This means

$$\begin{aligned} \mathbb{E}\nabla_x F(x,W) &= (p-c)\mathbb{P}[x* \leq W] - c\mathbb{P}[x* > W], \\ &= (p-c)\mathbb{P}[x* \leq W] - c(1 - \mathbb{P}[x* \leq W]), \\ &= 0. \end{aligned}$$

Solving for $\mathbb{P}[x^* \leq W]$ gives

$$\mathbb{P}[x^* \leq W] = \frac{c}{p}. \tag{4.7}$$

Classes of algorithmic strategies

- Exact solution methods

4.2.4 Optimal control

In section 2.13, we formulated an optimal control problem of the form

$$\min_{u_0,\ldots,u_T} \sum_{t=0}^{T} L_t(x_t, u_t).$$

where states evolve according to $x_{t+1} = f(x_t, u_t)$. We may introduce a stochastic noise term giving us the state transition equation

$$x_{t+1} = f(x_t, u_t) + w_t,$$

where (following the standard convention of the controls community) w_t is random at time t. The historical basis for this notational convention is the roots of optimal control in

Classes of algorithmic strategies

- Exact solution methods

continuous time, where w_t would represent the noise between t and $t+dt$. In the presence of noise, we need to introduce a policy $U^\pi(x_t)$. We would now write our objective function as

$$\min_\pi \mathbb{E} \sum_{t=0}^{T} L_t(x_t, U_t^\pi(x_t)). \tag{4.11}$$

Now assume that the loss function has the quadratic form

$$L_t(x_t, u_t) = (x_t)^T Q_t x_t + (u_t)^T R_t u_t.$$

After quite a bit of algebra, it is possible to show that the optimal policy has the form

$$U_t^\pi(x_t) = K_t x_t, \tag{4.12}$$

where K_t is a complex matrix that depends on the matrices Q_t and R_t.

This solution depends on three critical features of this problem:

- The objective function is quadratic in the state x_t and the control u_t.
- The control u_t is unconstrained.
- The noise term w_t is additive in the transition function.

Classes of algorithmic strategies

- Exact solution methods
  - » Bellman's optimality equation

$$V_t(S_t) = \min_{a_t \in \mathcal{A}} \left(C(S_t, a_t) + \gamma \mathbb{E}\left\{ V_{t+1}(S_{t+1}) \mid S_t \right\} \right)$$

$$= \min_{a_t \in \mathcal{A}} \left(C(S_t, a_t) + \gamma \sum_{s'} p(S_{t+1} = s' \mid S_t, a_t) V_{t+1}(S_{t+1}) \right)$$

  - » If we can compute the one step transition matrix, then this is a deterministic problem.
  - » We will get to this later in the course.

Sampled models

Classes of algorithmic strategies

- Sampled models
  - » When we cannot compute the expectation, we can sometimes get a good solution by replacing the full set of possible outcomes with a sample.
  - » That is, we replace

$$\max_x \mathbb{E}F\left(x,W\right)$$

  - » with

$$\max_x \frac{1}{N}\sum_{n=1}^{N} F\left(x,W^n\right)$$

  - » This can be used even when x and W are high dimensional.

Classes of algorithmic strategies

- Sampled models

4.3.1.3 Sampled parametric models Sampled models may take other forms. Imagine that we wish to model demand as a function of price using a logistic function

$$D(p|\theta) = D^0 \frac{e^{\theta_0 - \theta_1 p}}{1 + e^{\theta_0 - \theta_1 p}}.$$

We want to pick a price that maximizes revenue using

$$R(p|\theta) = pD(p|\theta).$$

Our problem is that we do not know θ. We might assume that our vector θ follows a multivariate normal distribution, in which case we would want to solve

$$\max_p \mathbb{E}_\theta pD(p|\theta), \tag{4.16}$$

but computing the expectation may be hard. However, perhaps we are willing to say that θ may take on one of a set of values $\theta_1, \ldots, \theta_K$, each with probability q_k. Now we can solve

$$\max_p \sum_{k=1}^{K} pD(p|\theta_k)q_k. \tag{4.17}$$

Whereas equation (4.16) may be intractable, (4.17) may be much easier.

Classes of algorithmic strategies

- Notes:
  - » There are very few problems where the expectation can be computed exactly.
  - » Sampling is a very powerful strategy, one that we will use. But is just a way of approximating an expectation.
  - » Most problems where some form of iterative learning, which is the focus of the rest of the course.

Sampled approximations

- Canonical stochastic optimization problem
  - » Remember that the expectation operator $\mathbb{E}$ should always be interpreted as "taking an average." It is rare that we can do this exactly. Most of the time, we will be taking an average.
  - » Assume we have a sample of values of W that we write as

$$W^1, W^2, \ldots, W^N$$

  - » Then we can approximate our expectation by

$$\mathbb{E}_W \{F(x,W) \mid S_0\} \approx \frac{1}{N} \sum_{n=1}^{N} F(x,W^n)$$

Intro to stochastic optimization

4.3.3 Convergence

The first question that arises with sampled models concerns how large N needs to be. Fortunately, the sample average approximation enjoys some nice convergence properties. We start by defining

$$\begin{aligned} F(x) &= \mathbb{E}F(x,W), \\ \overline{F}^N(x) &= \frac{1}{N}\sum_{n=1}^{N} F(x,W^n). \end{aligned}$$

The simplest (and most intuitive) result is that we get closer to the optimal solution as the sample size grows. We write this by saying

$$\lim_{N\to\infty} \overline{F}^N(x) \to \mathbb{E}F(x,W).$$

Let x^N be the optimal solution of the approximate function, which is to say

$$x^N = \arg\max_{x\in\mathcal{X}} \overline{F}^N(x).$$

The asymptotic convergence means that we will eventually achieve the optimum solution, a result we state by writing

$$\lim_{N\to\infty} \overline{F}^N(x^N) \to F(x^*).$$

Intro to stochastic optimization

- Convergence of sampled approximations
  - » The optimal solution of a sampled problem converges exponentially to the optimal objective function:

$$\mathbb{P}[F(x^N) < F(x^*) - \epsilon] < |\mathcal{X}|e^{-\eta N}, \tag{4.27}$$

for some constant $\eta > 0$. What equation (4.27) is saying is that the probability that the quality of our estimated solution x^N, given by $F(x^N)$, is more than ϵ away from the optimal $F(x^*)$, decreases at an exponential rate $e^{-\eta N}$. The coefficient $\mathcal{X}$ is quite large, of course, and we have no idea of the magnitude of η. However, the result suggests that the probability that we underperform $F(x^*) - \epsilon$ declines exponentially with the same size N, which is comforting.

A similar but stronger result is available when x is continuous and $f(x, W)$ is convex, and the feasible region $\mathcal{X}$ might be specified by a set of linear inequalities. In this case, the convergence is given by

$$\mathbb{P}[F(x^N) < F(x^*) - \epsilon] < Ce^{-\eta N}, \tag{4.28}$$

for given constants $C > 0$ and $\eta > 0$. Note that (4.28) does not depend on the size of the feasible region, as is the case in (4.27).

Creating a sampled model

A particularly important problem with large-scale applications is the design of the sample $W^1, \ldots, W^N$. The most popular methods for generating a sample are:

- From history - We may not have a formal probability model for W, but we can draw samples from history. For example, W^n might be a sample of wind speeds over a week, or currency fluctuations over a year.
- Monte Carlo simulation - There is a powerful set of tools on the computer known as Monte Carlo simulation which allow us to create samples of random variables as long as we know the underlying distribution (we cover this in more detail in chapter 10).

State-independent vs. state-dependent

State dependency

- State independent problems
  - » The *problem* does not depend on the state of the system.

$$\max_x \mathbb{E}F(x,W) = \mathbb{E}\left\{p\min(x,W) - cx\right\}$$

  - » The only state variable is what we know (or believe) about the unknown function $\mathbb{E}F(x,W)$, called the belief state B_t, so $S_t = B_t$.
  - » State independent problems can also be called *pure learning problems.*

State dependency

- Examples of pure learning problems:
  - » What is the best path to your new job?
  - » What is the best lineup of five players for a basketball team?
  - » What is the best concentration of nanoparticles to achieve the best release profile?
  - » What is the best bid for an ad to maximize clicks?
  - » What are the best buy-low, sell-high prices for maximizing profits for buying and selling energy for grid storage?

State dependency

- State-dependent problems
  - » Now the *problem* may depend on what we know at time *t*:

$$\max_{0 \le x \le R_t} \mathbb{E}\, C(S, x, W) = \mathbb{E}\left\{ p_t \min(x, W) - cx \right\}$$

  - » Now the state is $S_t = (R_t, p_t, B_t)$

State dependency

- Notes:
  - » The distinguishing characteristic of this class of functions is that the *function* does not depend on the state S_t (other than S_0).
  - » Examples:
    - The conductivity of a material constructed using design x which depends on temperatures, concentrations, catalysts, …).
    - The choice of drug cocktail for someone with cancer.
    - The sales of a laptop with a set of features x (or x could just be price).
    - The revenue generated from a battery storage device which is buying and selling to/from the grid, using a policy determined by x.

Solution strategies and problem classes

- Examples of state-dependent problems:
  - » Virtually any resource allocation problem:
    - Managing inventories, machines, people, equipment…
    - Amazon logistics problems
    - Managing drones, sensors,
    - Asset selling, portfolio optimization
  - » "Contextual" learning problems
    - What is the best path *given the weather (or weather forecast)*
    - What is the best basketball team *given the opponent.*
    - What is the best bid for an ad to maximize clicks *given the attributes of the customer logged in.*
    - What are the best buy-low, sell-high prices for maximizing profits for buying and selling energy for grid storage *given forecasts of future temperatures.*

Solution strategies and problem classes

- Why do we make the distinction between state-independent and state-dependent?
  - » After all, state-independent is just a special case of state-dependent.
  - » The reason is that state-independent (learning) problems are much simpler, and they arise in many settings (including state-dependent problems).

Solution strategies and problem classes

- Ranking and selection (derivative free)
  - » Basic problem:

$$\max_{x\in\{x_1,\ldots,x_M\}} \mathbb{E}\{F(x,W)\,|\,S^0\}$$

  - » We need to design a policy $X^\pi(S^n)$ that finds a design given by $x^{\pi,N}$

$$\max_\pi \mathbb{E}\{F\left(x^{\pi,N},W\right)|\,S^0\}$$

  - » We refer to this objective as maximizing the *final reward.*

Modeling uncertainty

Offline vs. online learning

- Modeling uncertainty
  - » Model uncertainty S^0
    - Uncertainty in model parameters
    - Uncertainty in the model itself
  - » Learning uncertainty $W^1, W^2, \ldots, W^N$
    - This is the variability in the choice of the final design $x^{\pi,N}$
  - » Implementation uncertainty $\hat{W}$
    - This is the new information that arrives after we have picked our final design

Offline vs. online learning

- Illustration
  - » Use newsvendor problem with:
    - Uncertainty in the mean of the demand distribution.
    - Randomness in the sampled demands that you use when learning the best order quantity.
    - Randomness in the demand you use to test your solution.

Derivative-based stochastic search

Model uncertainty

We can illustrate uncertainty in a parameter by assuming that we do not know the mean demand μ. However, we may feel that we can describe its likely values using an exponential distribution given by

$$\mu \sim \lambda e^{-\lambda u},$$

where the parameter λ is known as a hyperparameter, which is to say it is a parameter that determines a distribution that describes the uncertainty of a problem parameter. The assumption is that even if we do not know λ precisely, it still does a good job of describing the uncertainty in the mean demand μ.

Now we have two random variables: the random demand $\hat{D}$, and the uncertain mean μ. In this case, we could write $W = (\mu, \hat{D})$. A better way would be to put the uncertainty around μ in the initial state S_0 (here it would be called a prior). In this case, we would write our problem as

$$\begin{aligned} F(x) &= \mathbb{E}\{F(x,W)|S_0\}, \\ &= \mathbb{E}_{S_0}\{\mathbb{E}_{W|S_0}F(x,W)|S_0\}. \end{aligned}$$

For our example, this would be translated as

$$F(x|\lambda) = \mathbb{E}_{\mu}\{\mathbb{E}_{\hat{D}|\mu}F(x,\hat{D})|\mu\}.$$

The notation $\mathbb{E}_{\hat{D}|\mu}$ means the conditional expectation given μ. Using our distributions, we would write this as

$$F(x|\lambda) = \int_{u=0}^{\infty} \lambda e^{-\lambda u} \sum_{d=0}^{\infty} \frac{u^d e^{-u}}{d!} \big(p \min(x,d) - cx\big).$$

Derivative-based stochastic search

5.2.3 Learning uncertainty

Finally, consider an adaptive algorithm (which we first introduced in chapter 4) that proceeds by guessing x^n and then observing W^{n+1} which leads to x^{n+1} and so on (we give examples of these procedures in this chapter). If we limit the algorithm to N iterations, our sequence will look like

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

Below, we are going to describe the rule that produces each x^n as a policy π, so it can help to write each decision as $x^{\pi,n}$, leading to the final solution $x^{\pi,N}$.

Clearly, the final solution $x^{\pi,N}$ (as well as all the intermediate solutions) depends on the sequence $W^1, \ldots, W^N$. That means that $x^{\pi,N}$ is a random variable that depends on the sequence of observations we make while executing the algorithm. In fact, it may easily be the case that an outcome W^n depends on the previous decision x^{n-1} (or even the entire history). For this reason, we are going to designate an outcome of the sequence $(W^1, \ldots, W^n, \ldots, W^N)$ by ω^π, which indicates that the sample path may depend on the policy. We then assume $\omega^\pi \in \Omega^\pi$, where Ω^π is the set of all possible sample paths.

When we finally obtain our solution $x^{\pi,N}$, we then have to evaluate the quality of the solution. For the moment, let's fix $x^{\pi,N}$. Now the only uncertainty is the random variable W when we go to implement $x^{\pi,N}$. However, now we are using W to test the performance of $x^{\pi,N}$. It is useful to distinguish the Ws that we use to find $x^{\pi,N}$, and the W we use to test it. For this reason, we will sometimes use $\widehat{W}$ as our "testing" W, while the sequence $W^1, \ldots, W^N$ is our "training" Ws.

Derivative-based stochastic search

- Learning uncertainty

We typically evaluate a policy by simulating a sequence of $W^1, \ldots, W^N$. Let $\hat{\Omega}$ be a set of possible samples (perhaps 20 of them) and let ω be one of the elements of the sample (we might think of ω as being a number from 1 to 20). Then $W^n(\omega)$ would be the actual realization of W^n for the nth sample, as illustrated in table 5.1.

Assume that we have a set of outcomes of W that we call $\hat{\Omega}$, where $\omega \in \hat{\Omega}$ is one outcome of W which we represent using $W(\omega)$. Once again assume that we have taken a random sample to create $\hat{\Omega}$ where every outcome is equally likely. Then we could evaluate our solution $x^{\pi,N}$ using

$$\overline{F}(x^{\pi,N}) = \frac{1}{|\hat{\Omega}|} \sum_{\omega \in \hat{\Omega}} F(x^{\pi,N}, W(\omega)).$$

The estimate $\overline{F}(x^{\pi,N})$ evaluates a single decision $x^{\pi,N}$, but not the policy that produced the decision. What we are really interested in is the performance of the policy, which we designate F^π. To do this, we might write

$$F^\pi = \mathbb{E}\{F(x^{\pi,N}, \widehat{W}) | S_0\}. \tag{5.7}$$

Derivative-based stochastic search

- Implementation uncertainty
  - » We can have (and can use) a probability distribution for W:

5.2.1 Implementation uncertainty W

We illustrate implementation uncertainty using our newsvendor problem, where we make a decision x, then observe a random demand $W = \hat{D}$, after which we calculate our profit using equation (5.3). Imagine that our demand follows a Poisson distribution given by

$$\mathbb{P}[\hat{D} = d] = \frac{\mu^d e^{-\mu}}{d!},$$

where $d = 0, 1, 2, \ldots$. In this setting, our expectation would be over the possible outcomes of $\hat{D}$, so we could write (5.3) as

$$F(x|\mu) = \sum_{d=0}^{\infty} \frac{\mu^d e^{-\mu}}{d!} \big(p \min\{x, d\} - cx\big).$$

- Computing expectations exactly is actually quite rare.

Derivative-based stochastic search

- Implementation uncertainty
  - » More often we are working with a sample, that might come from:
    - A probability distribution (e.g. a multivariate normal)
    - History/field observations

ω	$\hat{D}^1$	$\hat{D}^2$	$\hat{D}^3$	$\hat{D}^4$	$\hat{D}^5$	$\hat{D}^6$	$\hat{D}^7$	$\hat{D}^8$	$\hat{D}^9$	$\hat{D}^{10}$
1	0	1	6	3	6	1	6	0	2	4
2	3	2	2	1	7	5	4	6	5	4
3	5	2	3	2	3	4	2	7	7	5
4	6	3	7	3	2	3	4	7	3	4
5	3	1	4	5	2	4	3	4	3	1
6	3	4	4	3	3	3	2	2	6	1

Table 5.1 Illustration of six sample paths for the random variable $\hat{D}$.

Derivative-based stochastic search

- Implementation uncertainty
  - » We can now replace our expectation E with one based on a sampled representation

We may approximate our newsvendor problem by generating a sample of possible demands $\hat{D}$ and storing it in our set $\hat{\Omega}$. Thus, we might generate

$$\hat{\Omega} = \{1, 6, 5, 8, 2, 5, 4, 4, 3, 7\}.$$

Thus, for $\omega = 3$, $\hat{D}(\omega) = 5$.

Using this set, and assuming that each outcome is equally likely, we would write our estimate of the expectation as

$$\overline{F}(x) = \frac{1}{|\hat{\Omega}|} \sum_{\omega \in \hat{\Omega}} p \min\{x, \hat{D}(\omega)\} - cx.$$

Derivative-based stochastic search

The challenge is now to parse this down to something we can compute. To do this, we have to break down all the sources of uncertainty. It is instructional to unroll this in reverse as follows:

Evaluating $x^{\pi,N}$ - Given $x^{\pi,N}$, we have to simulate $\widehat{W}$ to find $F(x^{\pi,N},\widehat{W})$.

Finding $x^{\pi,N}$ - Here we have to simulate the sequence $W^1,\ldots,W^N$ while following policy π. In some cases the policy itself is random (for example, the policy might be to choose an action x with probability $\pi(x|\theta)$).

Sampling each W^n - While following policy π, an outcome W^n depends on our model specified in S_0, which may include distributional information about a parameter (for example).

The state S_0 - If our initial state includes distributional information about a parameter, we have to sample over this distribution so that we can simulate W^n.

Recognizing this sequence, we can now rewrite our expectation in (5.7) using

$$F^{\pi} = \mathbb{E}_{S_0}E_{W^1,\ldots,W^N|S_0}E_{x^{\pi,N}|W^1,\ldots,W^N}E_{\widehat{W}|x^{\pi,N}}F(x^{\pi,N},\widehat{W}).$$

In practice, we can replace each expectation by a sample over whatever is random. Furthermore, these samples can be a) sampled from a probability distribution, b) represented by a large, batch dataset, or c) observed from an exogenous process (which involves online learning).

Derivative-based stochastic search

- Evaluating a policy

$$F^{\pi} = \mathbb{E}_{S^0}\mathbb{E}_{W^1,\ldots,W^N|S^0}\mathbb{E}_{\hat{W}|S^0}F(x^{\pi,N},\hat{W})$$

» Note that $x^{\pi,N}$ is a random variable, but it is a deterministic function of $W^1,\ldots,W^N$ and the policy π.

Derivative-based stochastic search

- Evaluating a policy

$$F^{\pi} = \mathbb{E}_{S^0}\mathbb{E}_{W^1,\ldots,W^N|S^0}\mathbb{E}_{\hat{W}|S^0}F(x^{\pi,N},\hat{W})$$

Sample initial beliefs about parameters

Following policy π, simulate W's from sampled parameters in initial state

Finally, given $x^{\pi,N}$, we have to simulate how well the design works when it is implemented.

» Note that $x^{\pi,N}$ is a random variable, but it is a deterministic function of $W^1, \ldots, W^N$ and the policy π.

Derivative-based stochastic optimization

Derivative-based stochastic optimization

- Canonical stochastic optimization problem
  - » Basic problem:

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

  - » Decisions
    - Manufacturing network (x=design)
    - Unit commitment problem (x=day ahead decisions)
    - Inventory system (x=design, replenishment policy)
    - Battery system (x=choice of material)
    - Patient treatment cost (x=drug, treatments)
  - » A "decision" may be a policy.
    - Buy-sell signals for sell a stock or charging a battery

Derivative-based stochastic optimization

- Canonical stochastic optimization problem
  - » Basic problem:

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

  - » "Exogenous information" W:
    - Manufacturing network (W=performance)
    - Unit commitment problem (W=weather)
    - Inventory system (W=Demands)
    - Battery system (W=Lifetime, energy density)
    - Patient treatment cost (W=response to treatment)
    - Trucking company (W=profits, service)
  - » W may be a single realization, or an entire sequence.
    - x may be a buy-sell policy, and W may then be an entire sequence of prices $p_1, \ldots, p_T$

Derivative-based stochastic optimization

- Canonical stochastic optimization problem
  - » I prefer to write this as

$$\max_x \mathbb{E}\{F(x,W) \mid S_0\}$$

  - » The initial state S_0 carries:
    - Any deterministic parameters
      - Arrival rate of λ customers
    - Initial values of parameters that will vary over time:
      - Prices
      - Inventories
    - Beliefs about uncertain parameters
      - Beliefs about the arrival rate λ.
      - Response μ_x of a patient to a drug x.

Derivative-based stochastic optimization

- Writing expectations
  - » It can be useful to express the explicit random variables over which an expectation is being take.
  - » We might be finding the best drug x to maximize the reduction in blood sugar W_x which depends on the patience response μ_x:

$$F(x,W) = W_x \quad = \mu_x + \varepsilon^n$$

    which is itself unknown but has an assumed distribution.
  - » We would write our objective function as

$$\max_x \mathbb{E}_\mu \mathbb{E}_{W|\mu} F(x,W)$$

  - » We will often write $\mathbb{E}F(x,W)$, but it is a useful exercise to expand the expectation to express the random variables that the expectation is being taken over.

Derivative-based stochastic optimization

- Deterministic search (derivative based)
  - » Basic problem:

$$\max_x f(x)$$

  - » Steepest ascent

$$x^{n+1} = x^n + \alpha_n \nabla_x f(x^n)$$

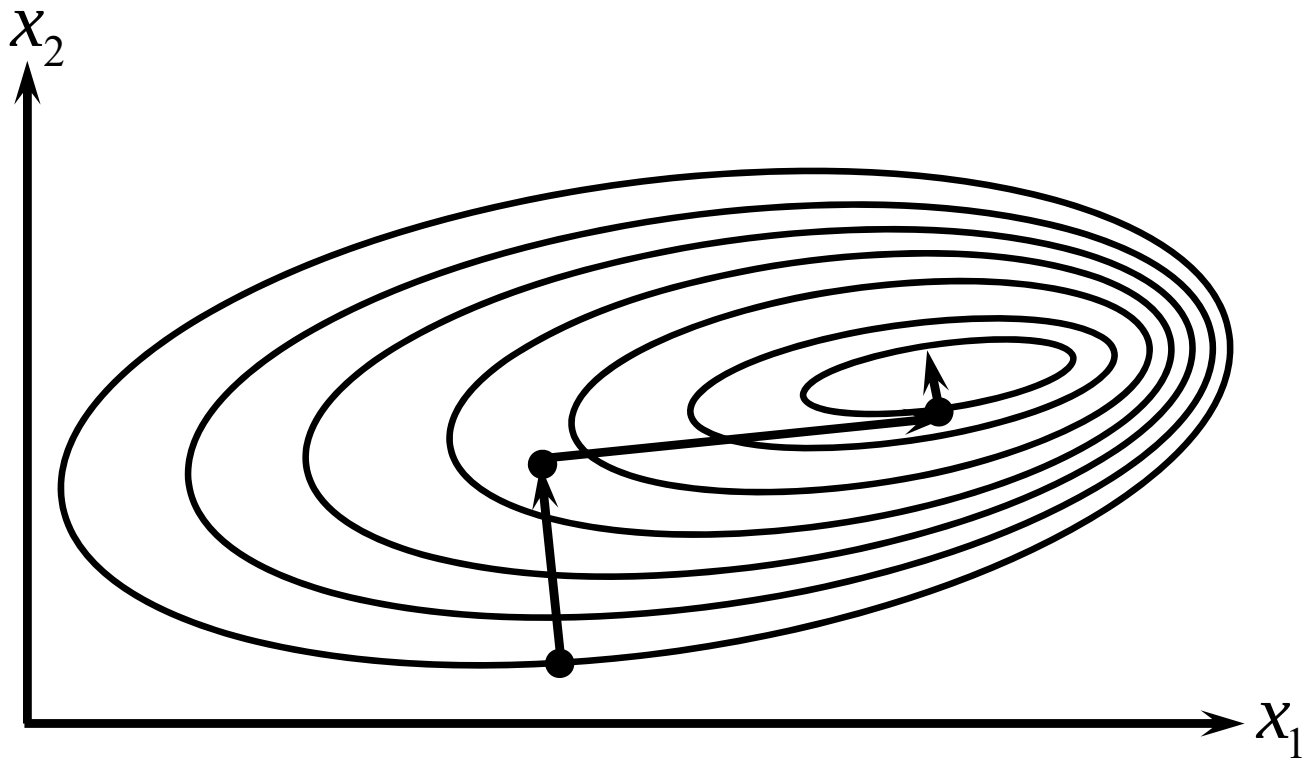
  - » Finding the stepsize:

$$\max_{\alpha \geq 0} f(x^n + \alpha \nabla f(x^n))$$

  - » We can use various algorithms for one-dimensional search.

Derivative-based stochastic optimization

- Classic steepest ascent



$$x^{n+1} = x^n + \alpha^n \nabla f(x^n)$$

45

Derivative-based stochastic optimization

- Stochastic search (derivative based)
  - » Basic problem:

$$\max_x \mathbb{E}\{F(x,W) \mid S_0\}$$

  - » Stochastic gradient

$$x^{n+1} = x^n + \alpha_n \nabla_x F(x^n, W^{n+1})$$

  - » But how do we find the stepsize?
  - » We can no longer do a one-dimensional search.

Derivative-based stochastic optimization

- Indexing notation
  - » Any variable indexed by n is a function of $W^1, \ldots, W^n$.
  - » Similarly if we are using time t.
  - » This means writing our updating equation as

$$x^{n+1} = x^n + \alpha_n \nabla F(x^n, W^{n+1})$$

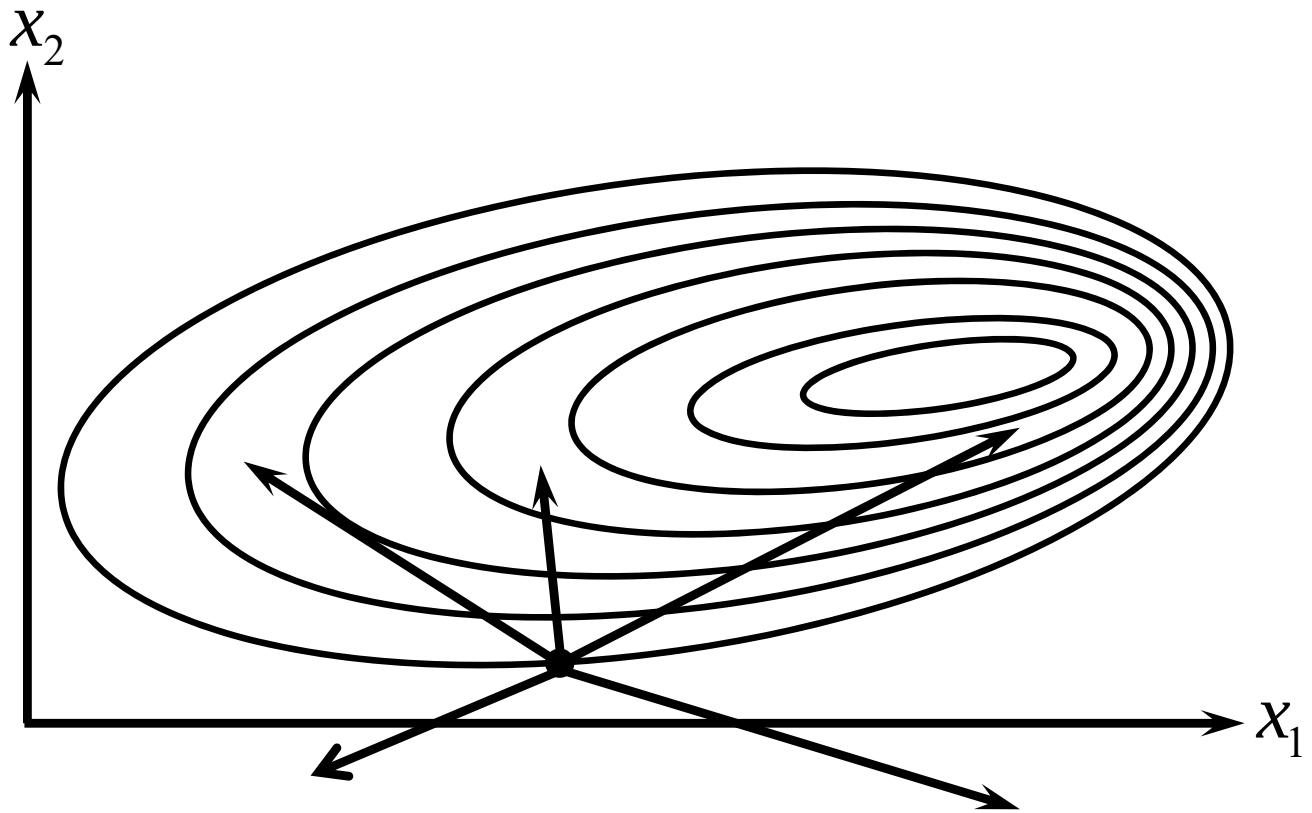
  - » This tells us that α_n may be stochastic (depends on information), but cannot depend on W^{n+1}.

Derivative-based stochastic optimization

- Notes:
  - » If the objective function is easy to compute, one-dimensional searches are relatively easy.
  - » If we know a maximum value for the stepsize α, the Fibonacci search method is optimal for derivative-free one-dimensional search.
  - » Golden section search is an easy variant of the Fibonacci.
  - » … but these methods do not work for stochastic problems.

Derivative-based stochastic optimization

- Stochastic gradients
  - » A stochastic gradient $\nabla_x F(x, W)$ may point in any direction, including the wrong direction



49

Derivative-based stochastic optimization

- Consider our newsvendor problem

$$\max_x F(x) = \mathbb{E}F(x,W) = \mathbb{E}\big(p\min\{x,W\} - cx\big). \tag{5.3}$$

We can use the fact that we can compute *stochastic gradients*, which are gradients that we compute only after we observe the demand W, given by

$$\nabla_x F(x,D) = \begin{cases} p-c & \text{If } x \le W \\ -c & \text{If } x > W. \end{cases} \tag{5.4}$$

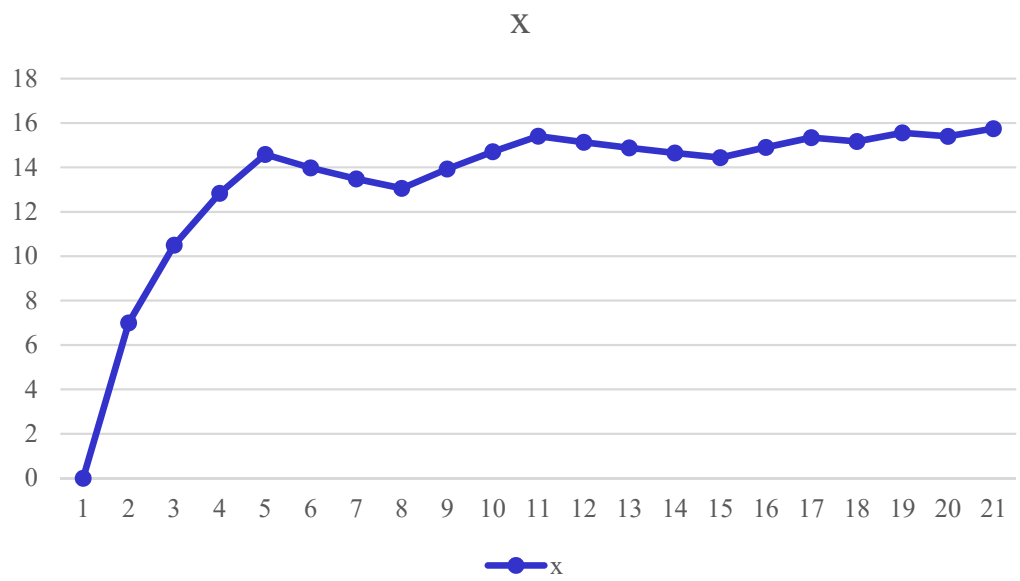
We are going to show how to design simple algorithms that exploit our ability to compute gradients after the random information becomes known. Even when we do not have direct access to gradients, we may be able to estimate them using finite differences. We are also going to see that the core ideas of stochastic gradient methods pervade a wide range of adaptive learning algorithms.

» Assume:
- $p = 10$
- $c = 3$
- W is uniform from 10 to 20.

Derivative-based stochastic optimization

- Numerical example
  - » P=10, c=3, $\alpha = \frac{1}{1+n}$

Iteration	x	alpha	W	Gradient
0	0	1.000	13.09	7
1	7	0.500	15.21	7
2	10.5	0.333	17.83	7
3	12.83333	0.250	12.68	-3
4	12.08333	0.200	11.23	-3
5	11.48333	0.167	15.47	7
6	12.65	0.143	15.33	7
7	13.65	0.125	13.27	-3
8	13.275	0.111	18.72	7
9	14.05278	0.100	12.86	-3
10	13.75278	0.091	14.60	7
11	14.38914	0.083	15.66	7
12	14.97247	0.077	18.95	7
13	15.51094	0.071	13.02	-3
14	15.29665	0.067	19.44	7
15	15.76332	0.063	11.98	-3
16	15.57582	0.059	15.29	-3
17	15.39935	0.056	18.20	7
18	15.78824	0.053	16.14	7
19	16.15666	0.050	18.38	7
20	16.50666	0.048	12.80	-3



Derivative-based stochastic optimization

- Finding the mean
  - » We can write the problem of finding the mean μ of a random variable W as the stochastic optimization problem

$$\min_{\mu} \mathbb{E}\frac{1}{2}(\mu - W)^2$$

  - » Here, our function is $F(\mu, W) = \frac{1}{2}(\mu - W)^2$. The stochastic gradient is $\nabla_\mu F(\mu, W) = (\mu - W)$. The stochastic gradient algorithm is

$$\begin{aligned}\mu^{n+1} &= \mu^n - \alpha_n \nabla_x F(\mu^n, W^{n+1}) \\ &= \mu^n - \alpha_n(\mu^n - W^{n+1}) \\ &= (1 - \alpha_n)\mu^n + \alpha_n W^{n+1}\end{aligned}$$

Derivative-based stochastic optimization

- Finding the mean
  - » Now consider

$$\mu^{n+1} = (1-\alpha_n)\mu^n + \alpha_n W^{n+1}$$

  - » For a stepsize, try $\alpha_n = \frac{1}{n+1}$.
  - » $\mu^1 = (1-1)\mu^0 + 1\, W^1 = W^1$

$$\mu^2 = \left(1-\frac{1}{2}\right)\mu^1 + \frac{1}{2}W^2 = \frac{1}{2}W^1 + \frac{1}{2}W^2$$

$$\mu^3 = \left(1-\frac{1}{3}\right)\mu^2 + \frac{1}{3}W^3 = \frac{2}{3}\mu^2 + \frac{1}{3}W^3 = \frac{1}{3}W^1 + \frac{1}{3}W^2 + \frac{1}{3}W^3$$

  - » In general we can write

$$\mu^n = \frac{1}{n}\sum_{i=1}^{n} W^i$$

  - » This means that using $\alpha_n = 1/n$ is the same as averaging. Possible to show that gives us the *best* convergence rate.

Derivative-based stochastic optimization

- Stepsize properties
  - » Finding an optimal stepsize, as we do in deterministic problems, is not possible for stochastic problems.
  - » Convergence theory (we prove convergence in the book) tells us that stepsize rules have to satisfy the following conditions

$$\alpha_n > 0$$

$$\sum_{n=1}^{\infty} \alpha_n = \infty$$

$$\sum_{n=1}^{\infty} \alpha_n^2 < \infty$$

  - » The stepsize $\alpha_n = \frac{1}{n}$ satisfies these conditions. For the case of finding an average, this is the best possible stepsize rule. But often this works very poorly. We will return to stepsizes later.

Derivative-based stochastic optimization

- Notes:
  - » Use this example to demonstrate how the gradient may take us in the wrong direction.
  - » Yet, averaging is the best possible way of estimating a mean!

Derivative-based stochastic optimization

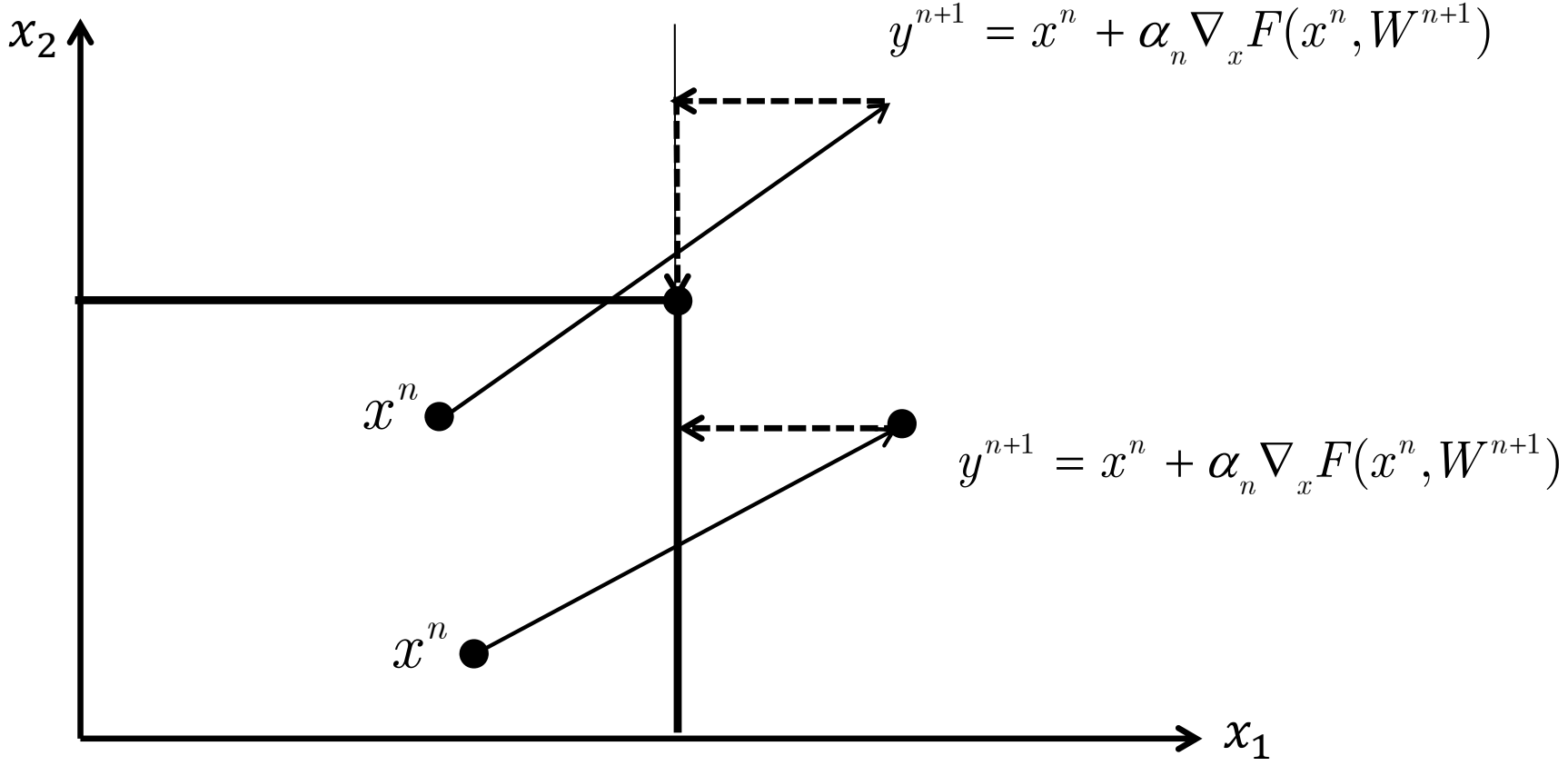
- Design elements:
  - » Starting points – it is common to initialize the algorithm from different starting points and choose the best final design. But picking the regions for these starting points can be tricky.
  - » Analytical vs. numerical gradients – There are many problems where we cannot compute analytical gradients, so we have to estimate them from finite differences.
  - » Gradient averaging – Gradients can be *very* noisy. Gradient smoothing can help. For numerical gradients, we might use repeated simulations to obtain better averages.
  - » Evaluating the algorithm. Need to compare final results to rate of convergence, and evaluate the consistency (how variable is the final solution).

Derivative-based stochastic search

Constrained optimization

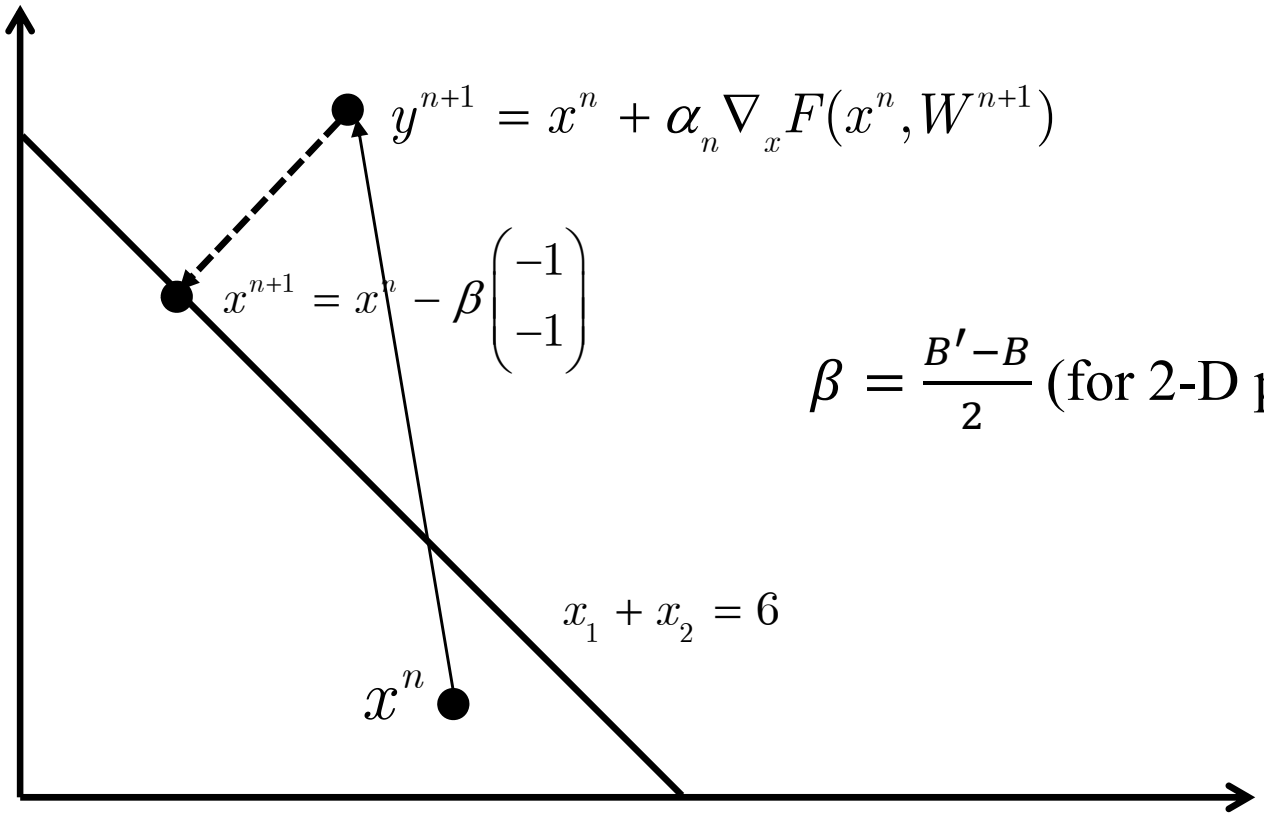
Derivative-based stochastic search

- Handling constrained problems:
  - » We might have constraints of the form: $x_i \leq u_i$.
  - » The update $y^{n+1} = x^n + \alpha_n \nabla_x F(x^n, W^{n+1})$ that takes us outside the box. We handle this using



Derivative-based stochastic search

- Sometimes we have budget constraints: $\sum_i x_i \leq B$
  - » The update $y^{n+1} = x^n + \alpha_n \nabla_x F(x^n, W^{n+1})$ might produce $\sum_i y_i^{n+1} = B' > B$. Map y^{n+1} back to x^{n+1} using:



$\beta = \frac{B'-B}{2}$ (for 2-D problem)

Derivative-based stochastic search

- For more general problems:
  - » Let
    $$X = \{x | Ax = b, x \geq 0\}$$
  - » We write the constrained version of the stochastic gradient operator as:
    $$x^n = \Pi_X[x^n + \alpha_n \nabla_x F(x^n)]$$
  - » We refer to Π_X as the projection operator onto the set X.
  - » A general way of performing the projection operator is to solve the nonlinear programming problem:
    $$\Pi_X[y^{n+1}] = argmax_x \left|\left|x - y^{n+1}\right|\right|_2$$
    where $\left|\left|x - y^{n+1}\right|\right|_2 = \sum_i (x_i - y_{i+1}^n)^2$.

Derivative-based stochastic search

As a sequential decision problem

Derivative-based stochastic search

- Finite time analysis
  - » In practice, we stop the algorithm after N iterations of running the stochastic gradient algorithm

$$x^n = x^{n-1} + \alpha_{n-1} \nabla_x F(x^{n-1}, W^n). \tag{5.12}$$

  - » With some stepsize rule such as

In practice we have to tune the stepsize formula. While there are many rules we could use, we will illustrate the key idea using a simple stepsize rule known as Kesten's rule given by

$$\alpha_n(\theta) = \frac{\theta}{\theta + N^n},$$

where θ is a tunable parameter and N^n counts the number of times that the objective function $F(x^n, W^{n+1})$ has declined (which means the stepsize remains constant while the function is improving).

Derivative-based stochastic search

- Finite time analysis
  - » There are two choices we have to make:
    - The type of stepsize policy
    - The type of gradient
      - – Gradient smoothing?
      - – Second-order methods?
    - Then we have to tune the stepsize policy

  - » A more formal statement of the problem is to write

Now, our stochastic gradient algorithm (5.12) becomes a policy $X^\pi(S^n)$ with state $S^n = (x^n, N^n)$, and where π captures the structure of the rule (e.g. the stepsize rule in (5.12)) and any tunable parameters (that is, θ). Let $x^{\pi,N}$ be the solution x^n for $n = N$, where we include π to indicate that our solution $x^{\pi,N}$ after N function evaluations depends on the policy π that we followed to get there. The problem of finding the best stepsize rule can be stated as

$$\max_\pi \mathbb{E}F(x^{\pi,N}, W), \tag{5.19}$$

Derivative-based stochastic optimization

- Stochastic search (derivative based)
  - » Basic problem:

    $$\max_x \mathbb{E}\{F(x,W) \mid S_0\}$$

  - » Stochastic gradient

    $$x^{n+1} = x^n + \alpha_n \nabla_x F(x^n, W^{n+1})$$

  - » Typically we are limited, to, say N iterations, so we want the best solution after N iterations. Let π be our "algorithm" and let $x^{\pi,N}$ be the solution that "algorithm" π produces after N iterations.
  - » Now we would state the optimization problem as

    $\max_\pi \mathbb{E}F(x^{\pi,n},W)$ where π is an algorithm (or policy)

  - » In other words, we want the *optimal algorithm*.

Derivative-based, finite horizon

- Derivative-based stochastic search – finite horizon
  - » We want a method (an algorithm) that produces the best solution by time N. This algorithm is like a policy, since it maps a state to an action:

$$x^{n+1} = x^n + \alpha_n \nabla_x F(x^n, W^{n+1})$$

  - » Assume that our stepsize rule is

$$\alpha_n = \frac{\theta}{\theta + N^n}$$

where N^n = number of times the solution has not improved.

  - » After n iterations, our "state" is

$$S^n = (x^n, N^n) \qquad S^{n+1} = S^M(S^n, \alpha_n, W^{n+1})$$

  - » Given the state S^n and the parameter θ, we can determine (after sampling W^{n+1}) the next state S^{n+1}.

Derivative-based, finite horizon

- Derivative-based stochastic search – finite horizon
  - » State variables
  - » $S^n = (x^n, N^n)$
  - » Decision variable: α_n made using policy $\pi(S^n) = \frac{\theta}{\theta+n}$
  - » Exogenous information: W^{n+1}
  - » Transition function $S^{n+1} = S^M(S^n, \alpha_n, W^{n+1})$

$$x^{n+1} = x^n + \alpha_n \nabla_x F(x^n, W^{n+1})$$

$$N^{n+1} = N^n + \begin{cases} 1 & If \nabla_x F(x^n, W^{n+1}) \nabla_x F(x^{n-1}, W^n) < 0 \\ 0 & Otherwise \end{cases}$$

  - » Our objective is to find the best stepsize policy that solves

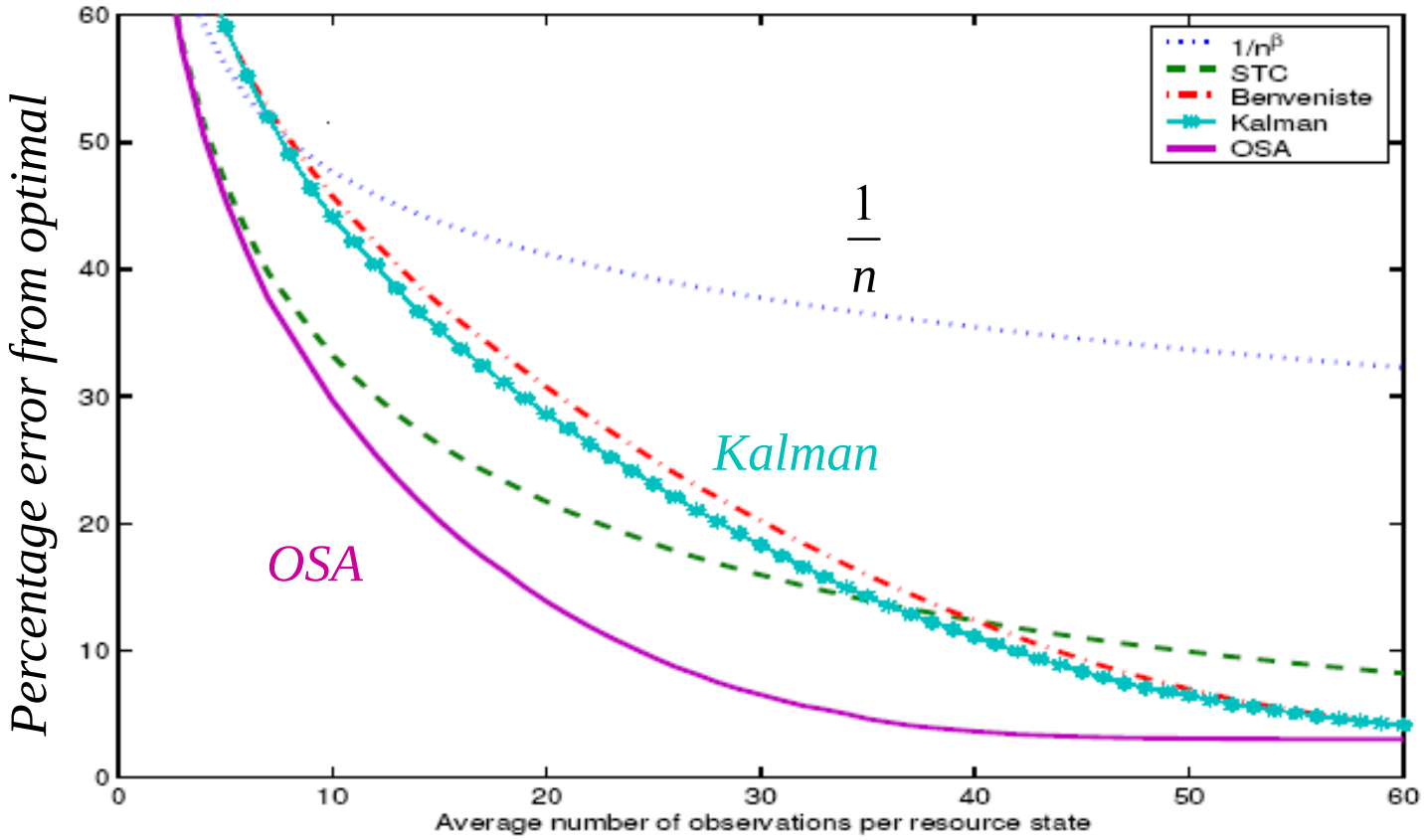
$$\max_\theta \mathbb{E} F(x^{\pi,N}, \widehat{W})$$

Derivative-based, finite horizon

- Notes:
  - » Our objective function might optimize over:
    - Different stepsize rules
    - Different parameters for each stepsize rule
    - Different ways for calculating the gradient
  - » We will review stepsize policies next.

Derivative-based, finite horizon

- Testing different algorithms ("policies")



» We want to optimize the rate of convergence:
  - Different stepsize rules
  - Different ways of computing the gradient

Derivative-based stochastic optimization

- Notes:
  - » The formulation of finding the *optimal algorithm* has been largely ignored in the research literature.
  - » Finding the best stepsize formula, or tuning the parameters of a stepsize formula, has been approached as an ad-hoc exercise:
    - ***Send the grad student down to the basement to test stepsize formulas and find the best one.***
  - » The problem is that the proper tuning can depend on issues like starting points. A poor choice of tuning parameters can destroy the performance of the algorithm.
  - » This has been largely ignored in the academic research literature, but it is important (and a potential area of research).

Numerical derivatives

Numerical derivatives

- Finite differences

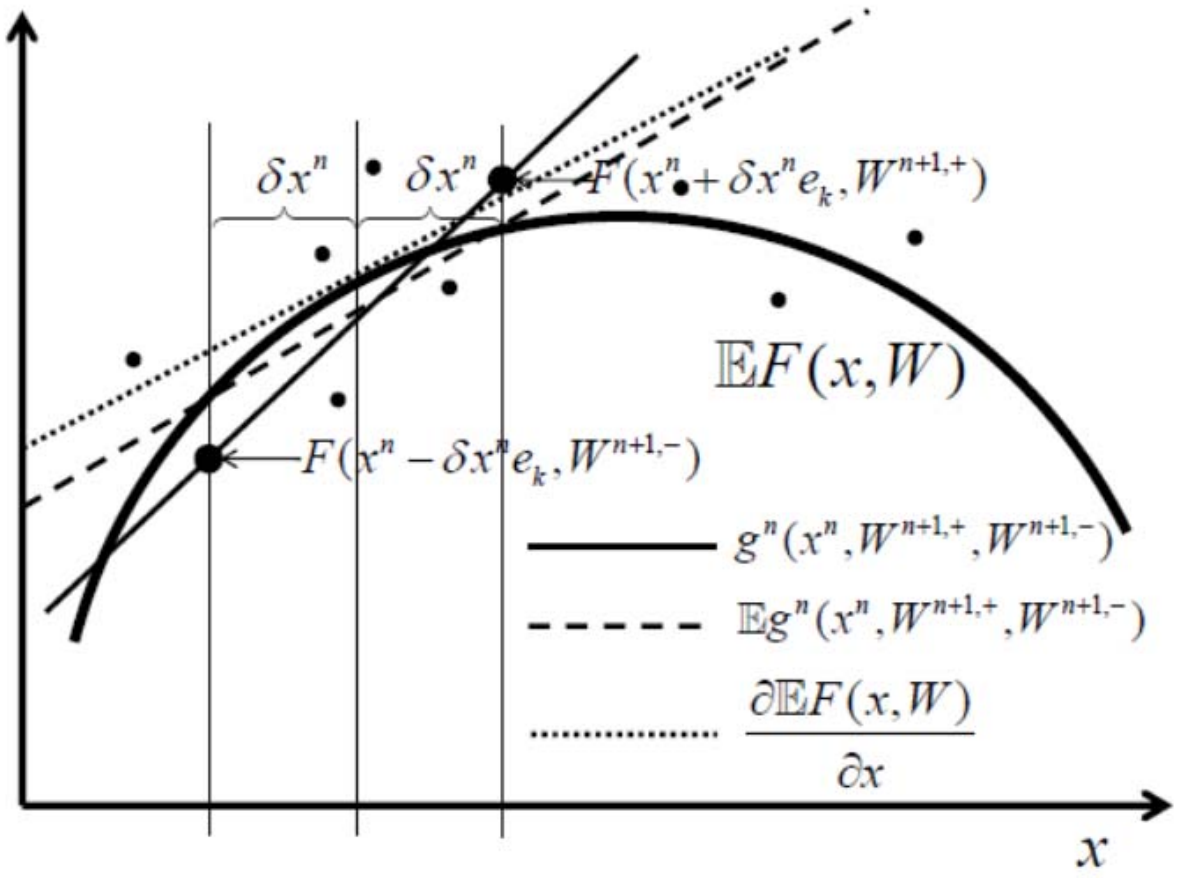


Figure 5.1 Different estimates of the gradient of $F(x,W)$ with a) the stochastic gradient $g^n(x^n, W^{n+1,+}, W^{n+1,-})$ (solid line), the expected finite difference $\mathbb{E}g^n(x^n, W^{n+1,+}, W^{n+1,-})$ (dashed line), and the exact slope at x^n, $\partial \mathbb{E}F(x^n, W^{n+1})/\partial x^n$.

Numerical derivatives

- Simulating the numerical derivative

  Let ω be a sample path of $W_1(\omega), W_2(\omega), \ldots$ with a sequence of states $S_{t+1}(\omega) = S^M(S_t(\omega), X_t^\pi(S_t(\omega)\,|\,\theta), W_{t+1}(\omega))$.

  Let

$$F^\pi(\theta,\omega) = \sum_{t=0}^{T} C\left(S_t(\omega), X_t^\pi(S_t(\omega)\,|\,\theta)\right)$$

  Now compute the stochastic numerical derivative using

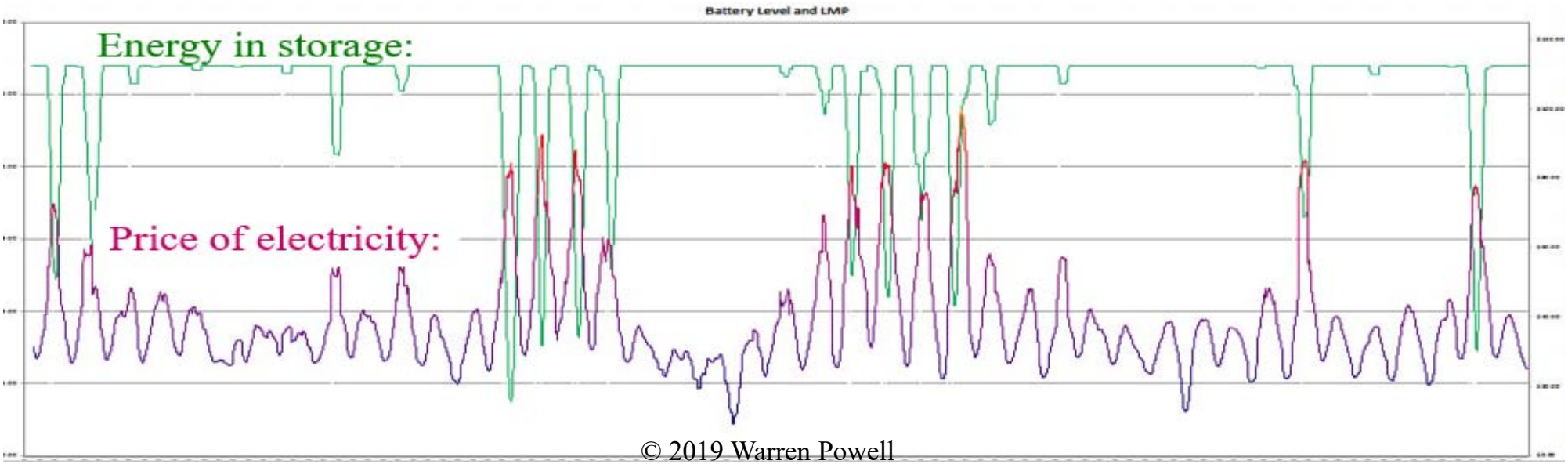
$$g(\omega) = \frac{F^\pi(\theta+\delta,\omega) - F^\pi(\theta,\omega)}{\delta}$$

  We are writing this assuming that $F^\pi(\theta+\delta,\omega)$ and $F^\pi(\theta,\omega)$ are both computed with the same sample path ω. This is best, but not required, and not always possible.

Numerical derivatives

- Finite differences
  - » We use finite differences in MOLTE-DB:
    - We wish to optimize the decision of when to charge or discharge a battery

$$X^{\pi}(S_t \mid \theta) = \begin{cases} +1 & \text{if } p_t < \theta^{\text{charge}} \\ 0 & \text{if } \theta^{\text{charge}} < p_t < \theta^{\text{discharge}} \\ -1 & \text{if } p_t > \theta^{\text{charge}} \end{cases}$$

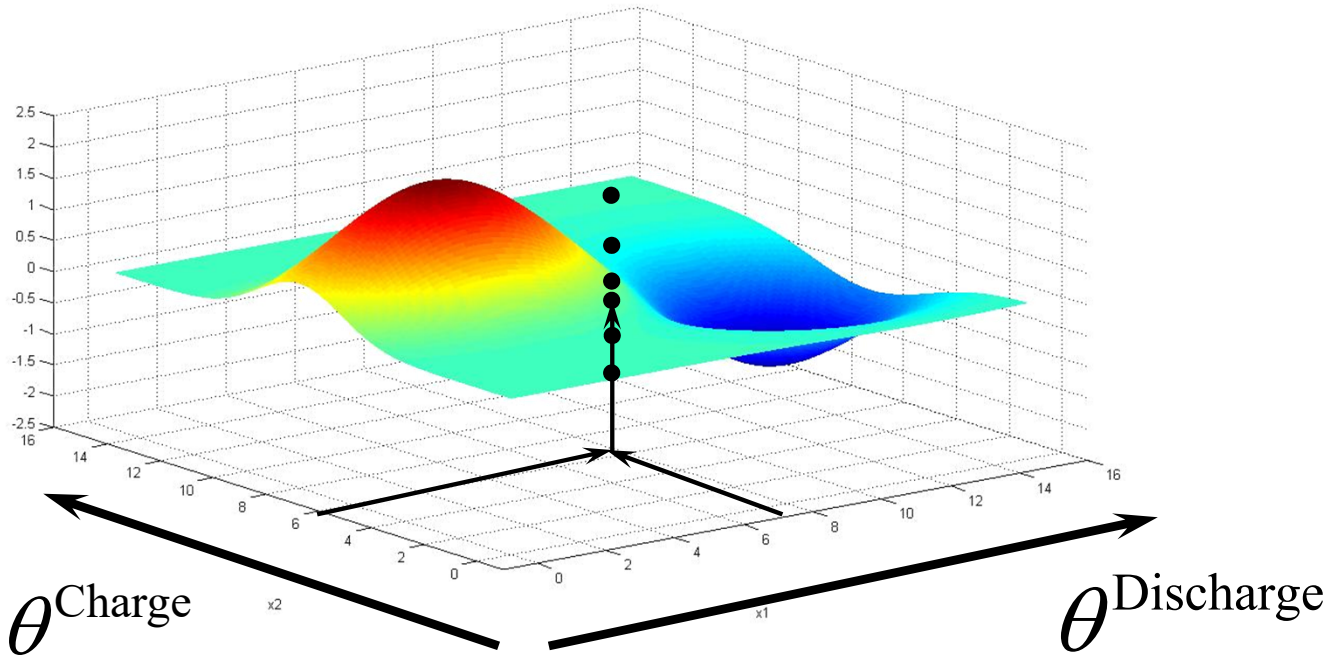


Numerical derivatives

- Finding the best policy
  - » We need to maximize

$$\max_\theta F(\theta) = \mathbb{E}\sum_{t=0}^{T} \gamma^t C\left(S_t, X_t^\pi(S_t \mid \theta)\right)$$

  - » We cannot compute the expectation, so we run simulations:



Numerical derivatives

- Simultaneous perturbation stochastic approximation
  - » Let:
    - x^n be a p – dimensional vector.
    - δ^n be a scalar perturbation
    - Z^n be a p –dimensional vector, with each element drawn from a normal (0,1) distribution.
  - » We can obtain a sampled estimate of the gradient $\nabla_x F(x^n, W^{n+1})$ using two function evaluations: $F(x^n + \delta^n Z^n)$ and $F(x^n + \delta^n Z^n)$

$$\nabla_x F(x^n, W^{n+1}) = \begin{bmatrix} \dfrac{F(x^n + \delta^n Z^n) - F(x^n + \delta^n Z^n)}{2\delta^n Z_1^n} \\ \dfrac{F(x^n + \delta^n Z^n) - F(x^n + \delta^n Z^n)}{2\delta^n Z_2^n} \\ \vdots \\ \dfrac{F(x^n + \delta^n Z^n) - F(x^n + \delta^n Z^n)}{2\delta^n Z_p^n} \end{bmatrix}$$

Numerical derivatives

- Notes:
  - » SPSA makes it possible to obtain sampled estimates of gradients for multi-dimensional vectors x with just two function evaluations.
  - » The gradients are not as high quality as those obtained by perturbing each individual dimension…
  - » … but there are many problems where the gradients are quite noisy. In these cases, it is necessary to perform multiple simulations and average them (called "mini-batches" in the literature). This is impractical even for relatively low-dimensional vectors. SPSA accelerates these computations dramatically when this is required.

Numerical derivatives

- Empirical issues
  - » Parameter tuning – These algorithms typically have several parameters:
    - One or more stepsize parameters
    - "Mini-batch" for averaging the gradient
  - » Scaling – The gradient and the decision variable is typically different units (in some cases, different decision variables can have different units).
  - » Simulating/testing the solution – Requires a stochastic model of the physical system to evaluate the solution.
  - » Robustness – How reliable/consistent is the algorithm?

A convergence proof

The older proof

An older proof

- Notes:
  - » The proof that follows is an older proof that uses fairly elementary mathematics (e.g. infinite sums).
  - » A key technique we will use is that we will assume that the algorithm is following a *sample path* ω.
  - » This means that every time we need a realization of the random variable W^n, we are going to substitute "ω" to represent a sample realization of W^n.
  - » Technically, anything that varies from one iteration to the next, such as x^n, depends on these observations, so we could write the decisions as $x^n(\omega)$, but this gets clumsy.
  - » We are going to prove that for a particular sequence $\omega = (W^1, W^2, \ldots, W^\infty)$, the decisions $x^n = x^n(\omega) \rightarrow x^*$ as $n \rightarrow \infty$.

An older proof

5.8.2 An older proof

Enough with probabilistic preliminaries. We wish to solve the unconstrained problem

$$\max_x \mathbb{E}F(x,\omega) \tag{5.34}$$

with x^* being the optimal solution. Let $g(x,\omega)$ be a stochastic ascent vector that satisfies

$$g(x,\omega)^T \nabla F(x,\omega) \geq 0. \tag{5.35}$$

For many problems, the most natural ascent vector is the gradient itself

$$g(x,\omega) = \nabla F(x,\omega) \tag{5.36}$$

which clearly satisfies (5.35).

We assume that $F(x) = \mathbb{E}F(x,\omega)$ is continuously differentiable and concave, with bounded first and second derivatives so that for finite M

$$-M \leq g(x,\omega)^T \nabla^2 F(x) g(x,\omega) \leq M. \tag{5.37}$$

A stochastic gradient algorithm (sometimes called a stochastic approximation method) is given by

$$\bar{x}^n = \bar{x}^{n-1} + \alpha_{n-1} g(\bar{x}^{n-1},\omega). \tag{5.38}$$

An older proof

- Stepsizes

We make the following (standard) assumptions on stepsizes

$$\alpha_n > 0 \text{ for all } n \geq 0, \tag{5.39}$$

$$\sum_{n=0}^{\infty} \alpha_n = \infty, \tag{5.40}$$

$$\sum_{n=0}^{\infty} (\alpha_n)^2 < \infty. \tag{5.41}$$

  - These conditions are satisfied by:
  - $\theta = \frac{1}{n}$
  - $\theta = \frac{\theta}{\theta + n}$
  - $\theta = \frac{1}{n^{\beta}}, \quad \beta \in (.5,1]$

An older proof

We want to show that under suitable assumptions, the sequence generated by (5.38) converges to an optimal solution. That is, we want to show that

$$\lim_{n\to\infty} x^n = x^* \ a.s. \tag{5.42}$$

We now use Taylor's theorem (remember Taylor's theorem from freshman calculus?), which says that for any continuously differentiable convex function $F(x)$, there exists a parameter $0 \leq \eta \leq 1$ that satisfies for a given x and x^0

$$F(x) = F(x^0) + \nabla F(x^0 + \eta(x - x^0))(x - x^0). \tag{5.43}$$

This is the first-order version of Taylor's theorem. The second-order version takes the form

$$F(x) = F(x^0) + \nabla F(x^0)(x - x^0) + \frac{1}{2}(x - x^0)^T \nabla^2 F(x^0 + \eta(x - x^0))(x - x^0) \tag{5.44}$$

for some $0 \leq \eta \leq 1$. We use the second-order version. In addition, since our problem is stochastic, we will replace $F(x)$ with $F(x, \omega)$ where ω tells us what sample path we are on, which in turn tells us the value of W.

An older proof

To simplify our notation, we are going to replace x^0 with x^{n-1}, x with x^n, and finally we will use

$$g^n = g(x^{n-1}, \omega). \tag{5.45}$$

This means that, by definition of our algorithm,

$$\begin{aligned} x - x^0 &= x^n - x^{n-1} \\ &= (x^{n-1} + \alpha_{n-1} g^n) - x^{n-1} \\ &= \alpha_{n-1} g^n. \end{aligned}$$

From our stochastic gradient algorithm (5.38), we may write

$$\begin{aligned} F(x^n, \omega) &= F(x^{n-1} + \alpha_{n-1} g^n, \omega) \\ &= F(x^{n-1}, \omega) + \nabla F(x^{n-1}, \omega)(\alpha_{n-1} g^n) \\ &\quad + \frac{1}{2}(\alpha_{n-1} g^n)^T \nabla^2 F(x^{n-1} + \eta \alpha_{n-1} g^n, \omega)(\alpha_{n-1} g^n). \end{aligned} \tag{5.46}$$

An older proof

It is now time to use a *standard mathematician's trick*. We sum both sides of (5.46) to get

$$\sum_{n=1}^{N} F(x^n,\omega) = \sum_{n=1}^{N} F(x^{n-1},\omega) + \sum_{n=1}^{N} \nabla F(x^{n-1},\omega)(\alpha_{n-1}g^n) + \frac{1}{2}\sum_{n=1}^{N}(\alpha_{n-1}g^n)^T\nabla^2 F\left(x^{n-1}+\eta\alpha_{n-1}g^n,\omega\right)(\alpha_{n-1}g^n). \quad (5.47)$$

Note that the terms $F(x^n), n = 2, 3, \ldots, N$ appear on both sides of (5.47). We can cancel these. We then use our lower bound on the quadratic term (5.37) to write

$$F(x^N,\omega) \geq F(x^0,\omega) + \sum_{n=1}^{N} \nabla F(x^{n-1},\omega)(\alpha_{n-1}g^n) + \frac{1}{2}\sum_{n=1}^{N}(\alpha_{n-1})^2(-M). \quad (5.48)$$

An older proof

We now want to take the limit of both sides of (5.48) as $N \rightarrow \infty$. In doing so, we want to show that everything must be bounded. We know that $F(x^N)$ is bounded (*almost surely*) because we assumed that the original function was bounded. We next use the assumption (5.41) that the infinite sum of the squares of the stepsizes is also bounded to conclude that the rightmost term in (5.48) is bounded. Finally, we use (5.35) to claim that all the terms in the remaining summation ($\sum_{n=1}^{N} \nabla F(x^{n-1})(\alpha_{n-1} g^n)$) are positive. That means that this term is also bounded (from both above and below).

What do we get with all this boundedness? Well, if

$$\sum_{n=1}^{\infty} \alpha_{n-1} \nabla F(x^n, \omega) g^n < \infty \text{ for all } \omega \tag{5.49}$$

and (from (5.40))

$$\sum_{n=1}^{\infty} \alpha_{n-1} = \infty. \tag{5.50}$$

We can conclude that

$$\sum_{n=1}^{\infty} \nabla F(x^{n-1}, \omega) g^n < \infty. \tag{5.51}$$

Since all the terms in (5.51) are positive, they must go to zero. (Remember, everything here is true *almost surely*; after a while, it gets a little boring to keep saying *almost surely* every time. It is a little like reading Chinese fortune cookies and adding the automatic phrase "under the sheets" at the end of every fortune.)

An older proof

We are basically done except for some relatively difficult (albeit important if you are ever going to do your own proofs) technical points to really prove convergence. At this point, we would use technical conditions on the properties of our ascent vector g^n to argue that if $\nabla F(x^n,\omega)g^n \to 0$ then $\nabla F(x^n,\omega) \to 0$, (it is okay if g^n goes to zero as $F(x^n,\omega)$ goes to zero, but it cannot go to zero too quickly).

This proof was first proposed in the early 1950's by Robbins and Monro and became the basis of a large area of investigation under the heading of stochastic approximation methods. A separate community, growing out of the Soviet literature in the 1960's, addressed these problems under the name of stochastic gradient (or stochastic quasi-gradient) methods. More modern proofs are based on the use of martingale processes, which do not start with Taylor's formula and do not (always) need the continuity conditions that this approach needs.

Our presentation does, however, help to present several key ideas that are present in most proofs of this type. First, concepts of almost sure convergence are virtually standard. Second, it is common to set up equations such as (5.46) and then take a finite sum as in (5.47) using the alternating terms in the sum to cancel all but the first and last elements of the sequence of some function (in our case, $F(x^{n-1},\omega)$). We then establish the boundedness of this expression as $N \to \infty$, which will require the assumption that $\sum_{n=1}^{\infty}(\alpha_{n-1})^2 < \infty$. Then, the assumption $\sum_{n=1}^{\infty}\alpha_{n-1} = \infty$ is used to show that if the remaining sum is bounded, then its terms must go to zero.

Chapter 6: Stepsizes

Stepsize policies

There is a wide range of adaptive learning problems that depend on an iteration of the form we first saw in chapter 5 that looks like

$$x^{n+1} = x^n + \alpha_n \nabla_x F(x^n, W^{n+1}). \tag{6.1}$$

The stochastic gradient $\nabla_x F(x^n, W^{n+1})$ tells us what direction to go in, but we need the stepsize α_n to tell us how far we should move.

There are two important settings where this formula is used. The first is where we are maximizing some metric such as contributions, utility or performance. In these settings, the units of $\nabla_x F(x^{n-1}, W^n)$ and the decision variable x are different, so the stepsize has to perform the scaling so that the size of $\alpha_n \nabla_x F(x^n, W^{n+1})$ is not too large or too small relative to x^n.

A second and very important setting arises in what are known as *supervisory learning* settings. In this context, we are trying to estimate some function $f(x|\theta)$ using observations $y = f(x|\theta) + \varepsilon$. In this context, $f(x|\theta)$ and y have the same scale. We encounter these problems in three settings:

- Approximating the function $\mathbb{E}F(x, W)$ to create an estimate $\overline{F}(x)$ that can be optimized.
- Approximating the value $V_t(S_t)$ of being in a state S_t and then following some policy (we encounter this problem starting in chapters 17 and 18 when we introduce approximate dynamic programming).
- Creating a parameterized policy $X^\pi(S|\theta)$ (we might call this $A^\pi(S)$ if we are using action a, or $U^\pi(S)$ if we are using control u). Here, we assume we have access to some method of creating a decision x and then we use this to create a parameterized policy $X^\pi(S|\theta)$.

Stepsize policies

In chapter 3, we saw a range of methods for approximating functions. Imagine that we face the simplest problem of estimating the mean of a random variable W. Assume we run our simulation observe W^{n+1}. We wish to create an estimate μ which is the solution of the following stochastic optimization problem

$$\min_x \mathbb{E}\frac{1}{2}(x - W)^2. \tag{6.2}$$

Let $F(x, W) = \frac{1}{2}(x - W)^2$. The stochastic gradient of $F(\mu, W)$ with respect to μ is

$$\nabla_\mu F(x, W) = (x - W).$$

We can optimize (6.2) using a stochastic gradient algorithm which we would write (remember that we are minimizing):

$$\begin{aligned} x^{n+1} &= x^n - \alpha_n \nabla F(x^n, W^{n+1}) && (6.3)\\ &= x^n - \alpha_n (x^n - W^{n+1}) && (6.4)\\ &= (1 - \alpha_n)x^n + \alpha_n W^{n+1}. && (6.5) \end{aligned}$$

Equation (6.5) will be familiar to many readers as a smoothing algorithm (also known as a *linear filter* in signal processing). The important observation is that in this setting, the stepsize α_n needs to be between 0 and 1 since μ and W are the same scale.

Stepsize policies

We made the argument in section 5.3.3 that the iterate in equation (6.1) can be viewed as a dynamical system which is controlled by the stepsize rule, which means it is effectively a form of policy. If we want the best answer within a budget of N iterations, we want to find the best rule (or policy) to achieve this.

We divide our presentation of stepsize rules into three classes:

Deterministic policies - These are stepsize policies that are deterministic functions of the iteration counter n.

Stochastic policies - These are policies where the stepsize at iteration n depends on the statistics computed from the trajectory of the algorithm.

Optimal policies - Our deterministic and stochastic stepsize policies are heuristic, and as a result require tuning one or more parameters. Optimal policies are derived from a formal model which is typically a simplified problem. These policies tend to be more complex, but eliminate or at least minimize the need for parameter tuning.

Stepsize policies

- Deterministic policies (state independent)

Constant stepsizes

A constant stepsize rule is simply

$$\alpha_{n-1} = \begin{cases} 1 & \text{if } n = 1, \\ \bar{\alpha} & \text{otherwise,} \end{cases}$$

where $\bar{\alpha}$ is a stepsize that we have chosen. It is common to start with a stepsize of 1 so that we do not need an initial value $\bar{\theta}^0$ for our statistic.

» Popular for transient systems

Stepsize policies

Generalized harmonic stepsizes

A generalization of the $1/n$ rule is the generalized harmonic sequence given by

$$\alpha_{n-1} = \frac{a}{a+n-1}. \tag{6.12}$$

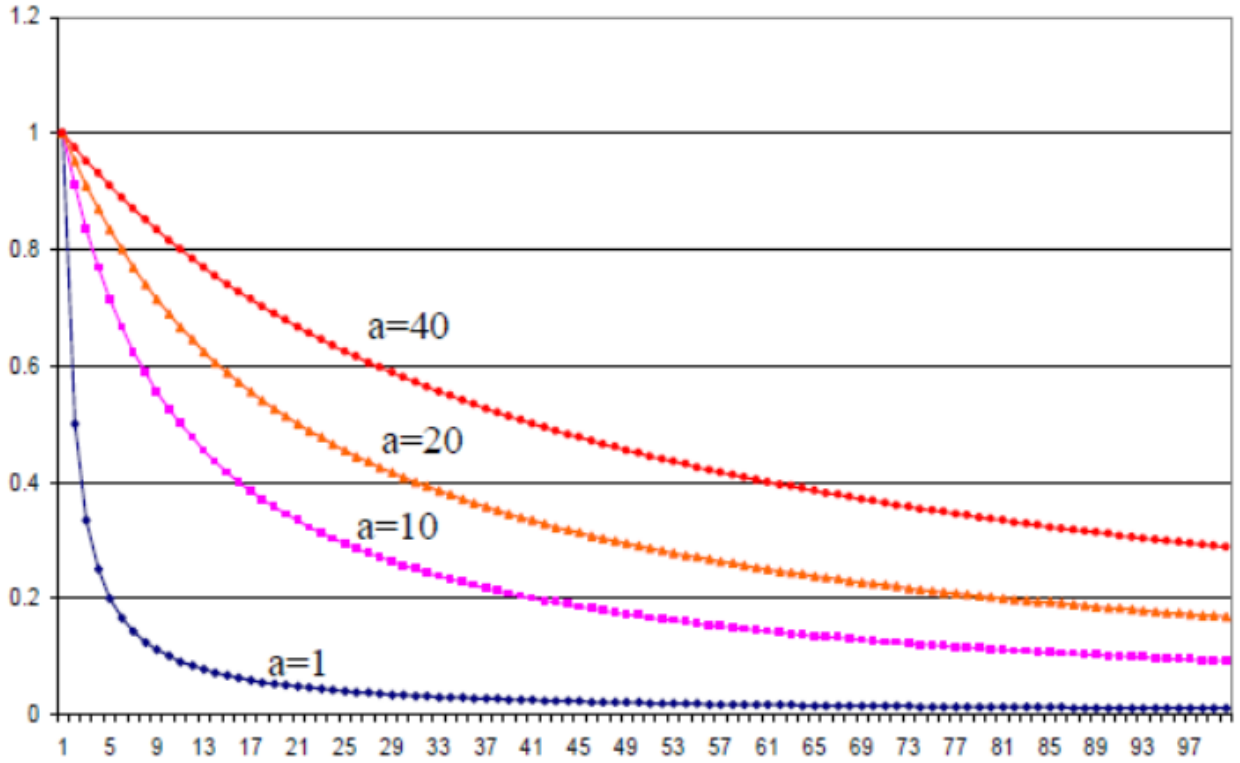


Figure 6.3 Stepsizes for $a/(a+n)$ while varying a.

Stepsize policies

- Stochastic stepsize policies (State-dependent)
  - » These adapt to the behavior of the algorithm

6.2.2 Convergence conditions

When the stepsize depends on the history of the process, the stepsize itself becomes a random variable. This change requires some subtle modifications to our requirements for convergence (equations (6.8) and (6.9)). For technical reasons, our convergence criteria change to

$$\alpha_n > 0, \text{ almost surely,} \tag{6.16}$$

$$\sum_{n=0}^{\infty} \alpha_n = \infty \text{ almost surely,} \tag{6.17}$$

$$\mathbb{E}\left\{\sum_{n=0}^{\infty} (\alpha_n)^2\right\} < \infty. \tag{6.18}$$

Stepsize policies

6.2.3 Recipes for stochastic stepsizes

The desire to find stepsize rules that adapt to the data has become a small cottage industry which has produced a variety of formulas with varying degrees of sophistication and convergence guarantees. This section provides a brief sample of these efforts.

To present our stochastic stepsize formulas, we need to define a few quantities. Recall that our basic updating expression is given by

$$\bar{\theta}^n = (1-\alpha_{n-1})\bar{\theta}^{n-1} + \alpha_{n-1}\hat{\theta}^n.$$

$\bar{\theta}^{n-1}$ is our estimate of the next observation, given by $\hat{\theta}^n$. The difference between the estimate and the actual can be treated as the error, given by

$$\varepsilon^n = \bar{\theta}^{n-1} - \hat{\theta}^n.$$

We may wish to smooth the error in the estimate, which we designate by the function

$$S(\varepsilon^n) = (1-\beta)S(\varepsilon^{n-1}) + \beta\varepsilon^n.$$

Some formulas depend on tracking changes in the sign of the error. This can be done using the indicator function

$$1_{\{X\}} = \begin{cases} 1 & \text{if the logical condition } X \text{ is true,} \\ 0 & \text{otherwise.} \end{cases}$$

Stepsize policies

Kesten's rule

Kesten's rule was one of the earliest stepsize rules which took advantage of a simple principle. If we are far from the optimal, the errors tend to all have the same sign. As we get close, the errors tend to alternate. Exploiting this simple observation, Kesten proposed the following simple rule:

$$\alpha_{n-1} = \frac{a}{a + K^n - 1}, \tag{6.19}$$

where a is a parameter to be calibrated. K^n counts the number of times that the sign of the error has changed, where we use

$$K^n = \begin{cases} n & \text{if } n = 1, 2, \\ K^{n-1} + 1_{\{\varepsilon^n \varepsilon^{n-1} < 0\}} & \text{if } n > 2. \end{cases} \tag{6.20}$$

Kesten's rule is particularly well suited to initialization problems. It slows the reduction in the stepsize as long as the error exhibits the same sign (and indication that the algorithm is still climbing into the correct region). However, the stepsize declines monotonically. This is typically fine for most dynamic programming applications, but can encounter problems in situations with delayed learning.

Stepsize policies

ADAM ADAM (Adaptive Moment Estimation) is another stepsize policy that has attracted attention in recent years. As above, let $g^n = \nabla_x F(x^{n-1}, W^n)$ be our gradient, and let g_i^n be the ith element. ADAM proceeds by adaptively computing means and variances according to

$$m_i^n = \beta_1 m_i^{n-1} + (1-\beta_1) g_i^n, \tag{6.24}$$

$$v_i^n = \beta_2 v_i^{n-1} + (1-\beta_2)(g_i^n)^2. \tag{6.25}$$

These updating equations introduce biases when the data is nonstationary, which is typically the case in stochastic optimization. ADAM compensates for these biases using

$$\bar{m}_i^n = \frac{m_i^n}{1-\beta_1},$$

$$\bar{v}_i^n = \frac{v_i^n}{1-\beta_2}.$$

The stochastic gradient equation for ADAM is then given by

$$x_i^{n+1} = x_i^n + \frac{\eta}{\sqrt{\bar{v}_i^n + \epsilon}}. \tag{6.26}$$

Stepsize policies

- Adagrad

AdaGrad AdaGrad ("adaptive gradient") is a relatively recent stepsize policy that has attracted considerable attention in the machine learning literature which not only enjoys nice theoretical performance guarantees, but has also become quite popular because it seems to work quite well in practice.

Assume that we are trying to solve our standard problem

$$\max_x \mathbb{E}_W F(x, W),$$

where we make the assumption that not only is x a vector, but also that the scaling for each dimension might be different (an issue we have ignored so far). To simplify the notation a bit, let the stochastic gradient with respect to x_i, $i = 1, \ldots, I$ be given by

$$g_i^n = \nabla_{x_i} F(x^{n-1}, W^n).$$

Now create a $I \times I$ diagonal matrix G^n where the $(i,i)th$ element G_{ii}^n is given by

$$G_{ii}^n = \sum_{m=1}^{n} (g_i^n)^2.$$

Stepsize policies

- Adagrad

We then set a stepsize for the ith dimension using

$$\alpha_{ni} = \frac{\eta}{(G^n_{ii})^2 + \epsilon}, \tag{6.27}$$

where ϵ is a small number (e.g. 10^{-8} to avoid the possibility of dividing by zero). This can be written in matrix form using

$$\alpha_n = \frac{\eta}{\sqrt{G^n + \epsilon}} \odot g_t, \tag{6.28}$$

where α_n is an I-dimensional matrix. The right way to understand equation (6.28) is equation (6.27).

AdaGrad does an unusually good job of adapting to the behavior of a function. It also adapts to potentially different behaviors of each dimension. For example, we might be solving a machine learning problem to learn a parameter vector θ (this would be the decision variable instead of x) for a linear model of the form

$$y = \theta_0 + \theta_1 X_1 + \theta_2 X_2 + \ldots.$$

The explanatory variables $X_1, X_2, \ldots$ can take on values in completely different ranges. In a medical setting, X_1 might be blood sugar with values between 5 and 8, while X_2 might be the weight of a patient that could range between 100 and 300 pounds. The coefficients θ_1 and θ_2 would be scaled according to the inverse of the scales of the explanatory variables.

6.3 OPTIMAL STEPSIZES

Given the variety of stepsize formulas we can choose from, it seems natural to ask whether there is an optimal stepsize rule. Before we can answer such a question, we have to define exactly what we mean by it. Assume that we are trying to estimate a parameter (such as a value of being in a state or the slope of a value function) that we denote by θ^n that may be changing over time. At iteration n, our estimate of θ^n, $\bar{\theta}^n$, is a random variable that depends on our stepsize rule. To express this dependence, let α represent a stepsize rule, and let $\bar{\theta}^n(\alpha)$ be the estimate of the parameter μ after iteration n using stepsize rule α. We would like to choose a stepsize rule to minimize

$$\min_{\alpha} \mathbb{E}(\bar{\theta}^n(\alpha) - \theta^n)^2. \tag{6.24}$$

Here, the expectation is over the entire history of the algorithm and requires (in principle) knowing the true value of the parameter being estimated. If we could solve this problem (which requires knowing certain parameters about the underlying distributions), we would obtain a deterministic stepsize rule. In practice, we do not generally know these parameters which need to be estimated from data, producing a stochastic stepsize rule.

There are other objective functions we could use. For example, instead of minimizing the distance to an unknown parameter sequence θ^n, we could solve the minimization problem

$$\min_{\alpha} \mathbb{E}\left\{(\bar{\theta}^n(\alpha) - \hat{\theta}^{n+1})^2\right\}, \tag{6.25}$$

where we are trying to minimize the deviation between our prediction, obtained at iteration n, and the actual observation at $n+1$. Here, we are again proposing an unconditional expectation, which means that $\bar{\theta}^n(\alpha)$ is a random variable within the expectation. Alternatively,

Stepsize policies

we could condition on our history up to iteration n

$$\min_{\alpha} \mathbb{E}^n \left\{ (\bar{\theta}^n(\alpha) - \hat{\theta}^{n+1})^2 \right\} \tag{6.26}$$

where $\mathbb{E}^n$ means that we are taking the expectation given what we know at iteration n (which means that $\bar{\theta}^n(\alpha)$ is a constant). For readers familiar with the the language of filtrations, we would write the expectation as $\mathbb{E}\left\{ (\bar{\theta}^n(\alpha) - \hat{\theta}^{n+1})^2 | H^n \right\}$, where H^n is the history of the process up through iteration n (that is, the entire sequence $W^1, \ldots, W^n$). In this formulation $\bar{\theta}^n(\alpha)$ is now deterministic at iteration n (because we are conditioning on the history up through iteration n), whereas in (6.25), $\bar{\theta}^n(\alpha)$ is random (since we are not conditioning on the history). The difference between these two objective functions is subtle but significant.

Stepsize policies

6.3.1 Optimal stepsizes for stationary data

Assume that we observe $\hat{\theta}^n$ at iteration n and that the observations $\hat{\theta}^n$ can be described by

$$\hat{\theta}^n = \theta + \varepsilon^n$$

where μ is an unknown constant and ε^n is a stationary sequence of independent and identically distributed random deviations with mean 0 and variance σ^2. We can approach the problem of estimating μ from two perspectives: choosing the best stepsize and choosing the best linear combination of the estimates. That is, we may choose to write our estimate $\bar{\theta}^n$ after n observations in the form

$$\bar{\theta}^n = \sum_{m=1}^{n} a_m^n \hat{\theta}_m.$$

For our discussion, we will fix n and work to determine the coefficients a_m (recognizing that they can depend on the iteration). We would like our statistic to have two properties: It should be unbiased, and it should have minimum variance (that is, it should solve (6.24)). To be unbiased, it should satisfy

$$\begin{aligned}\mathbb{E}\left[\sum_{m=1}^{n} a_m \hat{\theta}_m\right] &= \sum_{m=1}^{n} a_m \mathbb{E}\hat{\theta}_m \\ &= \sum_{m=1}^{n} a_m \theta \\ &= \theta.\end{aligned}$$

Stepsize policies

The variance of our estimator is given by:

$$Var(\bar{\theta}^n) = Var\left[\sum_{m=1}^{n} a_m \hat{\theta}_m\right].$$

We use our assumption that the random deviations are independent, which allows us to write

$$\begin{aligned} Var(\bar{\theta}^n) &= \sum_{m=1}^{n} Var[a_m \hat{\theta}_m] \\ &= \sum_{m=1}^{n} a_m^2 Var[\hat{\theta}_m] \\ &= \sigma^2 \sum_{m=1}^{n} a_m^2. \end{aligned} \tag{6.27}$$

Now we face the problem of finding $a_1, \ldots, a_n$ to minimize (6.27) subject to the requirement that $\sum_m a_m = 1$. This problem is easily solved using the Lagrange multiplier method.

Stepsize policies

We start with the nonlinear programming problem

$$\min_{\{a_m\}} \sum_{m=1}^{n} a_m^2$$

subject to

$$\sum_{m=1}^{n} a_m = 1, \tag{6.28}$$

$$a_m \geq 0. \tag{6.29}$$

We relax constraint (6.28) and add it to the objective function

$$\min_{\{a_m\}} L(a,\lambda) = \sum_{m=1}^{n} a_m^2 - \lambda\left(\sum_{m=1}^{n} a_m - 1\right)$$

subject to (6.29). We are now going to try to solve $L(a,\lambda)$ (known as the "Lagrangian") and hope that the coefficients a are all nonnegative. If this is true, we can take derivatives and set them equal to zero

$$\frac{\partial L(a,\lambda)}{\partial a_m} = 2a_m - \lambda. \tag{6.30}$$

Stepsize policies

The optimal solution (a^*, λ^*) would then satisfy

$$\frac{\partial L(a, \lambda)}{\partial a_m} = 0.$$

This means that at optimality

$$a_m = \lambda/2,$$

which tells us that the coefficients a_m are all equal. Combining this result with the requirement that they sum to one gives the expected result:

$$a_m = \frac{1}{n}.$$

In other words, our best estimate is a sample average. From this (somewhat obvious) result, we can obtain the optimal stepsize, since we already know that $\alpha_{n-1} = 1/n$ is the same as using a sample average.

This result tells us that if the underlying data is stationary, and we have no prior information about the sample mean, then the best stepsize rule is the basic $1/n$ rule. Using any other rule requires that there be some violation in our basic assumptions. In practice, the most common violation is that the observations are not stationary because they are derived from a process where we are searching for the best solution.

Stepsize policies

6.3.2 Optimal stepsizes for nonstationary data - I

Assume now that our parameter evolves over time (iterations) according to the process

$$\theta^n = \theta^{n-1} + \xi^n, \tag{6.31}$$

where $\mathbb{E}\xi^n = 0$ is a zero mean drift term with variance $(\sigma^\xi)^2$. As before, we measure θ^n with an error according to

$$\hat{\theta}^n = \theta^n + \varepsilon^n.$$

We want to choose a stepsize so that we minimize the mean squared error. This problem can be solved using the Kalman filter. The Kalman filter is a powerful recursive regression technique, but we adapt it here for the problem of estimating a single parameter. Typical applications of the Kalman filter assume that the variance of ξ^n, given by $(\sigma^\xi)^2$, and the variance of the measurement error, ε^n, given by σ^2, are known. In this case, the Kalman filter would compute a stepsize (generally referred to as the gain) using

$$\alpha_n = \frac{(\sigma^\xi)^2}{\nu^n + \sigma^2}, \tag{6.32}$$

where ν^n is computed recursively using

$$\nu^n = (1 - \alpha_{n-1})\nu^{n-1} + (\sigma^\xi)^2. \tag{6.33}$$

Stepsize policies

Remember that $\alpha_0 = 1$, so we do not need a value of ν^0. For our application, we do not know the variances so these have to be estimated from data. We first estimate the bias using

$$\bar{\beta}^n = (1 - \eta_{n-1})\,\bar{\beta}^{n-1} + \eta_{n-1}\left(\bar{\theta}^{n-1} - \hat{\theta}^n\right), \tag{6.34}$$

where η_{n-1} is a simple stepsize rule such as the harmonic stepsize rule or McClain's formula. We then estimate the total error sum of squares using

$$\bar{\nu}^n = (1 - \eta_{n-1})\,\bar{\nu}^{n-1} + \eta_{n-1}\left(\bar{\theta}^{n-1} - \hat{\theta}^n\right)^2. \tag{6.35}$$

Finally, we estimate the variance of the error using

$$(\bar{\sigma}^n)^2 = \frac{\bar{\nu}^n - (\bar{\beta}^n)^2}{1 + \bar{\lambda}^{n-1}}, \tag{6.36}$$

where $\bar{\lambda}^{n-1}$ is computed using

$$\lambda^n = \begin{cases} (\alpha_{n-1})^2, & n = 1, \\ (1 - \alpha_{n-1})^2\lambda^{n-1} + (\alpha_{n-1})^2, & n > 1. \end{cases}$$

We use $(\bar{\sigma}^n)^2$ as our estimate of σ^2. We then propose to use $\left(\bar{\beta}^n\right)^2$ as our estimate of $(\sigma^\xi)^2$. This is purely an approximation, but experimental work suggests that it performs quite well, and it is relatively easy to implement.

6.3.3 Optimal stepsizes for nonstationary data - II

In dynamic programming, we are trying to estimate the value of being in a state (call it v) by $\bar{v}$ which is estimated from a sequence of random observations $\hat{v}$. The problem we encounter is that $\hat{v}$ might depend on a value function approximation which is steadily increasing, which means that the observations $\hat{v}$ are nonstationary. Furthermore, unlike the assumption made by the Kalman filter that the mean of $\hat{v}$ is varying in a zero-mean way, our observations of $\hat{v}$ might be steadily increasing. This would be the same as assuming that $\mathbb{E}\xi = \mu > 0$ in the section above. In this section, we derive the Kalman filter learning rate for biased estimates.

Our challenge is to devise a stepsize that strikes a balance between minimizing error (which prefers a smaller stepsize) and responding to the nonstationary data (which works better with a large stepsize). We return to our basic model

$$\hat{\theta}^n = \theta^n + \varepsilon^n,$$

where θ^n varies over time, but it might be steadily increasing or decreasing. This would be similar to the model in the previous section (equation (6.31)) but where ξ^n has a nonzero mean. As before we assume that $\{\varepsilon^n\}_{n=1,2,\ldots}$ are independent and identically distributed with mean value of zero and variance, σ^2. We perform the usual stochastic gradient update to obtain our estimates of the mean

$$\bar{\theta}^n(\alpha_{n-1}) \quad = \quad (1-\alpha_{n-1})\,\bar{\theta}^{n-1} + \alpha_{n-1}\hat{\theta}^n. \tag{6.37}$$

We wish to find α_{n-1} that solves,

$$\min_{\alpha_{n-1}} F(\alpha_{n-1}) = \mathbb{E}\left[\left(\bar{\theta}^n(\alpha_{n-1}) - \theta^n\right)^2\right]. \tag{6.38}$$

Stepsize policies

We wish to find α_{n-1} that solves,

$$\min_{\alpha_{n-1}} F(\alpha_{n-1}) = \mathbb{E}\left[\left(\bar{\theta}^n(\alpha_{n-1}) - \theta^n\right)^2\right]. \tag{6.38}$$

It is important to realize that we are trying to choose α_{n-1} to minimize the *unconditional* expectation of the error between $\bar{\theta}^n$ and the true value θ^n. For this reason, our stepsize rule will be deterministic, since we are not allowing it to depend on the information obtained up through iteration n.

We assume that the observation at iteration n is unbiased, which is to say

$$\mathbb{E}\left[\hat{\theta}^n\right] = \theta^n. \tag{6.39}$$

But the smoothed estimate is biased because we are using simple smoothing on nonstationary data. We denote this bias as

$$\begin{aligned} \beta^{n-1} &= \mathbb{E}\left[\bar{\theta}^{n-1} - \theta^n\right] \\ &= \mathbb{E}\left[\bar{\theta}^{n-1}\right] - \theta^n. \end{aligned} \tag{6.40}$$

Stepsize policies

We note that β^{n-1} is the bias computed after iteration $n-1$ (that is, after we have computed $\bar{\theta}^{n-1}$). β^{n-1} is the bias when we use $\bar{\theta}^{n-1}$ as an estimate of θ^n.

The variance of the observation $\hat{\theta}^n$ is computed as follows:

$$\begin{aligned} Var\left[\hat{\theta}^n\right] &= \mathbb{E}\left[\left(\hat{\theta}^n - \theta^n\right)^2\right] \\ &= \mathbb{E}\left[(\varepsilon^n)^2\right] \\ &= \sigma^2. \end{aligned} \tag{6.41}$$

It can be shown (see section 6.7.1) that the optimal stepsize is given by

$$\alpha_{n-1} = 1 - \frac{\sigma^2}{(1+\lambda^{n-1})\,\sigma^2 + (\beta^n)^2}, \tag{6.42}$$

where λ is computed recursively using

$$\lambda^n = \begin{cases} (\alpha_{n-1})^2, & n = 1 \\ (1-\alpha_{n-1})^2\lambda^{n-1} + (\alpha_{n-1})^2, & n > 1. \end{cases} \tag{6.43}$$

The BAKF stepsize formula enjoys several nice properties:

Stepsize policies

The BAKF stepsize formula enjoys several nice properties:

Stationary data For a sequence with a static mean, the optimal stepsizes are given by

$$\alpha_{n-1} = \frac{1}{n} \quad \forall n = 1, 2, \ldots . \tag{6.44}$$

This is the optimal stepsize for stationary data.

No noise For the case where there is no noise ($\sigma^2 = 0$), we have the following:

$$\alpha_{n-1} = 1 \quad \forall n = 1, 2, \ldots . \tag{6.45}$$

This is ideal for nonstationary data with no noise.

Bounded by $1/n$ At all times, the stepsize obeys

$$\alpha_{n-1} \geq \frac{1}{n} \quad \forall n = 1, 2, \ldots .$$

This is important since it guarantees asymptotic convergence.

Step 0. Initialization:

Step 0a. Set the baseline to its initial value, $\bar{\theta}_0$.

Step 0b. Initialize the parameters - $\bar{\beta}_0, \bar{\nu}_0$ and $\bar{\lambda}_0$.

Step 0c. Set initial stepsizes $\alpha_0 = \eta_0 = 1$, and specify the stepsize rule for η.

Step 0d. Set the iteration counter, $n = 1$.

Step 1. Obtain the new observation, $\hat{\theta}^n$.

Step 2. Smooth the baseline estimate.

$$\bar{\theta}^n = (1-\alpha_{n-1})\bar{\theta}^{n-1} + \alpha_{n-1}\hat{\theta}^n$$

Step 3. Update the following parameters:

$$\begin{aligned}
\varepsilon^n &= \bar{\theta}^{n-1} - \hat{\theta}^n, \\
\bar{\beta}^n &= (1-\eta_{n-1})\,\bar{\beta}^{n-1} + \eta_{n-1}\varepsilon^n, \\
\bar{\nu}^n &= (1-\eta_{n-1})\bar{\nu}^{n-1} + \eta_{n-1}(\varepsilon^n)^2, \\
(\bar{\sigma}^2)^n &= \frac{\bar{\nu}^n - (\bar{\beta}^n)^2}{1+\lambda^{n-1}}.
\end{aligned}$$

Step 4. Evaluate the stepsizes for the next iteration.

$$\begin{aligned}
\alpha_n &= \begin{cases} 1/(n+1) & n = 1, 2, \\ 1 - \frac{(\bar{\sigma}^2)^n}{\bar{\nu}^n}, & n > 2, \end{cases} \\
\eta_n &= \frac{a}{a+n-1}. \quad \text{Note that this gives us } \eta_1 = 1.
\end{aligned}$$

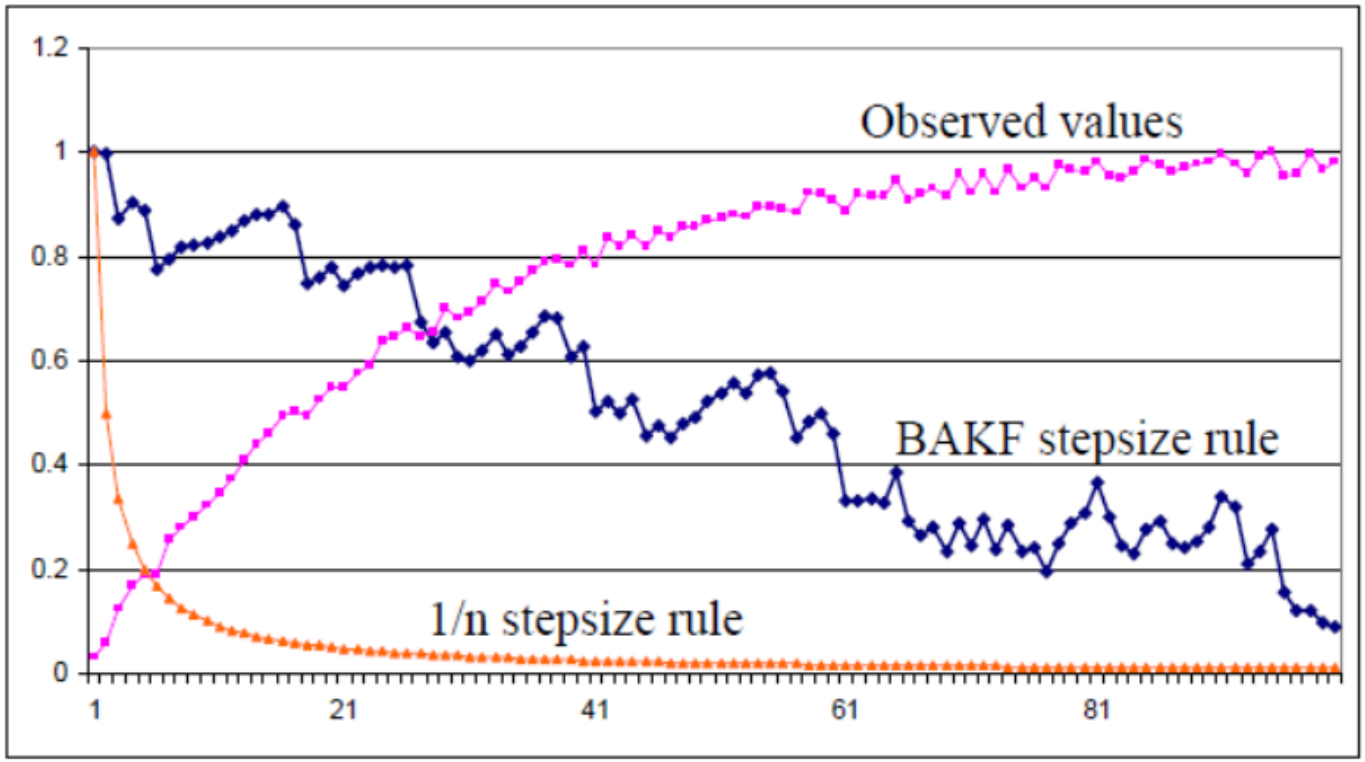
Step 5. Compute the coefficient for the variance of the smoothed estimate of the baseline.

$$\bar{\lambda}^n = (1-\alpha_{n-1})^2\bar{\lambda}^{n-1} + (\alpha_{n-1})^2.$$

Step 6. If $n < N$, then $n = n + 1$ and go to Step 1, else stop.

Stepsize policies

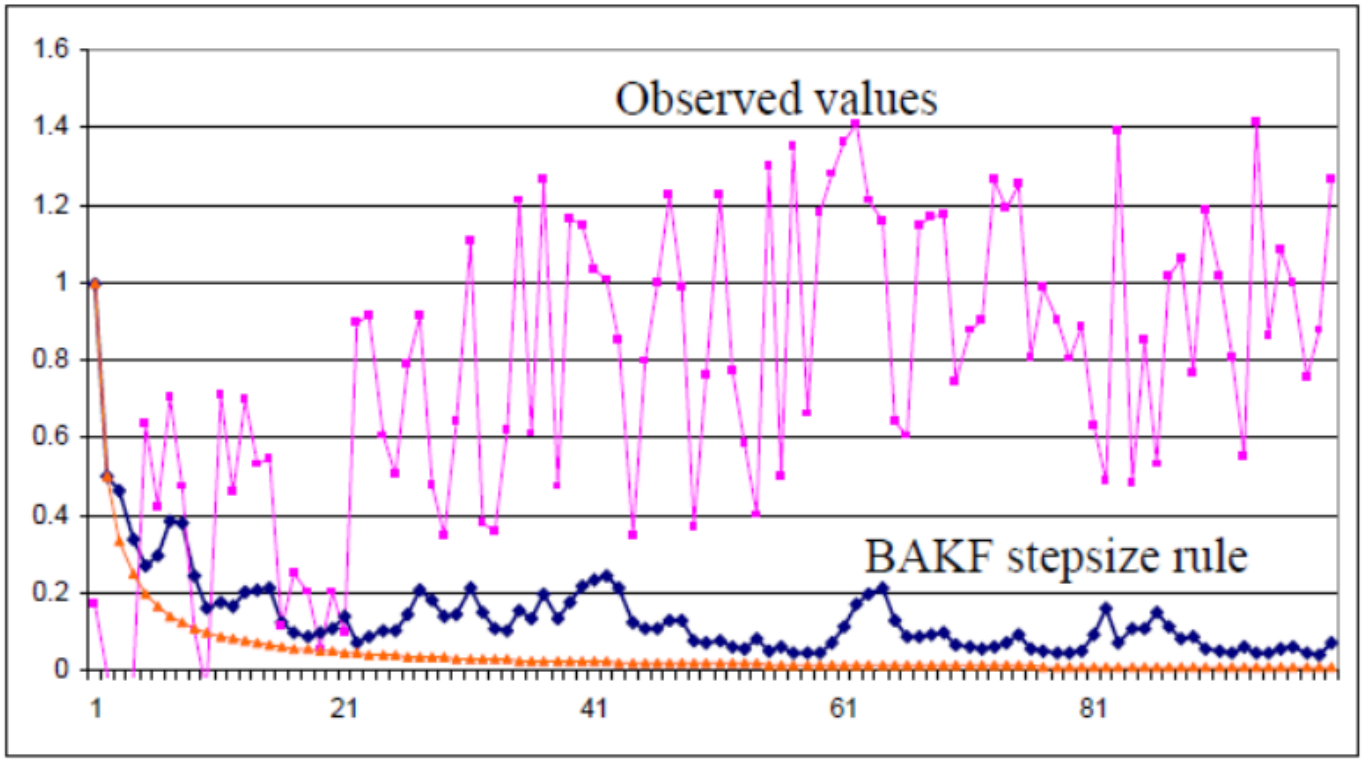
- Low noise experiment



6.9a Bias-adjusted Kalman filter for a signal with low noise.

Stepsize policies

- High-noise experiment



6.9b Bias-adjusted Kalman filter for a signal with higher noise.

Stepsize policies

- Notes:
  - » This stepsize rule was designed for approximating value functions, where the data is inherently nonstationary.
  - » It has effectively one, scale-free tunable parameter, eta, that is used to learn the bias.
  - » Provides very natural behavior:
    - Close to 1 for very low noise
    - Close to 1/n for high noise
    - Never less than 1/n (needed for convergence)
  - » It is not hard to implement, but it does require several equations, rather than the simplicity of the harmonic stepsize rule, which remains popular even in my lab.