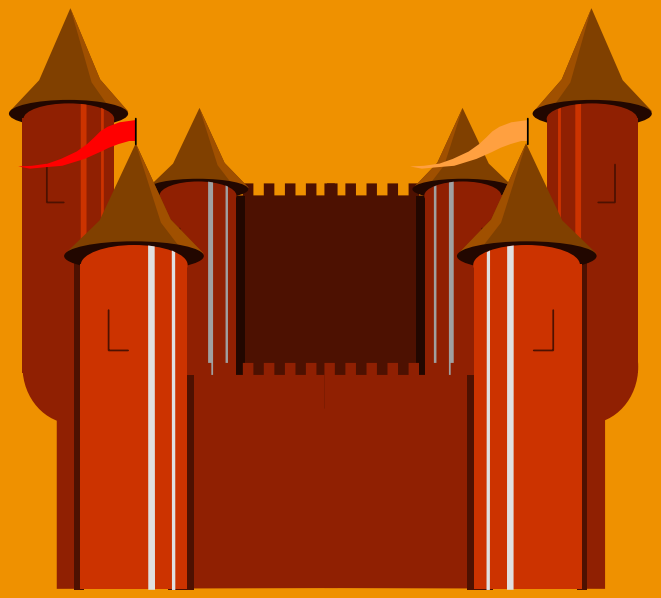
ORF 544

Stochastic Optimization and Learning

Spring, 2019



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

Week 5

Chapter 7: Derivative-free stochastic search II
Value function approximation policies
Direct lookahead policies

Designing policies

Designing policies

- Two fundamental strategies:

1) Policy search – Search over a class of functions for making decisions to optimize some metric.

$$\max_{\pi=(f\in F,\theta^f\in\Theta^f)} E\left\{\sum_{t=0}^{T} C\left(S_t, X_t^{\pi}(S_t\,|\,\theta)\right)|\, S_0\right\}$$

2) Lookahead approximations – Approximate the impact of a decision now on the future.

$$X_t^*(S_t) = \arg\max_{x_t}\left(C(S_t,x_t) + \mathbb{E}\left\{\max_{\pi\in\Pi}\left\{\mathbb{E}\sum_{t'=t+1}^{T} C(S_{t'}, X_{t'}^{\pi}(S_{t'}))\,|\,S_{t+1}\right\}|\,S_t, x_t\right\}\right)$$

Designing policies

- Policy search:

1a) Policy function approximations (PFAs) $x_t = X^{PFA}(S_t \mid \theta)$

- Lookup tables
  - "when in this state, take this action"
- Parametric functions
  - Order-up-to policies: if inventory is less than s, order up to S.
  - Affine policies - $x_t = X^{PFA}(S_t \mid \theta) = \sum_{f \in F} \theta_f \phi_f(S_t)$
  - Neural networks
- Locally/semi/non parametric
  - Requires optimizing over local regions

1b) Cost function approximations (CFAs)

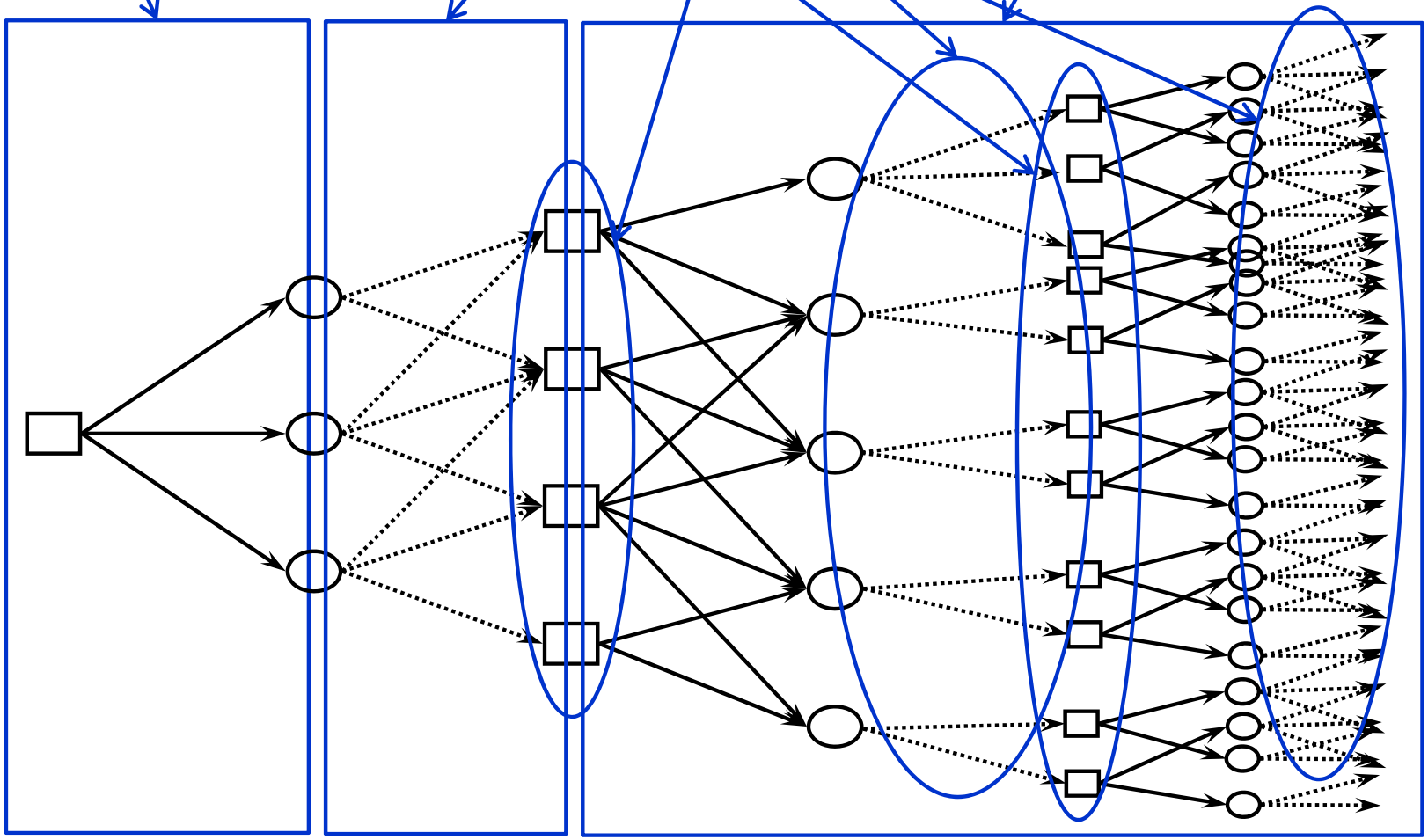
- Optimizing a deterministic model modified to handle uncertainty (buffer stocks, schedule slack)

$$X^{CFA}(S_t \mid \theta) = \arg\max_{x_t \in \bar{X}_t^{\pi}(\theta)} \bar{C}^{\pi}(S_t, x_t \mid \theta)$$

Designing policies

- The ultimate lookahead policy is optimal

$$X_t^*(S_t) = \arg\max_{x_t} \left(C(S_t, x_t) + \mathbb{E}\left\{ \max_{\pi \in \Pi} \left\{ \mathbb{E} \sum_{t'=t+1}^{T} C(S_{t'}, X_{t'}^{\pi}(S_{t'})) \mid S_{t+1} \right\} \mid S_t, x_t \right\} \right)$$



Designing policies

- Lookahead approximations – Approximate the impact of a decision now on the future:
  - » An optimal policy (based on looking ahead):

$$X_t^*(S_t) = \arg\max_{x_t} \left(C(S_t, x_t) + \mathbb{E}\left\{ \max_{\pi \in \Pi} \left\{ \mathbb{E} \sum_{t'=t+1}^{T} C(S_{t'}, X_{t'}^{\pi}(S_{t'})) \mid S_{t+1} \right\} \mid S_t, x_t \right\} \right)$$

2a) Approximating the value of being in a downstream state using machine learning ("value function approximations")

$$X_t^*(S_t) = \arg\max_{x_t} \left(C(S_t, x_t) + \mathbb{E}\left\{ V_{t+1}(S_{t+1}) \mid S_t, x_t \right\} \right)$$

$$X_t^{VFA}(S_t) = \arg\max_{x_t} \left(C(S_t, x_t) + \mathbb{E}\left\{ \bar{V}_{t+1}(S_{t+1}) \mid S_t, x_t \right\} \right)$$

$$= \arg\max_{x_t} \left(C(S_t, x_t) + \bar{V}_t^x(S_t^x) \right)$$

Designing policies

- The ultimate lookahead policy is optimal

$$X_t^*(S_t) = \arg\max_{x_t}\left(C(S_t,x_t) + \mathbb{E}\left\{\max_{\pi\in\Pi}\left\{\mathbb{E}\sum_{t'=t+1}^{T} C(S_{t'}, X_{t'}^{\pi}(S_{t'}))\,|\,S_{t+1}\right\}|\,S_t,x_t\right\}\right)$$

» 2b) Instead, we have to solve an approximation called the *lookahead model:*

$$X_t^*(S_t) = \arg\max_{x_t}\left(C(S_t,x_t) + \tilde{\mathbb{E}}\left\{\max_{\tilde{\pi}\in\tilde{\Pi}}\left\{\tilde{\mathbb{E}}\sum_{t'=t+1}^{t+H} C(\tilde{S}_{tt'}, \tilde{X}_{tt'}^{\pi}(\tilde{S}_{tt'}))\,|\,\tilde{S}_{t,t+1}\right\}|\,S_t,x_t\right\}\right)$$

» *A lookahead policy* works by approximating the *lookahead model*.

Designing policies

- Types of lookahead approximations
  - » One-step lookahead – Widely used in pure learning policies:
    - Bayes greedy/naïve Bayes
    - Expected improvement
    - Value of information (knowledge gradient)
  - » Multi-step lookahead
    - Deterministic lookahead, also known as model predictive control, rolling horizon procedure
    - Stochastic lookahead:
      - Two-stage (widely used in stochastic linear programming)
      - Multistage
        - » Monte carlo tree search (MCTS) for discrete action spaces
        - » Multistage scenario trees (stochastic linear programming) – typically not tractable.

Four (meta)classes of policies

Policy search

1) **Policy function approximations (PFAs)**

» Lookup tables, rules, parametric/nonparametric functions

2) **Cost function approximation (CFAs)**

» $X^{CFA}(S_t \mid \theta) = \arg\max_{x_t \in \bar{X}_t^{\pi}(\theta)} \bar{C}^{\pi}(S_t, x_t \mid \theta)$

Lookahead approximations

3) **Policies based on value function approximations (VFAs)**

» $X_t^{VFA}(S_t) = \arg\max_{x_t} \left(C(S_t, x_t) + \bar{V}_t^{x} \left(S_t^{x}(S_t, x_t) \right) \right)$

4) **Direct lookahead policies (DLAs)**

» *Deterministic lookahead/rolling horizon proc./model predictive control*

$$X_t^{LA-D}(S_t) = \arg\max_{\tilde{x}_{tt}, \ldots, \tilde{x}_{t,t+H}} C(\tilde{S}_{tt}, \tilde{x}_{tt}) + \sum_{t'=t+1}^{T} C(\tilde{S}_{tt'}, \tilde{x}_{tt'})$$

» *Chance constrained programming*

$$P[A_t x_t \le f(W)] \le 1 - \delta$$

» *Stochastic lookahead /stochastic prog/Monte Carlo tree search*

$$X_t^{LA-S}(S_t) = \arg\max_{\tilde{x}_{tt}, \tilde{x}_{t,t+1}, \ldots, \tilde{x}_{t,t+T}} C(\tilde{S}_{tt}, \tilde{x}_{tt}) + \sum_{\tilde{\omega} \in \tilde{\Omega}_t} p(\tilde{\omega}) \sum_{t'=t+1}^{T} C(\tilde{S}_{tt'}(\tilde{\omega}), \tilde{x}_{tt'}(\tilde{\omega}))$$

» *"Robust optimization"*

$$X_t^{LA-RO}(S_t) = \arg\max_{\tilde{x}_{tt}, \ldots, \tilde{x}_{t,t+H}} \min_{w \in W_t(\theta)} C(\tilde{S}_{tt}, \tilde{x}_{tt}) + \sum_{t'=t+1}^{T} C(\tilde{S}_{tt'}(w), \tilde{x}_{tt'}(w))$$

Lookahead policies:
Value function approximations

Multiarmed bandit problems

- Multi-armed bandit problems
  - » We do not know the expected winnings from each slot machine ("arm").
  - » We need to find a policy $X^{\pi}(S^n)$ for playing machine x that maximizes:

$$\max_{\pi} \mathbb{E}\sum_{n=0}^{N-1} W_{x^n}^{n+1}$$

where

$$S^n = \text{State of knowledge}$$

$$x^n = X^{\pi}(S^n)$$

$$W^{n+1} = \text{"winnings"}$$

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 by 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

Multiarmed bandit problems

- Bandit problems

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

Multiarmed bandit problems

7.7 GITTINS INDICES FOR LEARNING WITH CUMULATIVE REWARDS

The most basic learning problem with cumulative rewards is the multiarmed bandit problem, which we first saw in chapter 2. The problem is to try to learn the value of $\mu_x = \mathbb{E}F(x, W)$ for discrete x by trying different alternatives and observing $W_x^n = \mu_x + \varepsilon_x^n$. The challenge is to find a policy $X^\pi(S^n)$ that solves

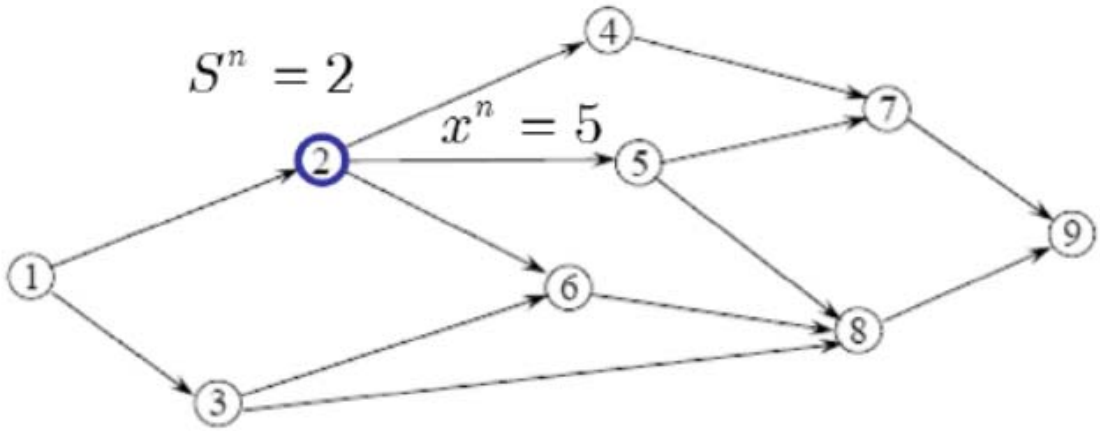
$$\max_\pi \mathbb{E} \sum_{n=0}^{N-1} W_{X^\pi(S^n)}^{n+1},$$

where $x^n = X^\pi(S^n)$.

The first implementable and provably optimal policy for this problem class was developed by John Gittins in 1974, with a strategy that became known as *Gittins indices*. The idea is to compute a single value (called an index) for each alternative x, and then evaluate the alternative with the highest index. While the study of learning problems has its roots in the 1950's, it was the introduction of Gittins indices that launched much of the modern research into efficient learning.

Multiarmed bandit problems

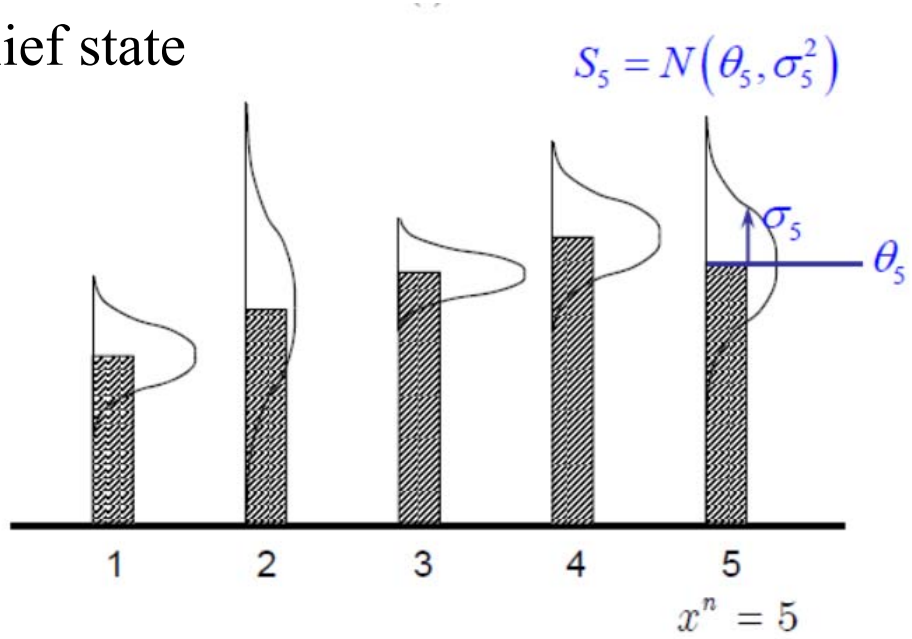
- Bellman's equation
  - » A graph



$$V(S) = \max_x \left(C(S,x) + V(S^M(S,x))\right).$$

Multiarmed bandit problems

- Bellman's equation
  - » A belief state



  - » Bellman equation (state = node)

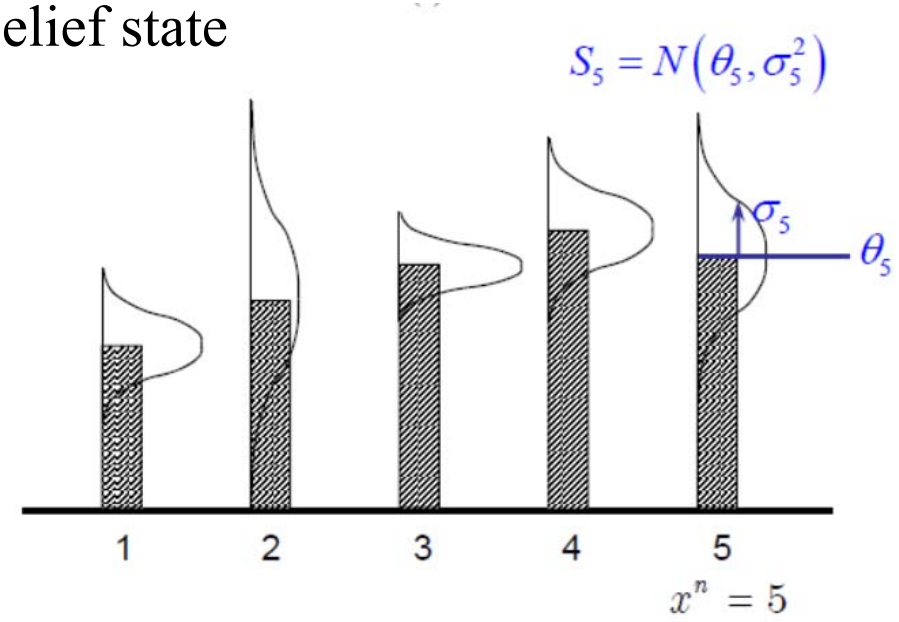
$$V(S) = \max_x \left(C(S,x) + \mathbb{E}_W V(S^M(S,x,W))\right).$$

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

  - » Can only solve this in very special cases (and not with normally distributed beliefs). 5 alternatives -> 10-dimensional continuous state.

Multiarmed bandit problems

- Bellman's equation for online and offline learning
  - » A belief state



  - » Online learning

$$V(S) = \max_x \left(C(S,x) + \mathbb{E}_W V(S^M(S,x,W))\right). \qquad (3.7)$$

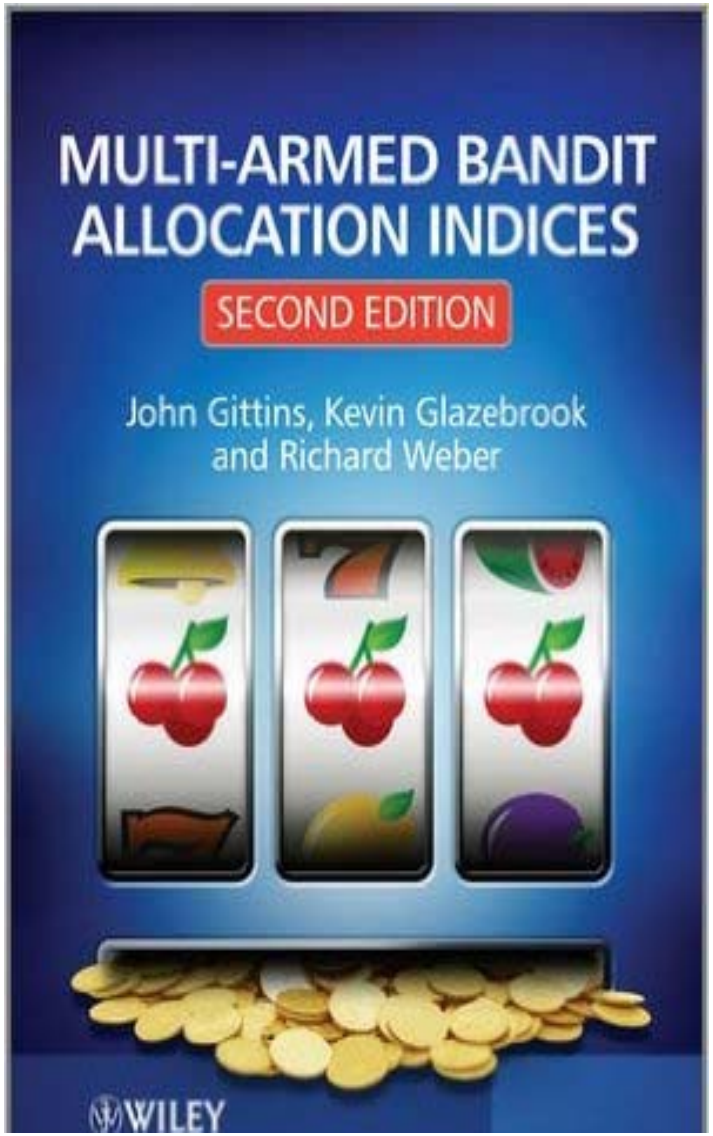
  - » Offline learning

$$V(S) = \max_x \left(\qquad \mathbb{E}_W V(S^M(S,x,W))\right). \qquad (3.7)$$

Gittins indices

Gittins indices

- Historical notes:
  - » 1970 – Degroot gives Bellman equation for optimal learning (but cannot be computed).
  - » 1974 – Gittins publishes a decomposition method that breaks the "curse of dimensionality," requiring that we solve a DP for one "arm" at a time.
  - » 1985 – Lai and Robbins introduce upper confidence bounding which takes off in the computer science community.



Gittins indices

5.1 THE THEORY AND PRACTICE OF 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 state of knowledge 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

$$V(S^n, r) = \max\left[r + \gamma V(S^n, r), \mathbb{E}\left\{W^{n+1} + \gamma\, V(S^{n+1}, r)\middle|\, S^n\right\}\right]. \quad (5.6)$$

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 state of knowledge. 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.

Gittins indices

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

$$V(S^n, r) = \max \left[\frac{r}{1-\gamma}, \mathbb{E}\left\{ W^{n+1} + \gamma\, V(S^{n+1}, r) \middle| S^n \right\} \right], \tag{5.7}$$

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

$$\frac{r}{1-\gamma} = \mathbb{E}\left\{ W^{n+1} + \gamma\, V(S^{n+1}, r) \middle| S^n \right\} \tag{5.8}$$

for r. The Gittins index $I^{Gitt,n}$ is the particular value of r that solves (5.8). 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

5.1.1 Gittins indices in the beta-Bernoulli model

The Gittins recursion in (5.7) 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 (5.7) 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 knowledge state for a single slot machine is simply $S^n = (\alpha^n, \beta^n)$. Consequently,

$$\begin{aligned} \mathbb{E}\left(W^{n+1} \,|\, S^n\right) &= \mathbb{E}\left[\mathbb{E}\left(W^{n+1} \,|\, S^n, \rho\right) \,|\, S^n\right] \\ &= \mathbb{E}\left(\rho \,|\, S^n\right) \\ &= \frac{\alpha^n}{\alpha^n + \beta^n}. \end{aligned}$$

Gittins indices

- Beta distribution

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 2.2 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

$$\mathbb{E}(\rho) = \frac{\alpha^0}{\alpha^0 + \beta^0}. \tag{2.33}$$

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.

Gittins indices

- Beta distributions

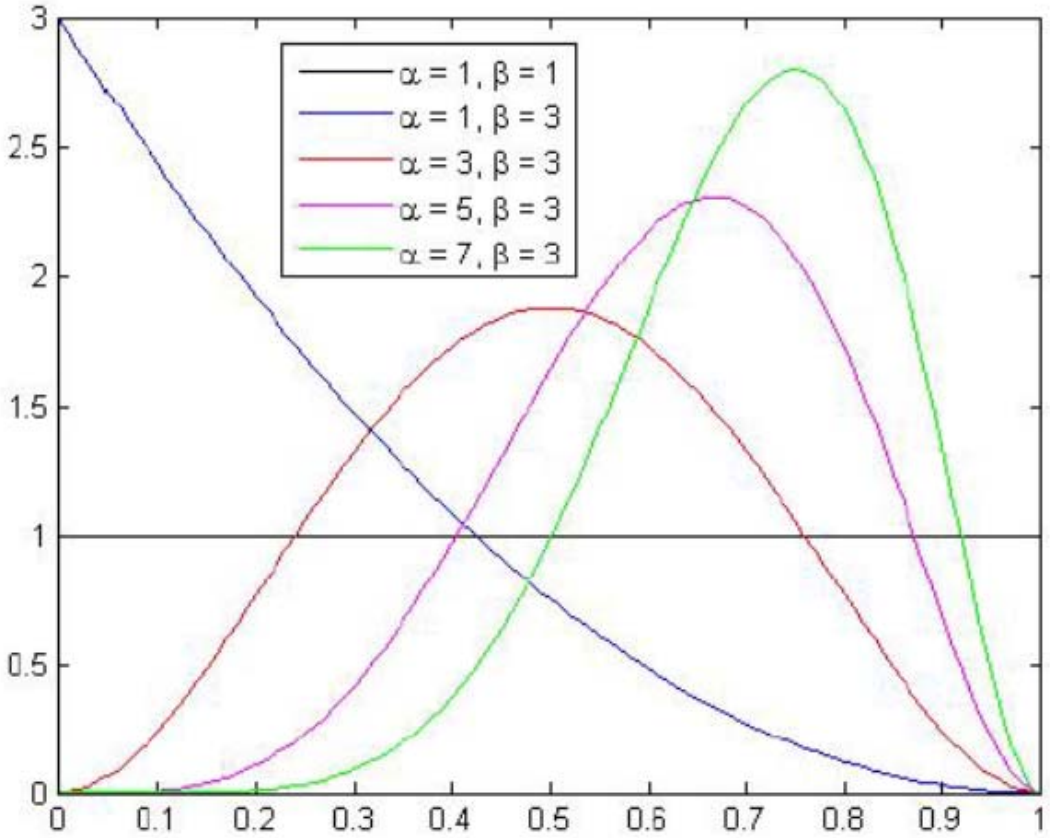


Figure 2.2 Illustration of a family of beta distributions.

Gittins indices

Then, writing $V(S^n, r)$ as $V(\alpha^n, \beta^n, r)$, we obtain

$$\mathbb{E}\left\{W^{n+1} + \gamma V(S^{n+1}, r) \middle| S^n\right\} = \frac{\alpha^n}{\alpha^n + \beta^n} + \gamma \frac{\alpha^n}{\alpha^n + \beta^n} V(\alpha^n + 1, \beta^n, r) + \gamma \frac{\beta^n}{\alpha^n + \beta^n} V(\alpha^n, \beta^n + 1, r). \quad (5.9)$$

For fixed α and β, the quantity $V(\alpha, \beta, r)$ is a constant. However, if the observation W^{n+1} is a success, we will transition to the knowledge 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 (5.9), it becomes clear why Gittins indices are difficult to compute exactly. For any value of r and any α, β, we need to know $V(\alpha + 1, \beta, r)$ as well as $V(\alpha, \beta + 1, r)$ before we can compute $V(\alpha, \beta, r)$. 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 $V(\alpha, \beta, r)$ approximately. For all α and β such that $\alpha + \beta$ is "large enough," we could assume that $V(\alpha, \beta, r)$ is equal to some value, perhaps zero. Then, a backwards recursion using these terminal values would give us approximations of $V(\alpha, \beta, r)$ 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

$$V(\alpha, \beta, r) \approx V(\alpha + 1, \beta, r) \approx V(\alpha, \beta + 1, r).$$

Then, we can combine (5.7) with (5.9) to approximate the Gittins recursion as

$$V(\alpha, \beta, r) = \max\left[\frac{r}{1-\gamma}, \frac{\alpha}{\alpha+\beta} + \gamma V(\alpha, \beta, r)\right]. \tag{5.10}$$

In this case, it can be shown that (5.10) has the solution

$$V(\alpha, \beta, r) = \frac{1}{1-\gamma} \max\left(r, \frac{\alpha}{\alpha+\beta}\right),$$

and the solution to (5.8) in this case is simply

$$r^* = \frac{\alpha}{\alpha+\beta}.$$

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 (5.7) and (5.9) 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

Gittins indices

- General theory of Gittins indices

but μ could be a vector of parameters that characterizes the distribution of $\hat{F}$). μ^n represents our state variable (that is, our current estimate of the mean). Let μ^n be our current estimate of the reward we would receive from playing the machine given what we know after n plays. The optimality equations can now be written

$$V(\mu^n, \rho) = \max\left[\rho + \gamma V(\mu^n, \rho), \mu^n + \gamma \mathbb{E}\left\{V(\mu^{n+1}, \rho) \middle| \mu^n\right\}\right], \quad (7.19)$$

where we have written the value function to express the dependence on ρ.

Since we have an infinite horizon problem, the value function must satisfy the optimality equations

$$V(\mu, \rho) = \max\left[\rho + \gamma V(\mu, \rho), \mu + \gamma \mathbb{E}\left\{V(\mu', \rho) \middle| \theta\right\}\right],$$

where θ' is defined by by our Bayesian updating formula in equation (7.17).

It can be shown that if we choose to stop sampling in iteration n and accept the fixed payment ρ, then that is the optimal strategy for all future rounds. This means that starting at iteration n, our optimal future payoff (once we have decided to accept the fixed payment) is

$$\begin{aligned} V(\mu, \rho) &= \rho + \gamma\rho + \gamma^2\rho + \cdots \\ &= \frac{\rho}{1-\gamma}, \end{aligned}$$

which means that we can write our optimality recursion in the form

$$V(\mu^n, \rho) = \max\left[\frac{\rho}{1-\gamma}, \mu^n + \gamma \mathbb{E}\left\{V(\mu^{n+1}, \rho) \middle| \mu^n\right\}\right]. \quad (7.20)$$

Gittins indices

Now for the magic of Gittins indices. Let ν be the value of ρ which makes the two terms in the brackets in (7.20) equal. That is,

$$\frac{\nu}{1-\gamma} = \mu + \gamma \mathbb{E}\left\{V(\mu', \nu) | \mu\right\}. \tag{7.21}$$

Let $\nu^{Gitt}(\mu, \sigma, \sigma_W, \gamma)$ be the solution of (7.21). The optimal solution depends on the current estimate of the mean, μ, its variance σ^2, the variance of our measurements σ_W^2, and the discount factor γ. (For notational simplicity, we are assuming that the measurement noise σ_W^2 is independent of the action a, but this assumption is easily relaxed.) Next assume that we have a family of slot machines $\mathcal{A}$, and let $\nu_x^{Gitt,n}(\mu_x^n, \bar{\sigma}_x^n, \sigma_W, \gamma)$ be the value of ν that we compute for each slot machine $a \in \mathcal{A}$. An optimal policy for selecting slot machines is to choose the slot machine with the highest value for $\nu_x^{Gitt,n}(\mu_x^n, \bar{\sigma}_x^n, \sigma_W, \gamma)$. Such policies are known as *index policies*, which refers to the property that the parameter $\nu_x^{Gitt,n}(\mu_x^n, \bar{\sigma}_x^n, \sigma_W, \gamma)$ for alternative a depends only on the characteristics of alternative a. For this problem, the parameters $\nu_x^{Gitt,n}(\mu_x^n, \bar{\sigma}_x^n, \sigma_W, \gamma)$ are called Gittins indices.

Gittins indices

- Gittins indices
  - » The main result (for normally distributed rewards):

$$\nu^{Gitt,n}(\mu^n, \bar{\sigma}^n, \sigma_W, \gamma) = \mu + \Gamma(\frac{\bar{\sigma}^n}{\sigma_W}, \gamma)\sigma_W,$$

where

$$\Gamma(\frac{\bar{\sigma}^n}{\sigma_W}, \gamma) = \nu^{Gitt,n}(0, \sigma, 1, \gamma)$$

is a "standard normal Gittins index" for problems with mean 0 and variance 1. Note that $\bar{\sigma}^n/\sigma_W$ increases with n, and that $\Gamma(\frac{\bar{\sigma}^n}{\sigma_W}, \gamma)$ decreases toward zero as $\bar{\sigma}^n/\sigma_W$ increases. As $n \to \infty$, $\nu^{Gitt,n}(\mu^n, \bar{\sigma}^n, \sigma_W, \gamma) \to \mu^n$.

  - » While these are computable, they are hard to compute.
  - » Even if we could, Gittins indices are *not* optimal in practical applications.
  - » Has not attracted any attention from industry.

Gittins indices

- Approximations for Gittins indices:

Lacking standard software libraries for computing Gittins indices, researchers have developed simple approximations. As of this writing, the most recent of these works as follows. First, it is possible to show that

$$\Gamma(s,\gamma) = \sqrt{-\log\gamma}\cdot b\left(-\frac{s^2}{\log\gamma}\right). \tag{7.22}$$

A good approximation of $b(s)$, which we denote by $\tilde{b}(s)$, is given by

$$\tilde{b}(s) = \begin{cases} \frac{s}{\sqrt{2}} & s \le \frac{1}{7}, \\ e^{-0.02645(\log s)^2 + 0.89106\log s - 0.4873} & \frac{1}{7} < s \le 100, \\ \sqrt{s}\,(2\log s - \log\log s - \log 16\pi)^{\frac{1}{2}} & s > 100. \end{cases}$$

Thus, the approximate version of (7.22) is

$$\nu^{Gitt,n}(\mu,\sigma,\sigma_W,\gamma) \approx \mu^n + \sigma_W\sqrt{-\log\gamma}\cdot\tilde{b}\left(-\frac{\bar{\sigma}^{2,n}}{\sigma_W^2\log\gamma}\right). \tag{7.23}$$

Gittins indices

- Gittins indices for normal-normal model

$G(1/\sqrt{k}, \gamma)$ is increasing in γ

$G(1/\sqrt{k}, \gamma)$ is decreasing in k

	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).

Lookahead policies: One-period lookahead

The knowledge gradient

The knowledge gradient

- Direct lookahead policies:
  - » One-step lookaheads
    - Knowledge gradient
    - Expected improvement
  - » Limited multistep lookahead
    - Repeated lookahead for a single choice
  - » Full multistep lookahead
    - Full tree search (multiple decisions and outcomes each time period)

The knowledge gradient

- Expensive experiments
  - 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.
  - 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.

The knowledge gradient

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 state of knowledge as $S^n = (\bar{\mu}_x^n, \beta_x^n)_{x \in \mathcal{X}}$. If we stop measuring now, we would pick the best option, which we represent by

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

The value of being in state S^n is then given by

$$V^n(S^n) = \bar{\mu}_{x^n}^n.$$

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

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

$$\mathbb{E}\{V^{n+1}(S^{n+1}(x))|S^n\} \quad = \quad \mathbb{E}_\mu \mathbb{E}_{W|\mu}\{V^{n+1}(S^{n+1}(x))|S^n\},$$

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

The knowledge gradient

- The knowledge gradient is a one-period lookahead that maximizes the value of information:

$$\nu_x^{KG,n} = E\left\{\max_y F(y, B^{n+1}(x))\right\} - \max_y F(y, B^n)$$

The knowledge gradient

- The knowledge gradient is a one-period lookahead that maximizes the value of information:

$$\nu_x^{KG,n} = E\left\{\max_y F(y, B^{n+1}(x))\right\} - \max_y F(y, B^n)$$

Current belief state

The knowledge gradient

- The knowledge gradient is a one-period lookahead that maximizes the value of information:

$$\nu_x^{KG,n} = E\left\{\max_y F(y, B^{n+1}(x))\right\} - \max_y F(y, B^n)$$

Choosing the best design given what we know now.

The knowledge gradient

- The knowledge gradient is a one-period lookahead that maximizes the value of information:

$$\nu_x^{KG,n} = E\left\{\max_y F(y, B^{n+1}(x))\right\} - \max_y F(y, B^n)$$

Proposed experiment

The knowledge gradient

- The knowledge gradient is a one-period lookahead that maximizes the value of information:

$$\nu_x^{KG,n} = E\left\{\max_y F(y, B^{n+1}(x))\right\} - \max_y F(y, B^n)$$

Updated parameter estimates after running experiment with density x.

The knowledge gradient

- The knowledge gradient is a one-period lookahead that maximizes the value of information:

$$\nu_x^{KG,n} = E\left\{\max_y F(y, B^{n+1}(x))\right\} - \max_y F(y, B^n)$$

Finding the new design with our new knowledge (but without knowing the outcome of the experiment)

The knowledge gradient

- The knowledge gradient is a one-period lookahead that maximizes the value of information:

$$\nu_x^{KG,n} = E\left\{\max_y F(y, B^{n+1}(x))\right\} - \max_y F(y, B^n)$$

Averaging over the possible outcomes of the experiment (and our different beliefs about parameters)

The knowledge gradient

- Knowledge gradient

$$\nu_x^{KG,n} = \mathbb{E}_\mu \mathbb{E}_{W|\mu} \left\{ \max_{x'} \bar{\mu}_{x'}^{n+1}(x) \,|\, S^n \right\} - \max_{x'} \bar{\mu}_{x'}^n$$

» Discuss $\bar{\mu}_{x'}^{n+1}(x)$ as the estimated updated belief if we run experiment x .

» Think of this estimate as peeking into the future.

» We did this before in the decision tree. In fact, we did not just look one step ahead – we looked two steps (in our baseball example).

» Think about how we might estimate the expectations using simulation:

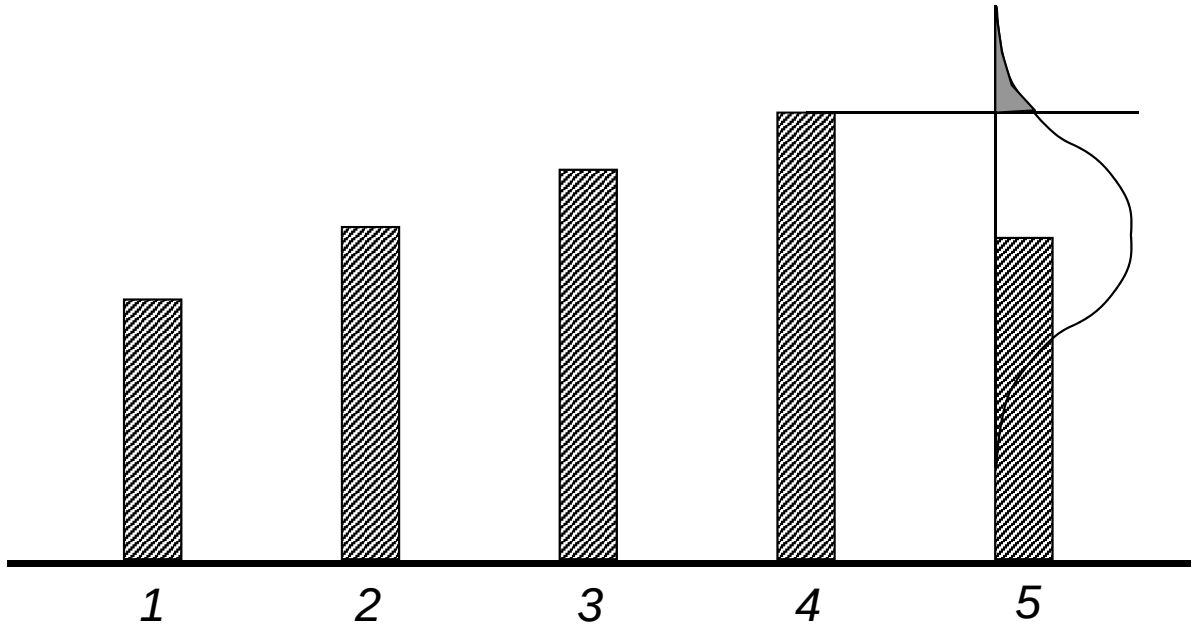
» Review how means are updated given estimates in S^n (which contains $\bar{\mu}_x^n$) and the observation $W_x^{l|k}$ for a truth μ_x^k and noise ϵ^l

$$\nu_x^{KG,n} = \frac{1}{K}\sum_{k=1}^{K}\frac{1}{L}\sum_{l=1}^{L} \max_{x'} \bar{\mu}_{x'}^{n+1}(x \,|\, S^n, W_x^{l|k} = \mu_x^k + \varepsilon^l) - \max_{x'} \bar{\mu}_{x'}^n$$

» Remember that μ_x^k is a possible *true value* of μ, not the estimate. We have an estimate, but the random outcome *W* comes from the *truth*.

The knowledge gradient

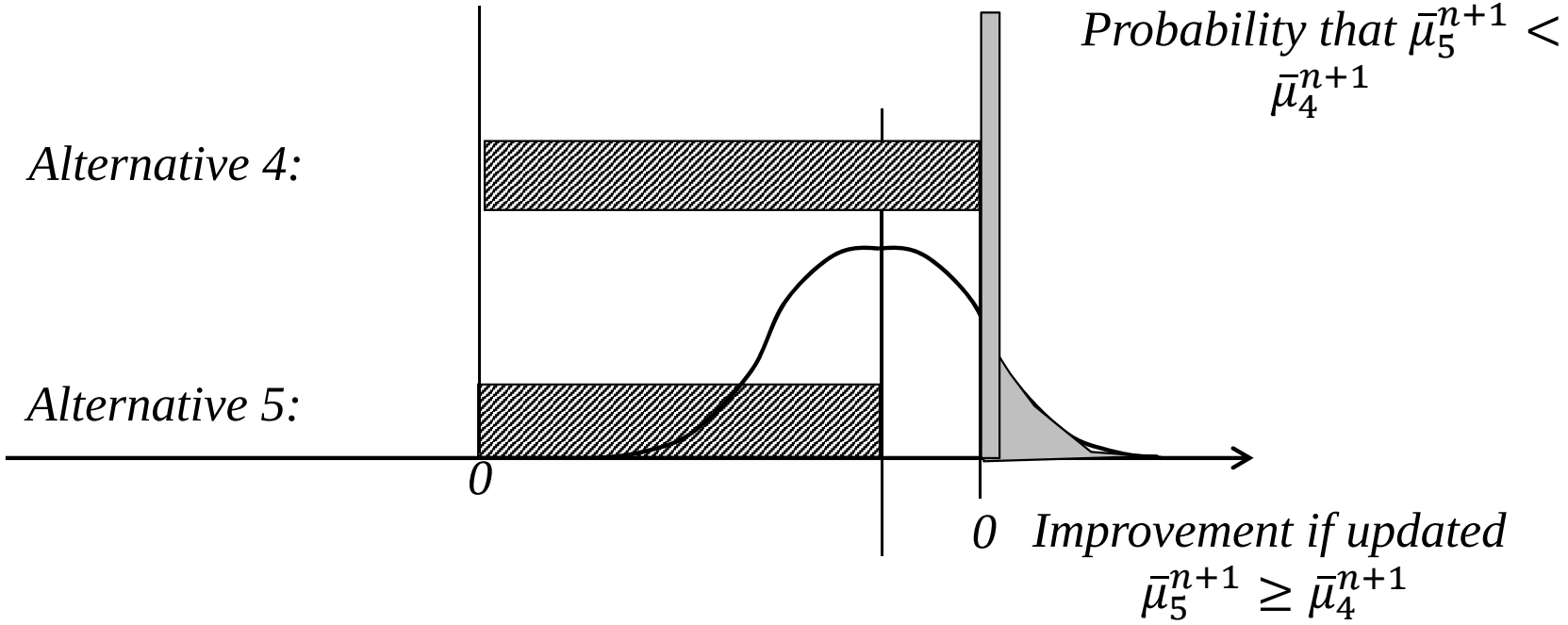
- Knowledge gradient policy



» We want to measure the weighted area under the curve for option 5 that is over the value for option 4.

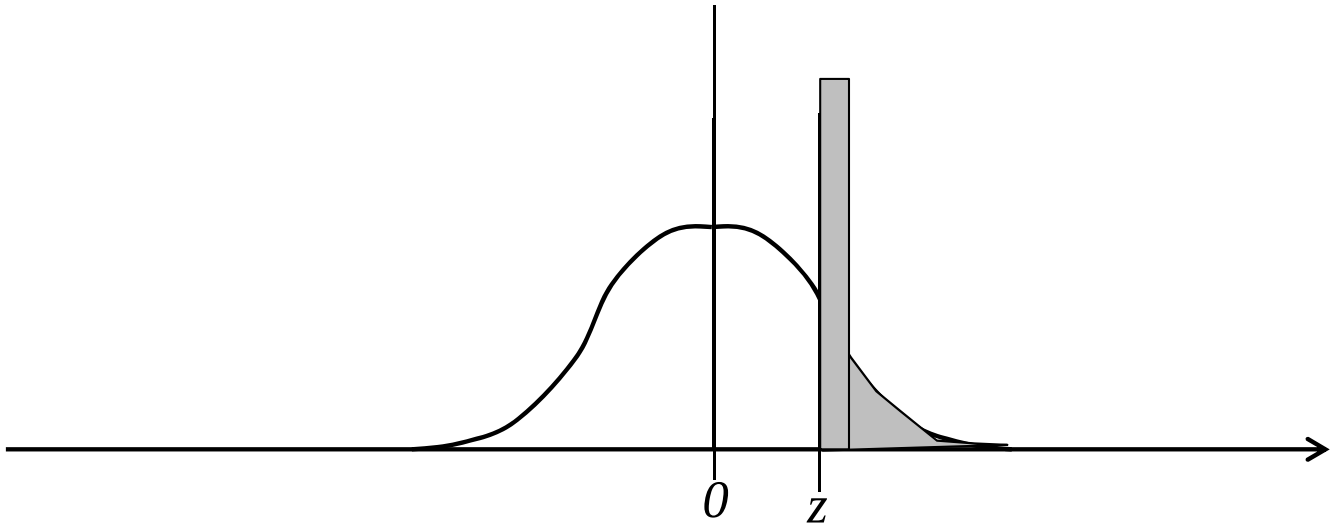
Knowledge gradient

- Value of information



The knowledge gradient is the expected improvement if we observe alternative 5. We have to capture how much better alternative 5 ends up looking relative to the current best, alternative 4. Here, "0" means "no better than alternative 4".

- Conditional expectation of standard normal random variable
  - » Let $Z \sim N(0,1)$. We want $\mathbb{E}\{Z|Z \geq z\}$.



$$\mathbb{E}[Z \mid Z \geq z\} = f(z) = z\Phi(z) + \phi(z)$$

$\Phi(z)$ = Cumulative standard normal distribution

$\phi(z)$ = Standard normal density

The knowledge gradient

- Derivation
  - » Notation

$$\beta_x^n = \text{Precision (inverse variance) of our estimate } \theta_x^n \text{ of the value of } x.$$
$$\beta^W = \text{Precision of the measurement noise } (=1/\sigma_W^2)$$
$$w_x^{n+1} = \text{Measurement of } x \text{ in iteration } n+1 \text{ (unknown at } n)$$

  - » We update the precision using

$$\beta_x^{n+1} = \beta_x^n + \beta^W$$

  - » In terms of the variance, this is the same as

$$\left(\sigma_x^{2,n+1}\right)^{-1} = \left(\sigma_x^{2,n}\right)^{-1} + \left(\sigma_W^2\right)^{-1}$$
$$\sigma_x^{2,n+1} = \frac{\sigma_x^{2,n}}{1+\sigma_x^{2,n}/\sigma_W^2}$$

The knowledge gradient

- Derivation
  - » The change in variance can be found to be

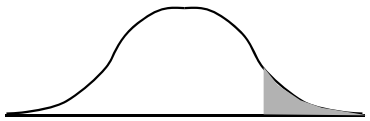
$$\tilde{\sigma}_x^{2,n} = Var\left[\bar{\mu}_x^{n+1} - \bar{\mu}_x^n \mid S^n\right]$$
$$= \sigma_x^{2,n} - \sigma_x^{2,n+1}$$
$$= \frac{\sigma_x^{2,n}}{1 + \sigma_W^2 / \sigma_x^{2,n}}$$

  - » Next compute the *normalized influence*:

$$\zeta_x^n = -\left|\frac{\bar{\mu}_x^n - \max_{x' \neq x} \bar{\mu}_{x'}^n}{\tilde{\sigma}_x^n}\right|.$$

  - » Let $f(\zeta) = \zeta\Phi(\zeta) + \phi(\zeta)$ $\quad \Phi(\zeta) =$ Cumulative standard normal distribution

$\phi(\zeta) =$ Standard normal density



  - » Knowledge gradient is computed using

$$\nu_x^{KG} = \tilde{\sigma}_x^n f\left(\zeta_x^n\right)$$

51

The knowledge gradient

- The normal distribution

We next compute

$$f(\zeta) = \zeta\Phi(\zeta) + \phi(\zeta), \tag{5.9}$$

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

$$\Phi(\zeta) = \int_{-\infty}^{\zeta} \phi(x)dx.$$

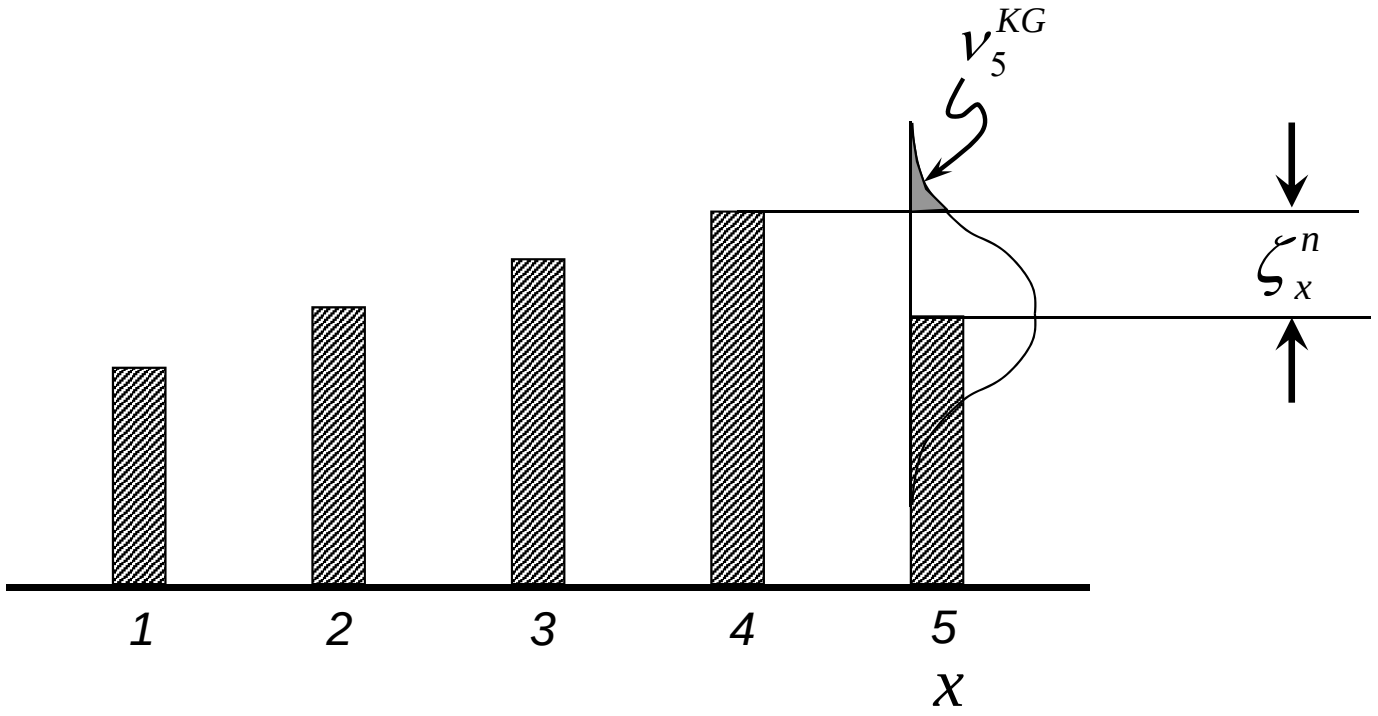
- Computing the cdf
  - » Matlab – normcdf(x,mu,sigma)
  - » Excel – normdist(x,mu,sigma,prob) z = standard normal

The knowledge gradient

- Knowledge gradient

$$\zeta_x^n = -\left|\frac{\bar{\mu}_x^n - \max_{x' \neq x} \bar{\mu}_{x'}^n}{\tilde{\sigma}_x^n}\right|$$

= Normalized distance to best (or second best)



53

The knowledge gradient

- KG calculations

Sigma^n		Decision	&	μ^n	&	β^n	&	β^{n+1}	&	$\sigmatilde$	&	$max_x' x'$	&	ζ	&	$f(z)$	&	$\nu^{KG)_x$
5.00		1	&	3.0	&	0.0400	&	1.0400	&	4.9029	&	5	&	-0.4079	&	0.2277	&	1.1165
8.00		2	&	4.0	&	0.0156	&	1.0156	&	7.9382	&	5	&	-0.1260	&	0.3391	&	2.6920
8.00		3	&	5.0	&	0.0156	&	1.0156	&	7.9382	&	4.5	&	-0.0630	&	0.3682	&	2.9232
9.00		4	&	4.5	&	0.0123	&	1.0123	&	8.9450	&	5	&	-0.0559	&	0.3716	&	3.3241
10.00		5	&	3.5	&	0.0100	&	1.0100	&	9.9504	&	5	&	-0.1507	&	0.3281	&	3.2646

» To link to spreadsheet, click here.

The knowledge gradient

Properties

Properties of the knowledge gradient

- Property 1: The knowledge gradient is always positive, $\nu_x^{KG,n} \geq 0$ 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.

56

Properties of the knowledge gradient

- 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 (3.9)) and interval estimation (θ^{IE} in equation (3.13)) 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.

57

Properties of the knowledge gradient

- Compare two policies:
  - » Interval estimation

$$X^{IE}(S^n|\theta^{IE}) = argmax_x(\bar{\mu}_x^n + \theta^{IE}\bar{\sigma}_x^n)$$

  - » Knowledge gradient

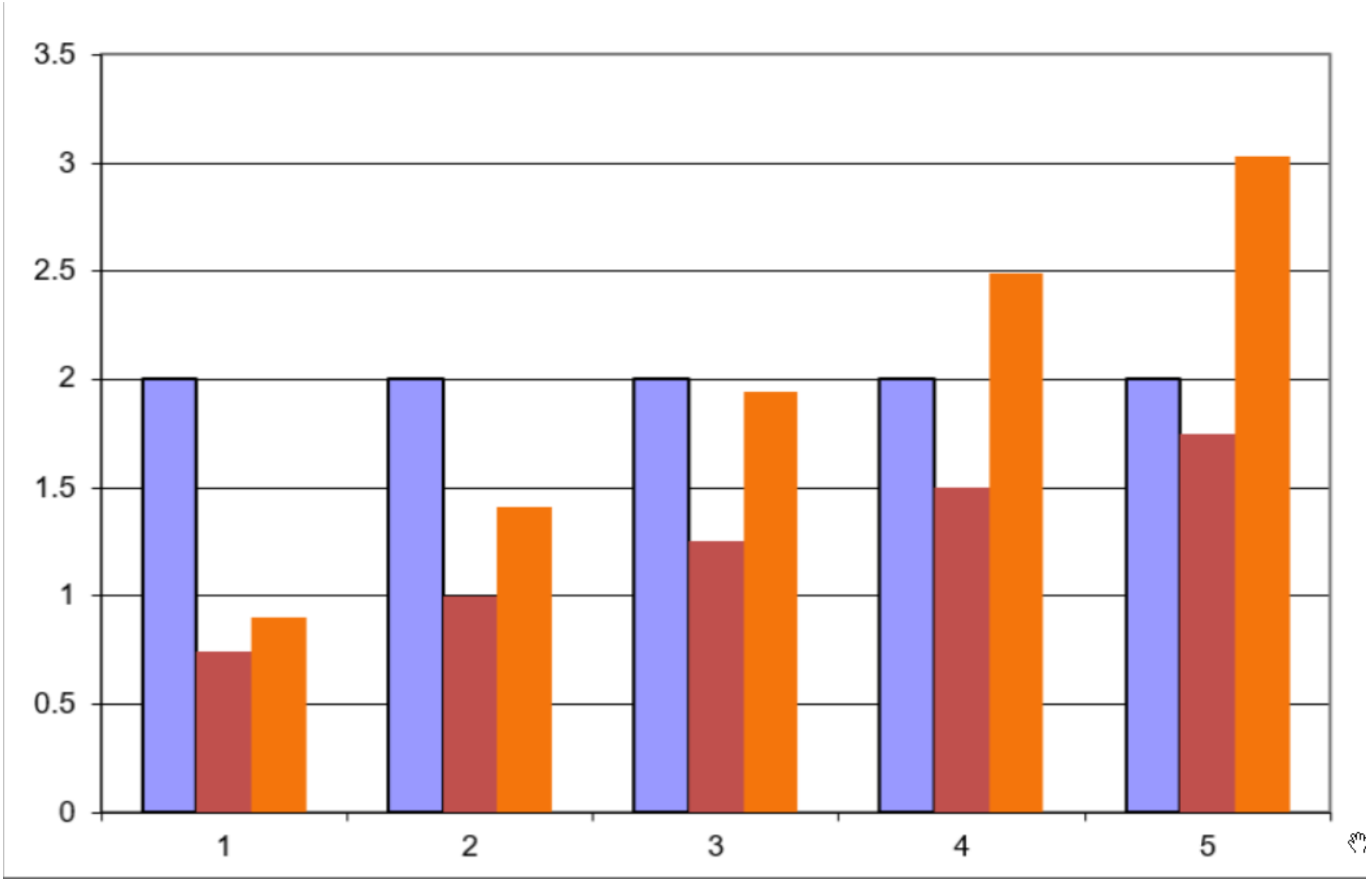
$$X^{KG}(S^n) = argmax_x \nu_x^{KG,n}$$

where (remember that x is the experiment we are thinking of doing)

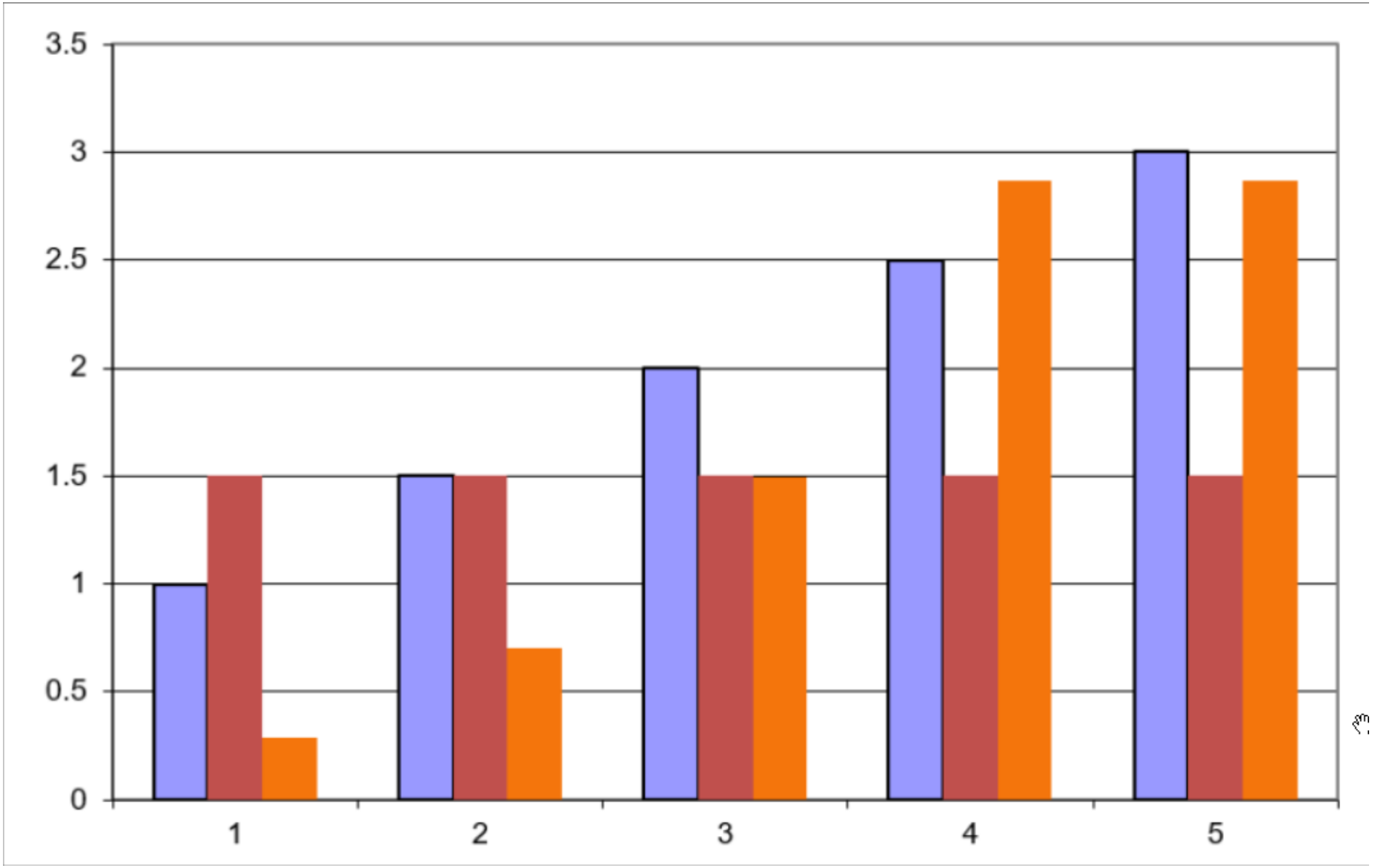
$$\nu_x^{KG,n} = \mathbb{E}\max_{x'} \bar{\mu}_{x'}^{n+1}(x) - \max_{x'} \bar{\mu}_{x'}^n$$

  - » Both policies balance the value of doing well now, and the value of information in the future

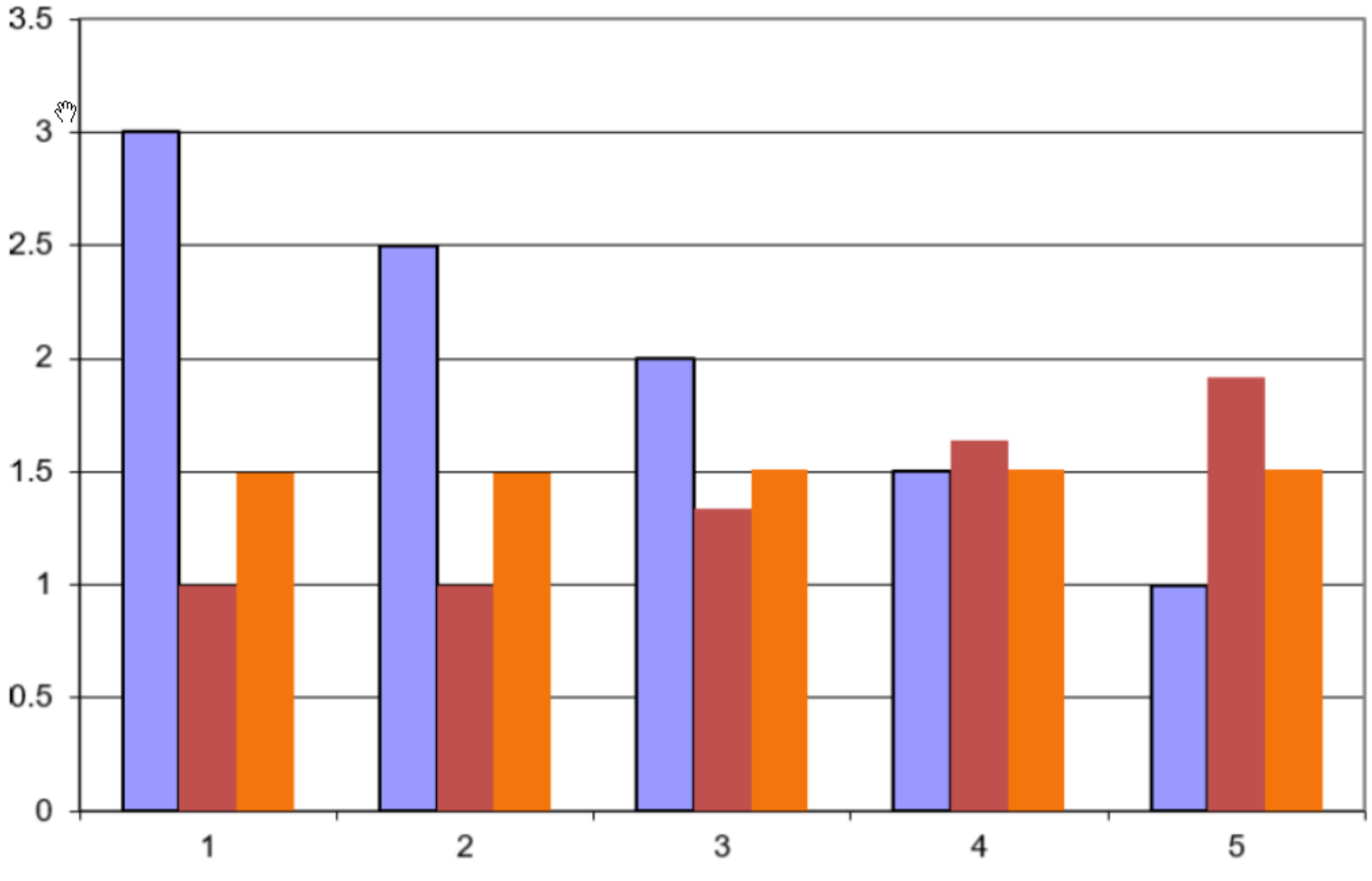
Properties of the knowledge gradient



Properties of the knowledge gradient



Properties of the knowledge gradient



Properties of the knowledge gradient

- Notes:
  - » The KG likes decisions that appear to be good.
  - » It also likes experiments that are uncertain.
  - » It will not pick choices where you are *sure* it is bad.
  - » It will never pick a choice where there is no uncertainty, because there is nothing to learn.
  - » It likes choices that have a *chance* of being the best.
  - » Refer to decision tree for hitter (next slide): we choose batter B because there is a *chance* that batter B might be best.

Lookahead policies: S-curve effect and a limited multistep lookahead

4.4.1 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

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

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.

The S-curve effect

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

$$\nu_x^{KG}(n_x) = \mathbb{E}\left[\max_{x'}(\bar{\mu}_{x'}^0 + \tilde{\sigma}_{x'}^0(n_{x'})Z_{x'})\right] - \max_{x'}\bar{\mu}_{x'}^0.$$

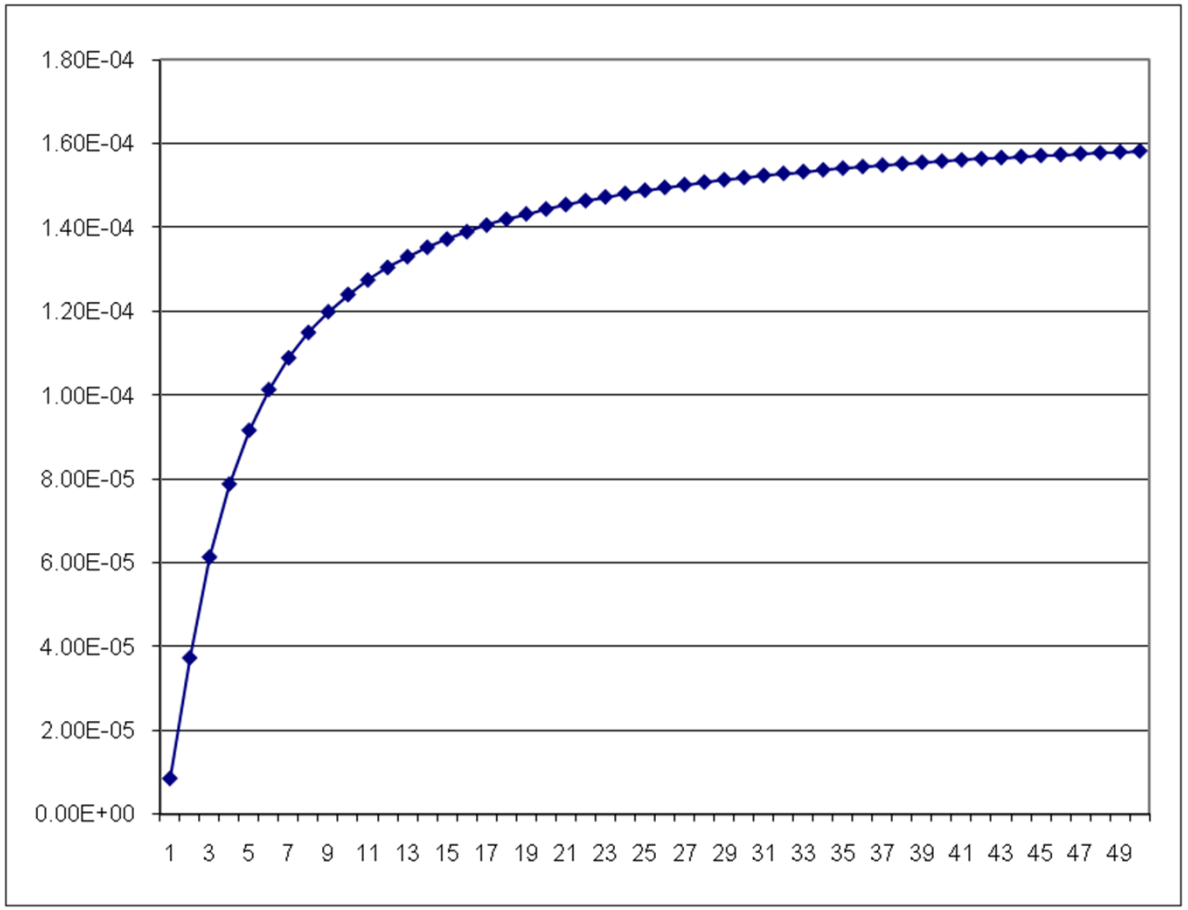
We can compute the value of n_x observations of alternative x using the knowledge gradient formula in equation (4.12),

$$\nu_x^{KG}(n_x) = \tilde{\sigma}_x^0(n_x) f\left(\frac{\bar{\mu}_x^0 - \max_{x' \neq x}\bar{\mu}_{x'}^0}{\tilde{\sigma}_x^0(n_x)}\right),$$

where $f(\zeta)$ is given in equation (4.11).

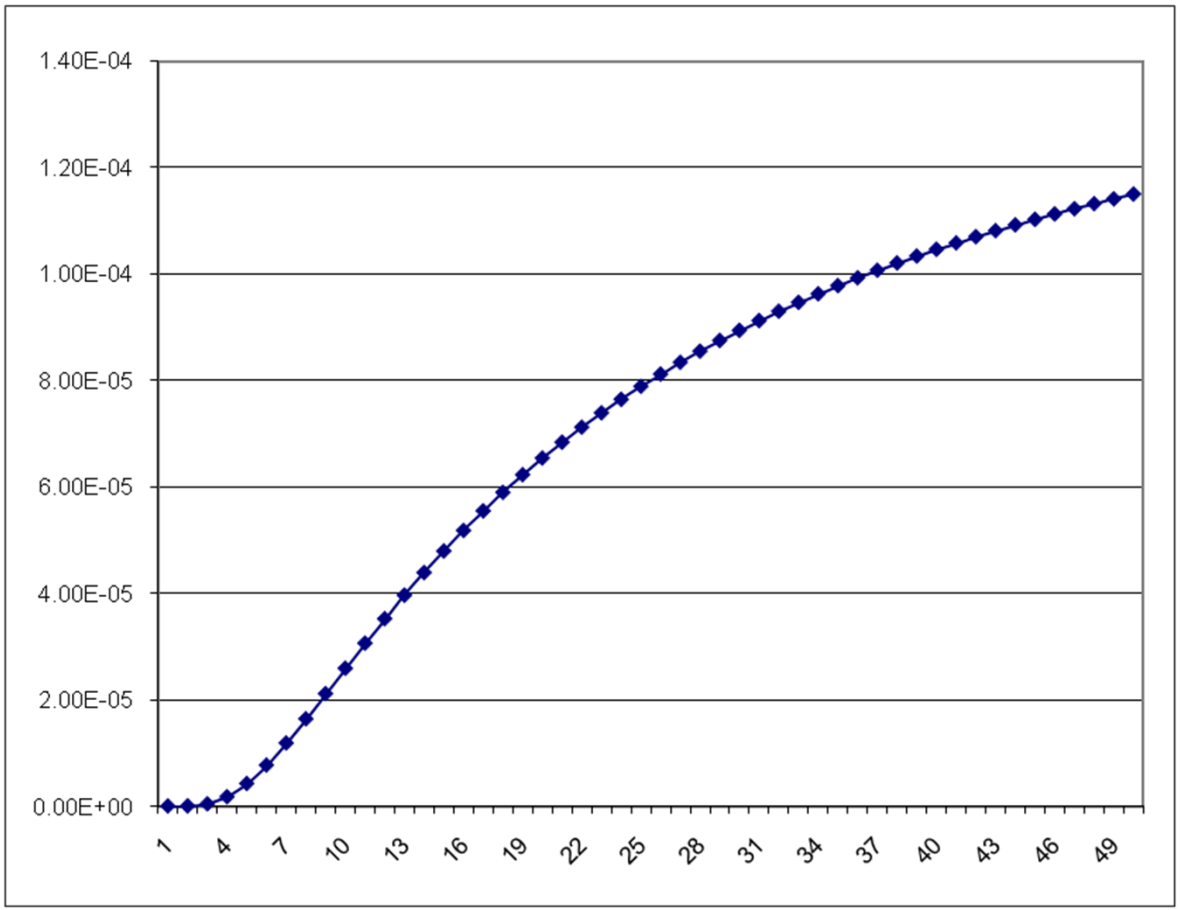
The S-curve effect

- The marginal value of information
  - » Imagine that we are choosing between an uncertain alternative we can measure, and a certain one. Click here for spreadsheet.



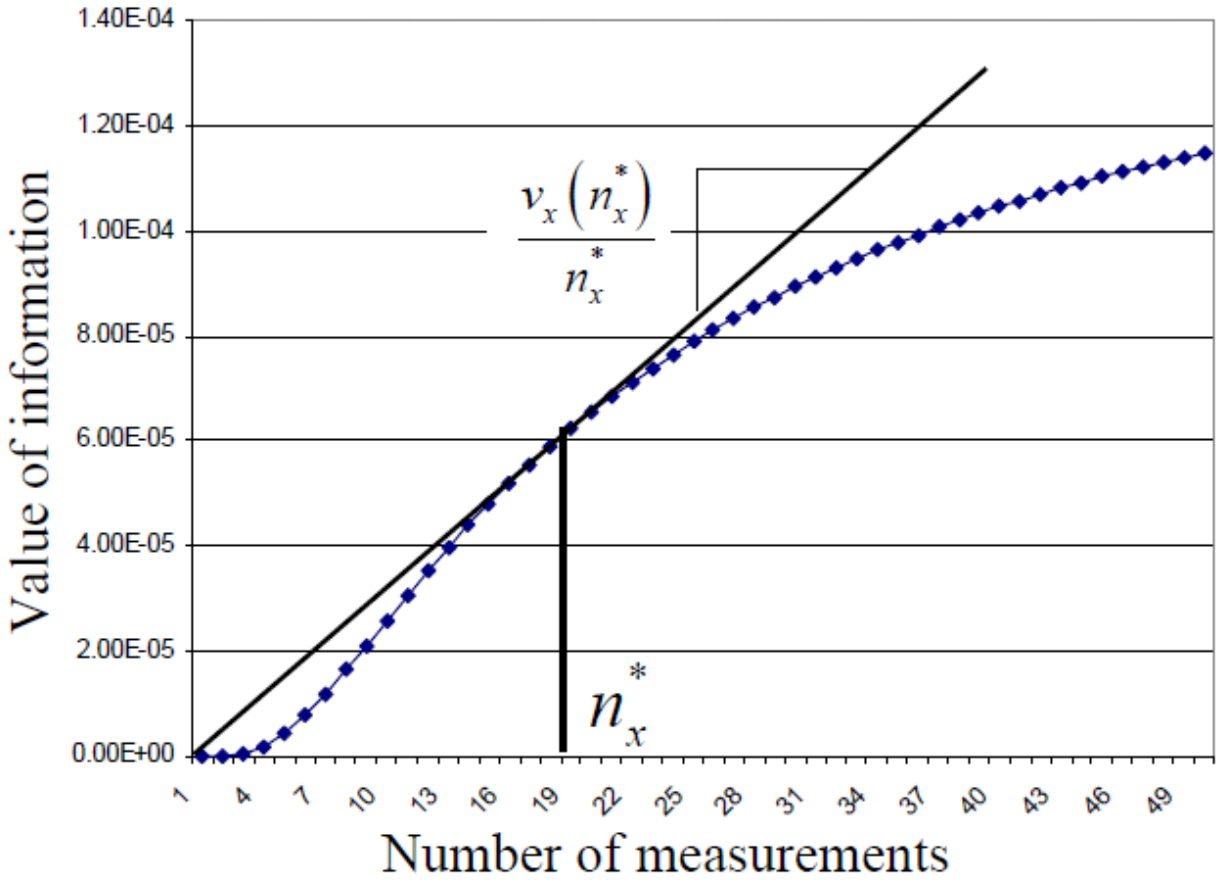
The S-curve effect

- The marginal value of information
  - » Imagine that we are choosing between an uncertain alternative we can measure, and a certain one. Click here for spreadsheet.



The S-curve effect

- The KG(*) policy
  - » Assume that we are going to make enough measurements to maximize the average value of an alternative:



$$n_x^* = \arg\max_{n_x>0} \frac{v_x(n_x)}{n_x}.$$

The S-curve effect

- The KG(*) policy
  - » The KG(*) policy is just like the KG policy with one twist…
  - » Instead of updating the precision with

$$\beta^{n+1} = \beta^n + \beta^W$$

… we use instead

$$\beta^{n+1} = \beta^n + n^*\beta^W$$

  - » What we are doing is pretending that the experiment is more precise than it is.
  - » As an alternative, we can introduce a tunable parameter τ:

$$\beta^{n+1}(\tau) = \beta^n + \tau\beta^W$$

  - » Now let $\nu_x^{KG,n}(\tau)$ be the knowledge gradient computed using precision $\tau\beta^W$ instead of β^W. Now we have a one-step lookahead policy, but with a parameter τ that now has to be tuned just as we did with the earlier heuristic policies.
  - » When the value of information is not concave, this can be quite valuable.

The S-curve effect

4.4.3 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 $\beta^1 = \beta^0 + \theta^{KG}\beta^W$.

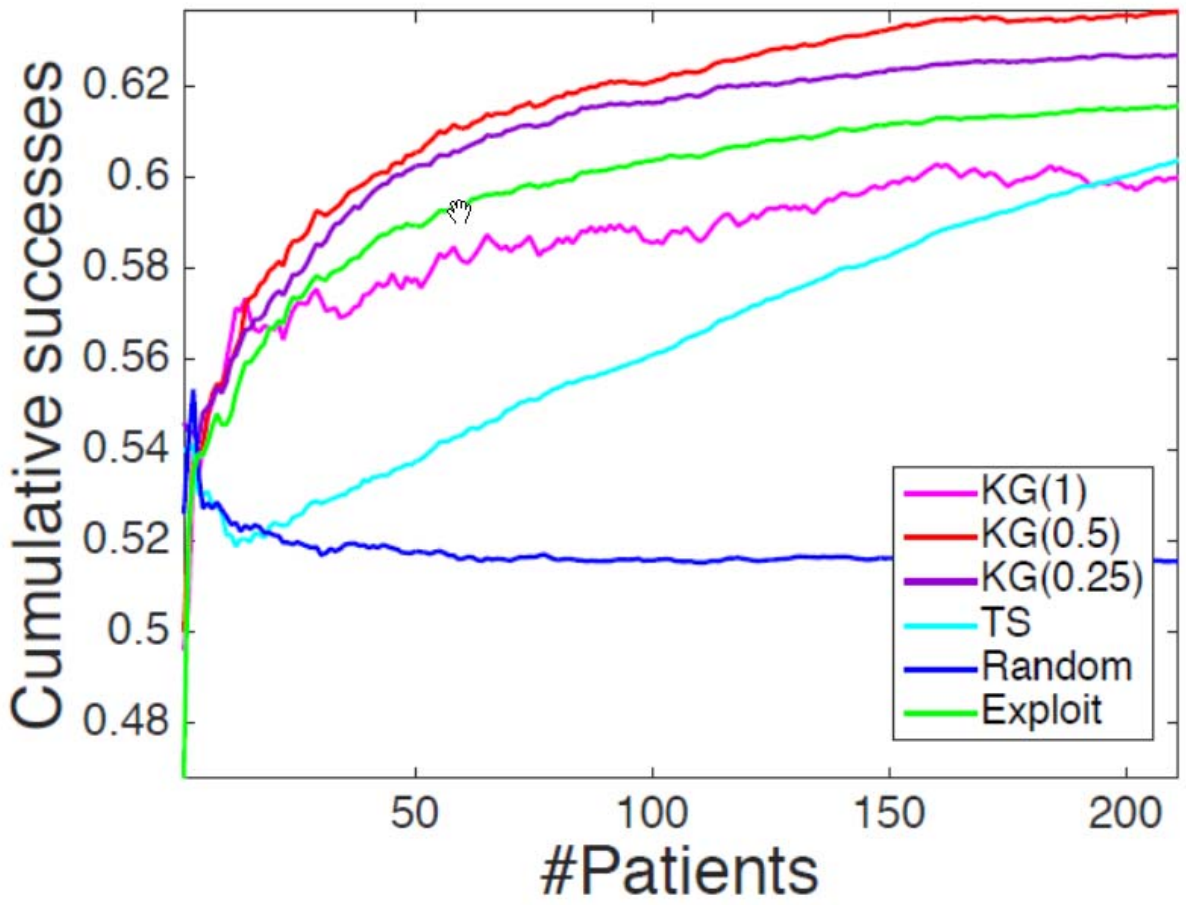
We then define our tunable KG policy as

$$X^{KG}(\theta^{KG}) = \arg\max_x \nu_x^{KG,n}(\theta^{KG})$$

We now have to tune our policy to find the best value of θ^{KG} (see section 3.4 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.

The S-curve effect

- Comparison of policies on health application
  - » Patients arrive at random with a set of characteristics *a*
  - » Choice *x* is a medical decision
  - » otherwise.



The S-curve effect

- The paradox of too many choices
  - » Situation: You are the coach of a travel baseball team. Would you rather have 100 kids show up to try out, or 200?
  - » You have a fixed amount of time to evaluate kids. Imagine that as you have more choices, you have to allocate your measurement budget more thinly.
  - » So $n_x = \frac{B}{M}$ where:
    - B = total number of measurements (budget)
    - M = number of alternatives

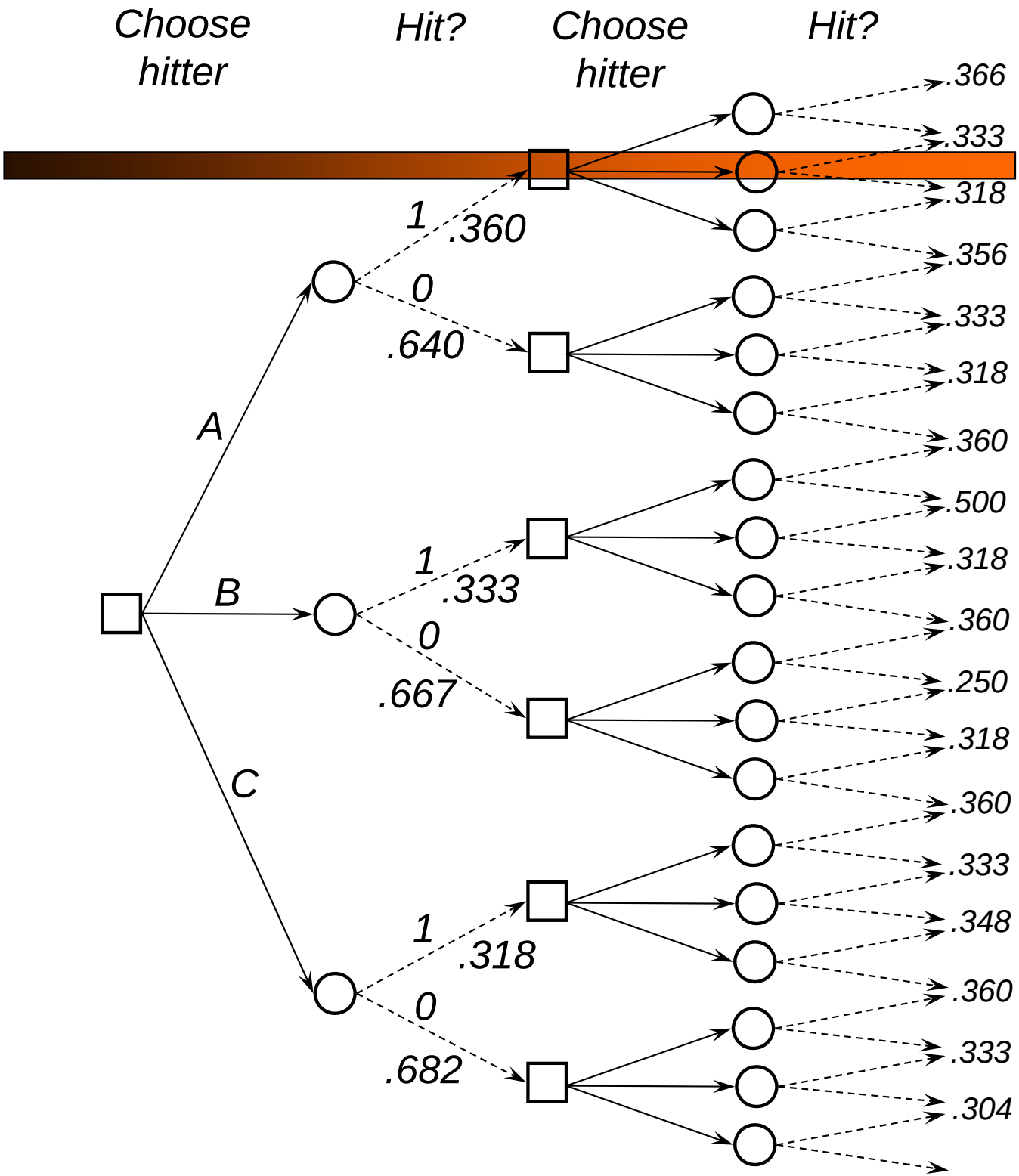
Lookahead policies:
Full tree search

Decision trees

- Looking further into the future

  - » Ultimately, we can always make decisions by planning for multiple periods into the future.

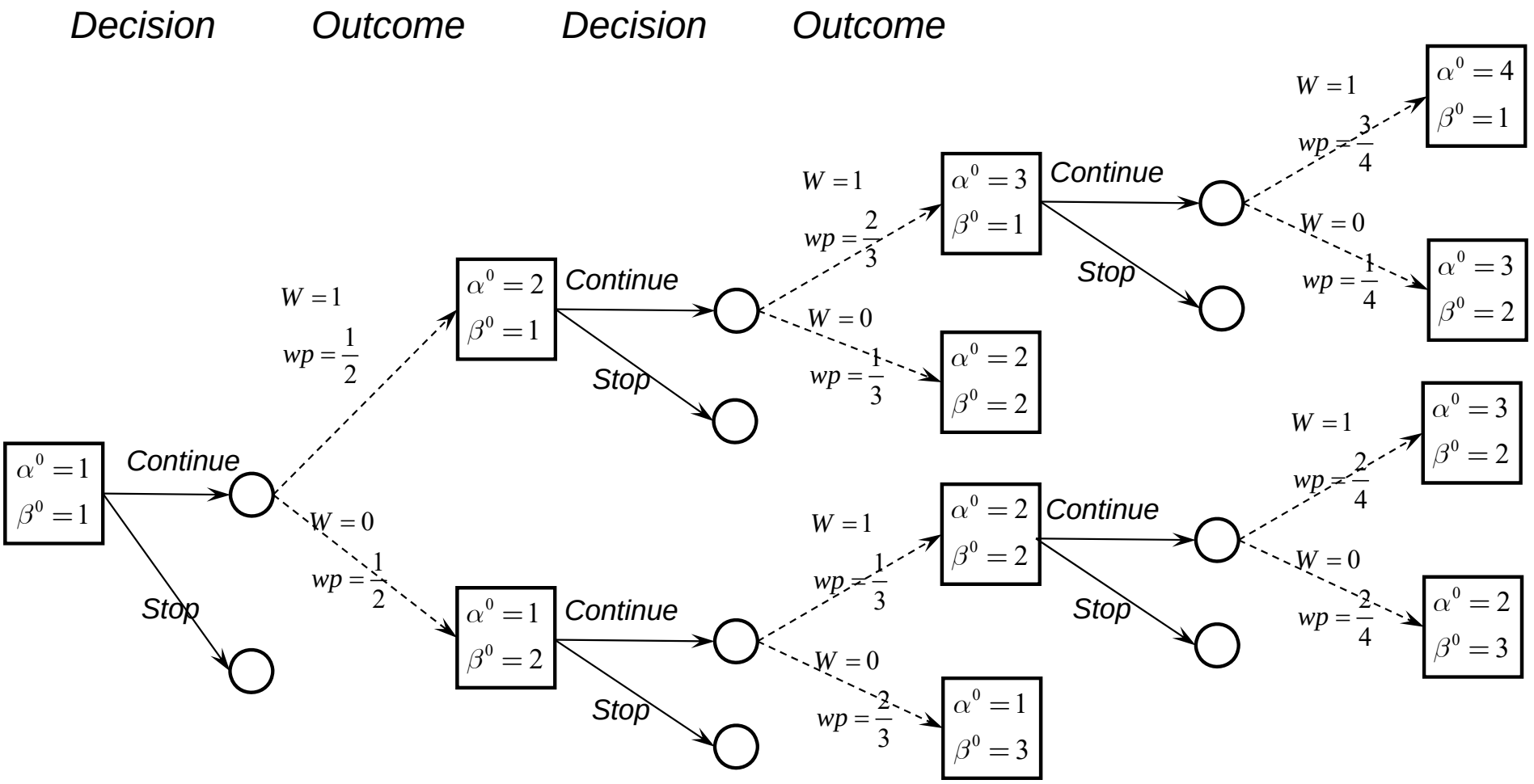
  - » We first introduced this idea with our baseball example in the second lecture…


Choose hitter
Hit?
Choose hitter
Hit?
A
B
C
1
.360
0
.640
1
.333
0
.667
1
.318
0
.682
.366
.333
.318
.356
.333
.318
.360
.500
.318
.360
.250
.318
.360
.333
.348
.360
.333
.304

Decision trees

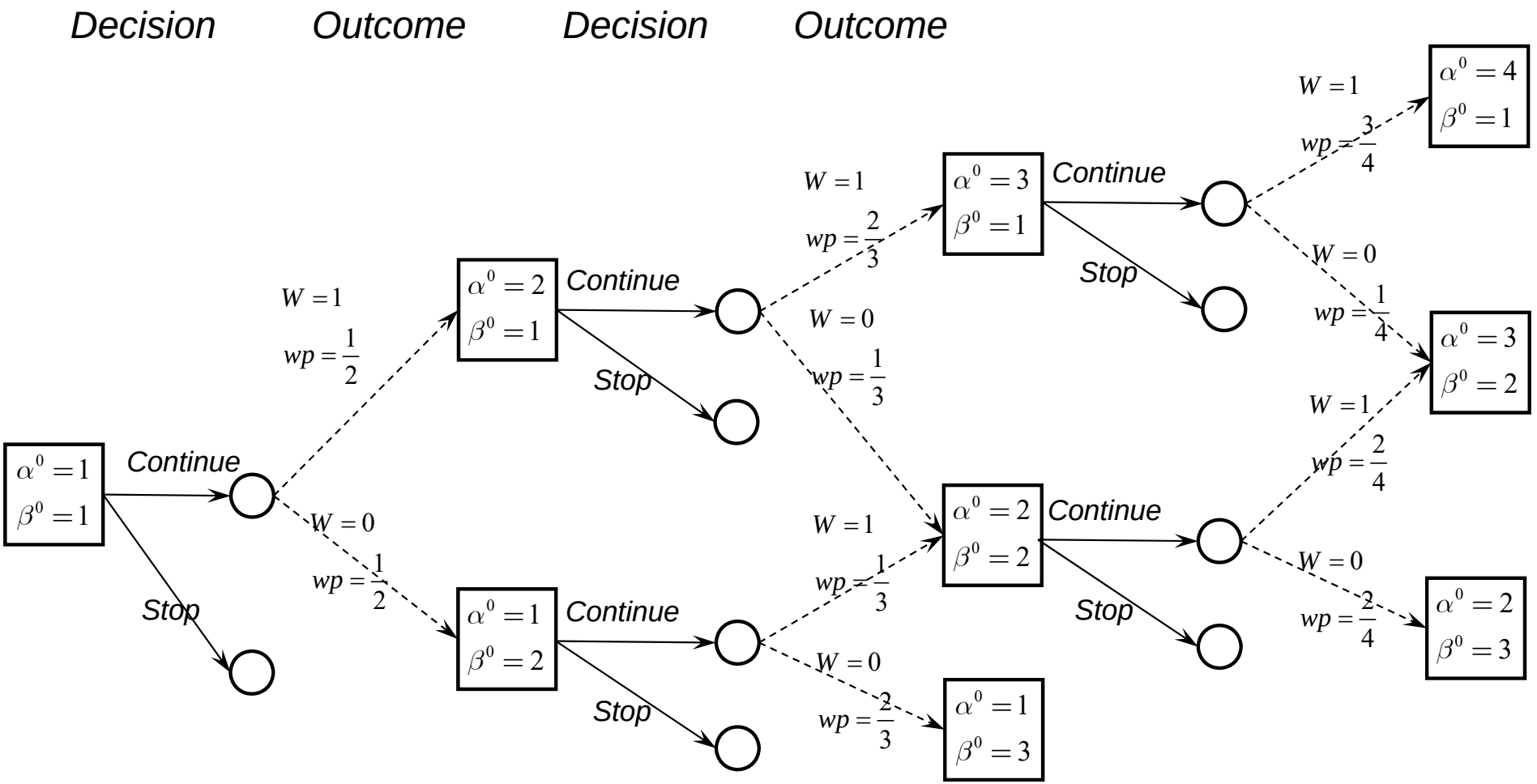
- From decision trees to dynamic programs.
  - » Some people equate a decision tree with a dynamic program.
  - » This is not true. The next two slides use a learning problem with a beta-prior (similar to the hitting example) where outcomes are success/failure.
  - » These slides illustrate the fundamental difference between decision trees and dynamic programs:
    - In a decision tree, the state variable is the history, which can contain more information than is necessary.
    - In a dynamic program the state variable is only the information that is needed.

Decision trees



As a decision tree (the "state variable" is the history)

Decision trees



As a dynamic program (state variables use only necessary information)

Decision trees

- From March 2, 2019:

Peter Jacko Sat, Mar 2, 8:20 PM (2 days ago)
to Reinforcement Unsubscribe

Dear all,

I am writing a paper about solving the two-armed Bayesian Beta-Bernoulli bandit problem by dynamic programming (DP), where I would like to give a summary what is solvable on a standard desktop/laptop, when the code is written by different people in different programming languages.

Those of you who have a code for computing the optimal solution by DP, could you please reply (to me only) with the information below? If such (or similar) information is published in a paper you are aware of, please provide a reference/link, I will cite it. If it is not published, you can send me that information privately, I will acknowledge you in the list of people who provided the results, but will anonymise your data, as I don't indend to create an undesirable discussion about coding skills of particular people...

I would appreciate your replies within the next 2 weeks, i.e. by 15 March 2019.

Many thanks,

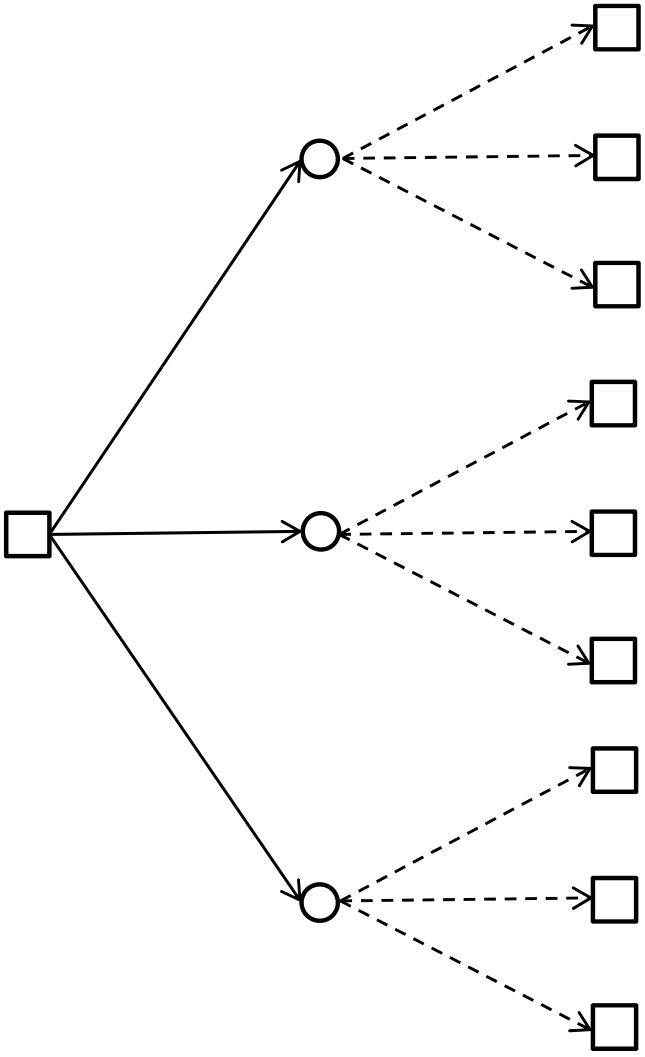
Peter.

Author of the code:
Programming language (list packages if using any):
Amount of available RAM memory (from the "Task manager") just before running the code to answer the next two questions:
Specifics of the computer (if not standard desktop/laptop with CPU around 2GHz):
How long does it take to compute the optimal policy array (i.e. action for every possible state) for horizon 60, 120, 180, 240, 300?
How long does it take to compute the optimal value (without memorizing the policy array) for horizon 60, 120, 180, 240, 300?
What is the largest horizon as a multiple of 60 for which you don't get an OutOfMemory error?

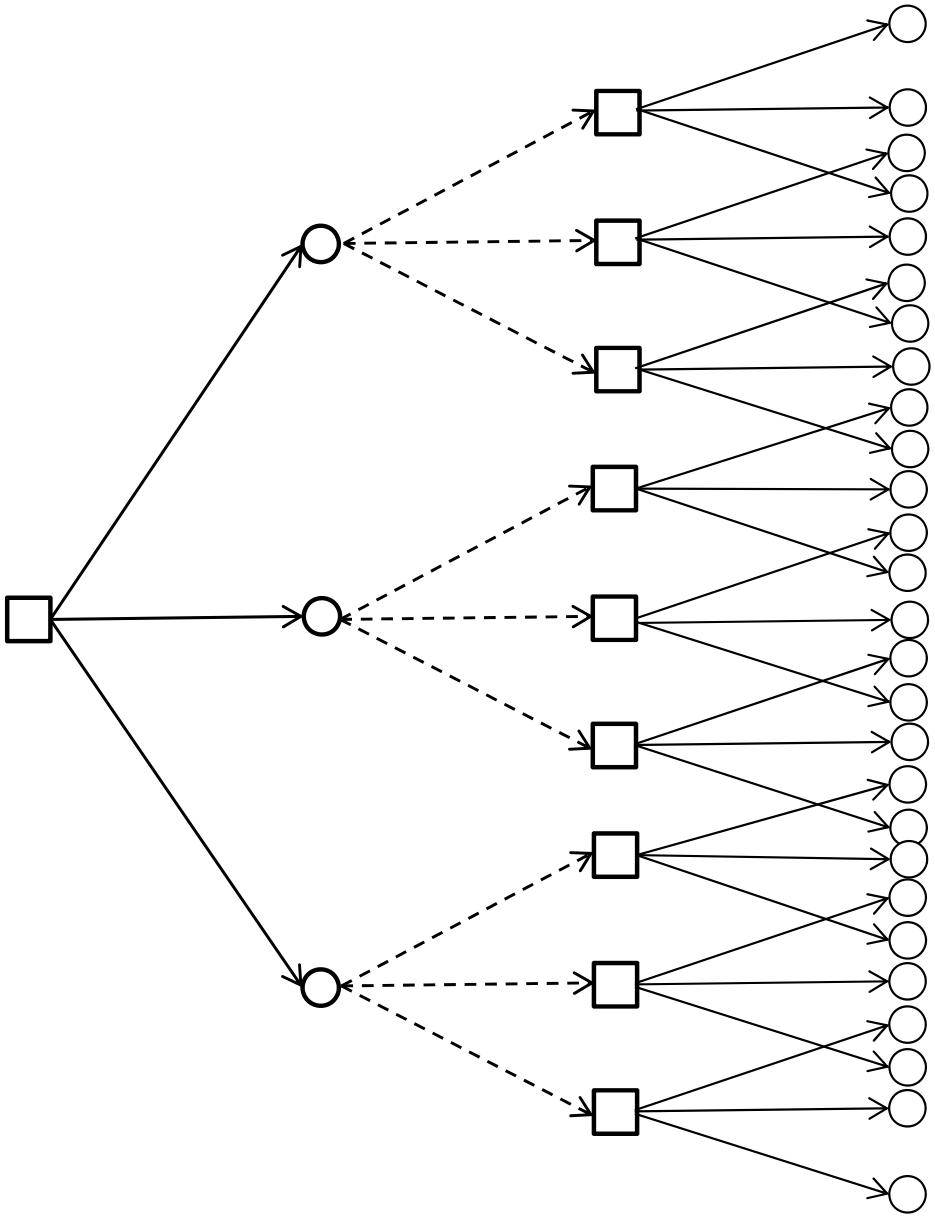
Decision trees

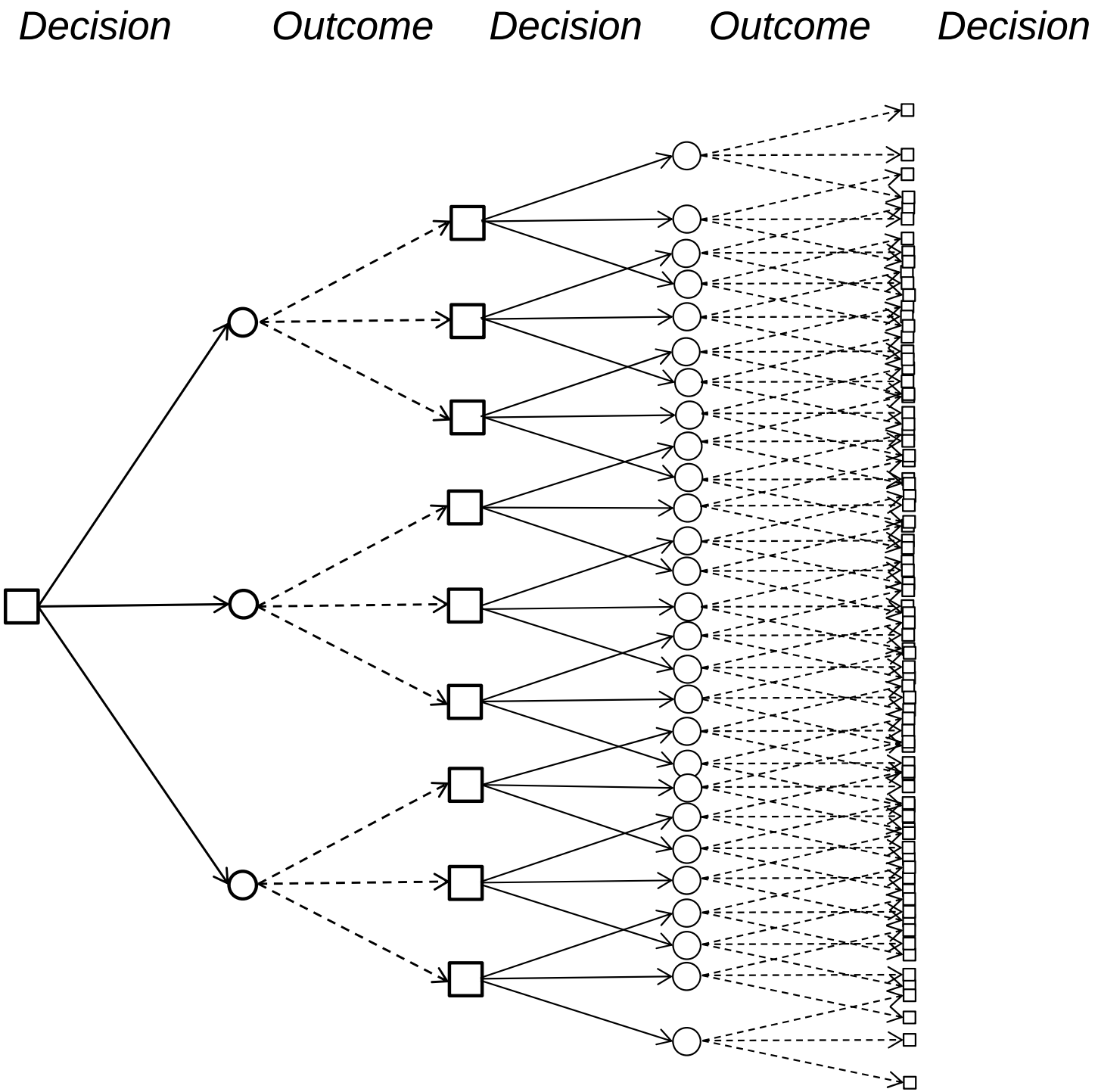
- The next series of slides illustrate the exponential expansion when we use decision trees (the path from the origin to each node is unique).

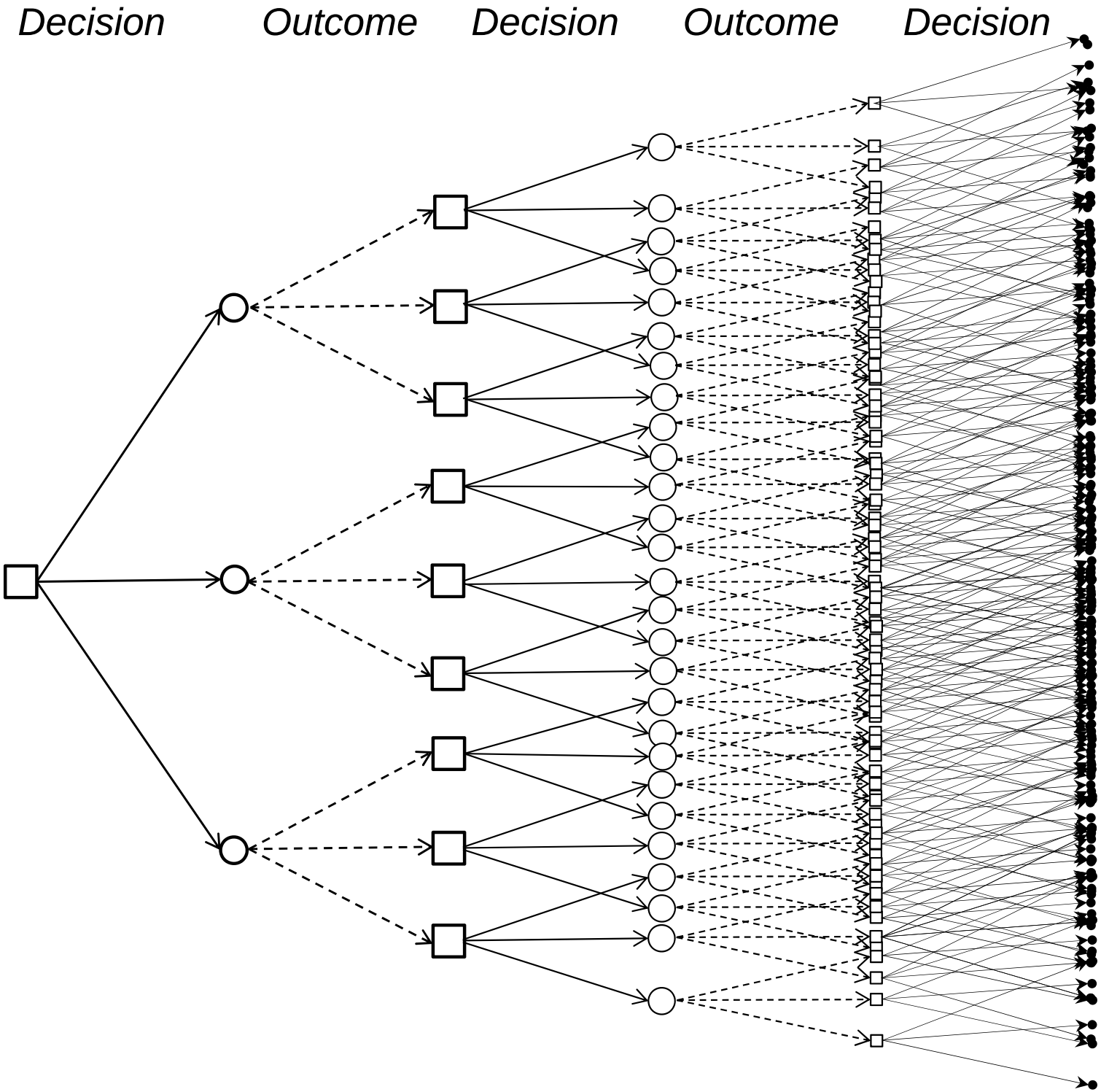
Decision *Outcome* *Decision* *Outcome* *Decision*



Decision *Outcome* *Decision* *Outcome* *Decision*




Decision
Outcome
Decision
Outcome
Decision


Decision
Outcome
Decision
Outcome
Decision

Offline vs. online objectives

Offline vs. online learning

- "Offline" optimization (maximizing final reward)
  - » We can iteratively search for the best solution, but only care about the final answer.
  - » Asymptotic formulation:

$$\max_x \mathbb{E}\{F(x,W) \mid S^0\}$$

  - » Finite horizon formulation:

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

    - This will be our standard formulation, since this is what we do in practice.
    - The policy $X^\pi(S^n)$ might be a rule for making a decision (e.g. choose x that appears to be best), but we might call it an algorithm.

Offline vs. online learning

- Evaluating an offline learning policy/algorithm:
  - » Over time, I will encourage you to write expectations subscripted by the random variable over which you are taking the expectation. If W is the only random variable, we would then write:]

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

  - » If we have uncertainty in the initial state, then we have to perform iterative learning, and we then have to test how well the policy works, we would have three levels of nesting:

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

  - » *Always* be ready to replace any expectation with a sampled approximation. This means you have to know what random variables you are sampling over.

Offline vs. online learning

- "Online" optimization (maximizing final reward)
  - » If we have an online learning policy, then we have only two sources of randomness:
    - The distribution of belief about the mean demand.
    - The randomness you experience while learning the best order quantity.
  - » We would write our nesting in the evaluation of a learning policy as

$$\max_{\pi} \mathbb{E}_{S^0} \mathbb{E}_{W^1,W^2,\ldots,W^N|S^0} \left\{ \sum_{n=0}^{N-1} F(X^{\pi}(S^n), W^{n+1}) \mid S_0 \right\}$$

Offline vs. online learning

- Notes:
  - » The machine learning community uses the term "online" to refer to any sequential algorithm, while "offline" refers to a batch algorithm.
  - » In optimization, "offline" typically means in a laboratory or simulator, while "online" means in the field, where you have to experience results as they happen.
  - » To be clear, I will use "final reward" when we only care about the quality of the final solution, rather than how we get there. "Cumulative reward" means we sum the rewards as we progress.

The knowledge gradient for online learning

Knowledge gradient for online learning

5.3.1 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

$$V^{Stop,n}(S^n) = (N - n + 1) \max_x \bar{\mu}_x^n, \tag{5.16}$$

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 $V^{Stop,n}(S^n) = 6 \cdot 5.5 = 33$.

Knowledge gradient for online learning

Consider a different thought experiment. We are still at time n, but now our next decision will change our beliefs as usual. 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 $\arg\max_x \bar{\mu}_x^{n+1}$. 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

$$X^{KG,n} = \arg\max_x \mathbb{E}\left[\mu_x + V^{Stop,n+1}\left(S^{n+1}\right) \mid S^n, x^n = x\right] \tag{5.17}$$

$$= \arg\max_x \bar{\mu}_x^n + (N-n)\,\mathbb{E}\left[\max_{x'} \bar{\mu}_{x'}^{n+1} \mid S^n, x^n = x\right] \tag{5.18}$$

$$= \arg\max_x \bar{\mu}_x^n + (N-n)\,\mathbb{E}\left[\max_{x'} \bar{\mu}_{x'}^{n+1} - \max_{x'} \bar{\mu}_{x'}^n \mid S^n, x^n = x\right] \tag{5.19}$$

$$= \arg\max_x \bar{\mu}_x^n + (N-n)\,\nu_x^{KG,n}, \tag{5.20}$$

where $\nu_x^{KG,n}$ is simply the knowledge gradient for ranking and selection, given by (4.12).

Knowledge gradient for online learning

- Discounted finite horizon

$$X^{KG,n} = \arg\max_x \bar{\mu}_x^n + \gamma \frac{1-\gamma^{N-n}}{1-\gamma} \nu_x^{KG,n}.$$

- Discounted infinite horizon

$$X^{KG,n} = \arg\max_x \bar{\mu}_x^n + \frac{\gamma}{1-\gamma} \nu_x^{KG,n}.$$

- Tunable versions:

$$\beta_x^{n+1} = \beta_x^n + \theta_1 \beta^W.$$

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.

Knowledge gradient for online learning

- Knowledge gradient policy
  - » For finite-horizon on-line problems:

$$\nu_x^{KG-OL,n} = \bar{\mu}_x^n + (N-n)\nu_x^{KG,n} \longrightarrow \textit{Value of information}$$

  - » For infinite-horizon discounted problems:

$$\nu_x^{KG-OL,n} = \bar{\mu}_x^n + \frac{\gamma}{1-\gamma}\nu_x^{KG,n} \longrightarrow \textit{Value of information}$$

- Compare to Gittins indices for bandit problems

$$\nu_x^{Gittins} = \bar{\mu}_x^n + \Gamma(n,\gamma)\sigma_W \longrightarrow ???$$

- …interval estimation

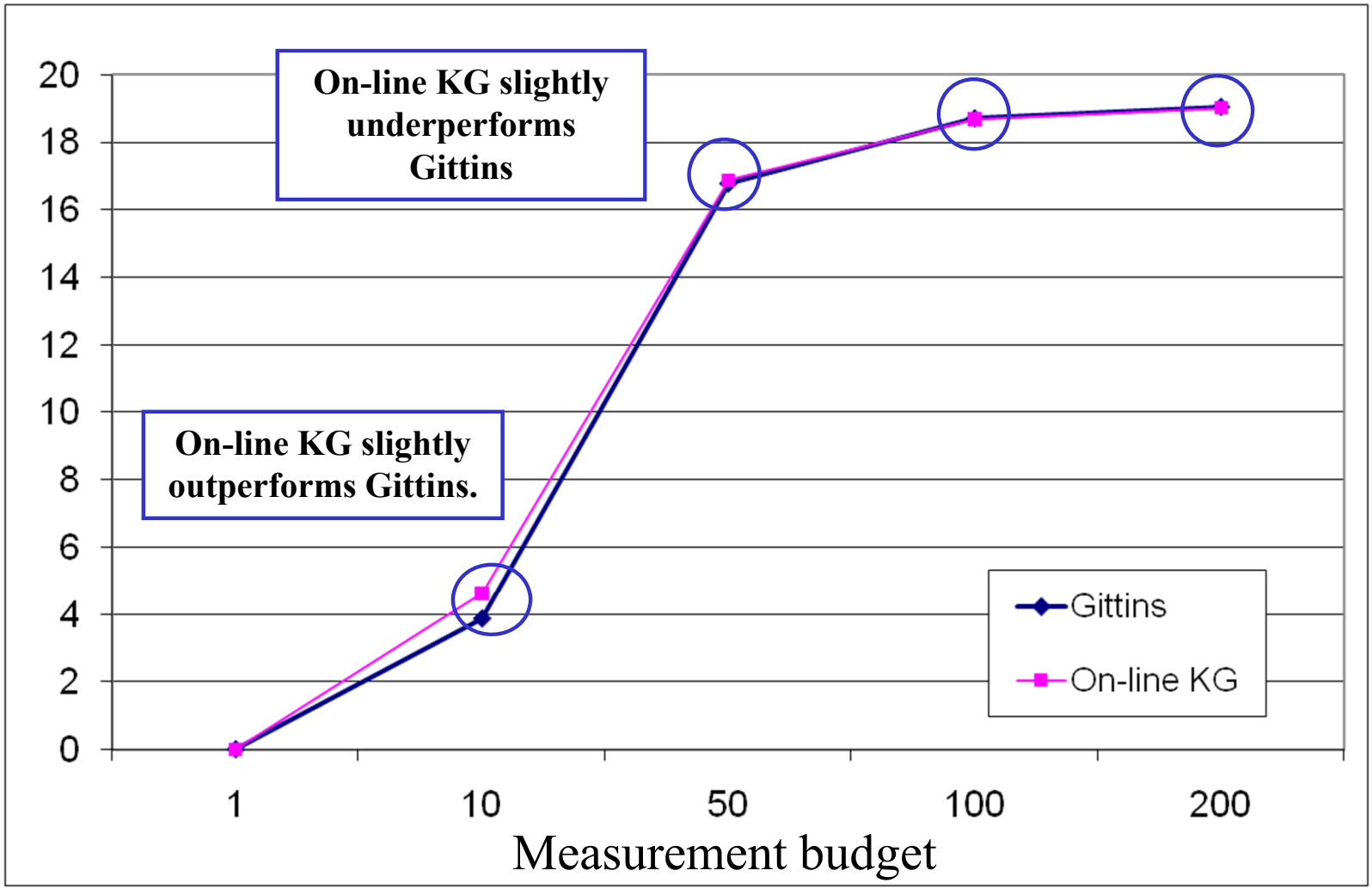
$$\nu_x^{IE,n} = \bar{\mu}_x^n + z_\alpha \bar{\sigma}_x^n \longrightarrow ???$$

- … and UCB

$$\nu_x^{UCB1} = \bar{\mu}_x^n + 4\sigma^\varepsilon \sqrt{\frac{\log n}{N_x^n}} \longrightarrow ???$$

Knowledge gradient for online learning

- On-line KG vs. Gittins for finite horizon problems



Knowledge gradient for online learning

- Contrast:
  - » Offline vs. online KG
    - Offline: $X^{\pi}(S^n) = argmax_x \nu_x^{KG,n}$
    - Online: $X^{\pi}(S^n) = argmax_x(\bar{\mu}_x^n + (N-n)\nu_x^{KG,n})$
  - » Tuning PFA policies for offline vs. online
    - Policies are the same – just tuned differently
    - E.g. Interval estimation:

      $$X^{\pi}(S^n|\theta) = argmax_x(\bar{\mu}_x^n + \theta\bar{\sigma}_x^n)$$

    - Optimize for offline:

      $$\max_{\theta} \mathbb{E}\_\mu \mathbb{E}_{W^1,\ldots,W^N|\mu} \mathbb{E}_{\widehat{W}} F(x^{\pi,N}, \widehat{W})$$

    - Optimize for online:

      $$\max_{\theta} \mathbb{E}_{\mu}\, \mathbb{E}_{W^1,\ldots,W^N|\mu} \sum_{n=0}^{N} F(X^{\pi}(S^n|\theta), W^{n+1})$$

Classifying policies

Policies

- Board setup:
  - » Left side – list examples of different policies
  - » Right side – The two ways of creating policies
    - Policy search – Top board
    - Lookahead policies – Bottom board
  - » Middle board – List the four classes of policies
    - Top board – PFAs, and CFAs
    - Bottom board – VFAs and DLAs
  - » Ask class to match policies with class

Policies

- List of policies (left board)
  - 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.

Policies

- List of policies (left board)
  - 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 5.1) - 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.

Learning policies

- Heuristic learning policies
  - » **Pure exploitation** – Always make the choice that appears to be the best.
  - » **Pure exploration** – Make choices at random so that you are always learning more.
  - » **Epsilon-greedy**
    - Explore with probability ε and exploit with probability $1-\varepsilon$
    - Epsilon-greedy exploration – explore with probability $\varepsilon^n = c/n$. Goes to zero as $n \to \infty$, but not too quickly.

- Discuss
  - » Convergence
  - » Applying to large problems

Learning policies

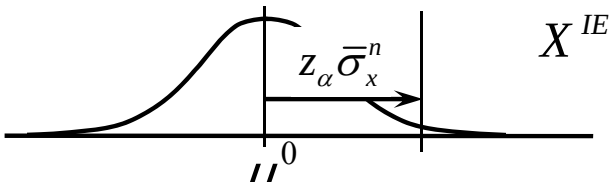
- Heuristic measurement policies
  - » **Boltzmann exploration**
    - Explore choice x with probability $P_x^n(\theta) = \dfrac{e^{\theta \bar{\mu}_x^n}}{\sum_{x'} e^{\theta \bar{\mu}_{x'}^n}}$
    - $X^{Boltz}(S^n|\theta) = \arg\max_x \{x | P_x^n(\theta) \leq U\}.\ U \sim [0,1]$
  - » **Upper confidence bounding**

$$X^{UCB}(S^n \mid \theta^{UCB}) = \arg\max_x \left(\bar{\mu}_x^n + \theta^{UCB} \sqrt{\frac{\log n}{N_x^n}} \right)$$

  - » **Thompson sampling**

$$X^{TS}(S^n) = \arg\max_x \hat{\mu}_x^{n+1} \quad \text{where } \hat{\mu}_x^n \sim N(\bar{\mu}_x^n, \beta_x^n)$$

  - » **Interval estimation** (or upper confidence bounding)
    - Choose x which maximizes



$$X^{IE}(S^n \mid \theta^{IE}) = \arg\max_x \bar{\mu}_x^n + \theta^{IE} \bar{\sigma}_x^n$$

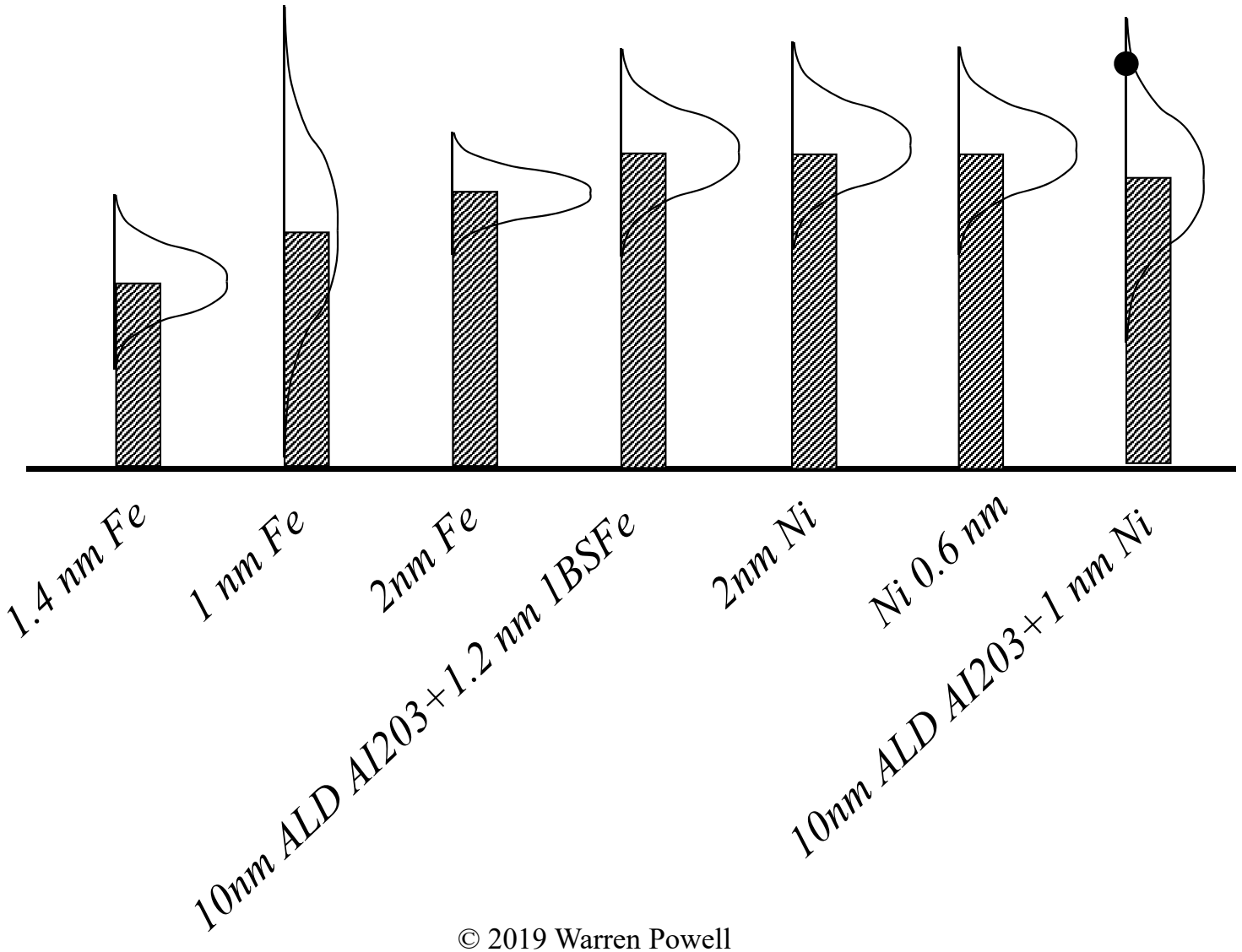
Creating belief models

Creating belief models

- Notes:
  - » At the heart of any learning problem is the belief model
  - » If experiments are expensive, you want to exploit domain knowledge about the structure of the problem.

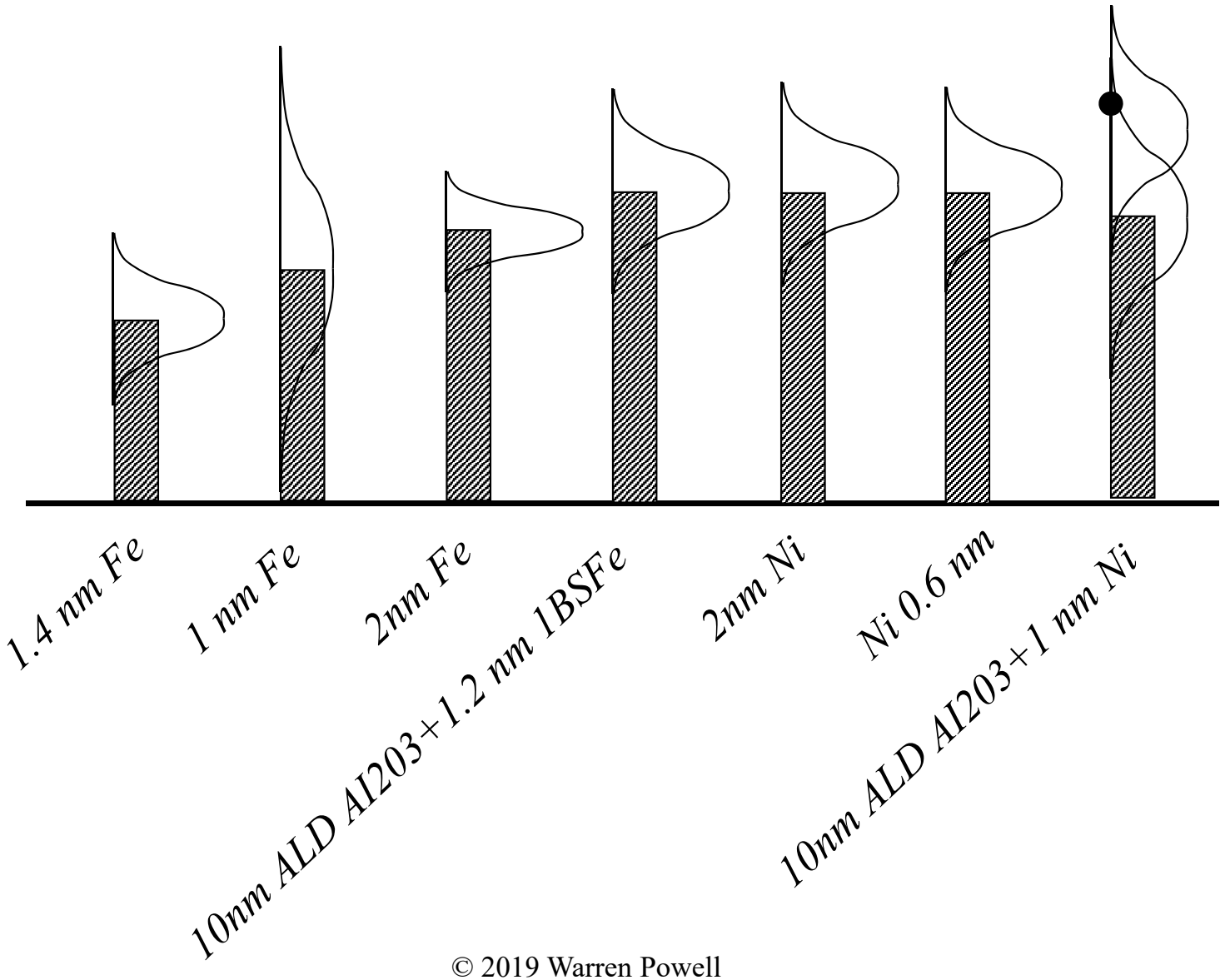
The learning challenge

- Finding the best material



The learning challenge

- Finding the best material



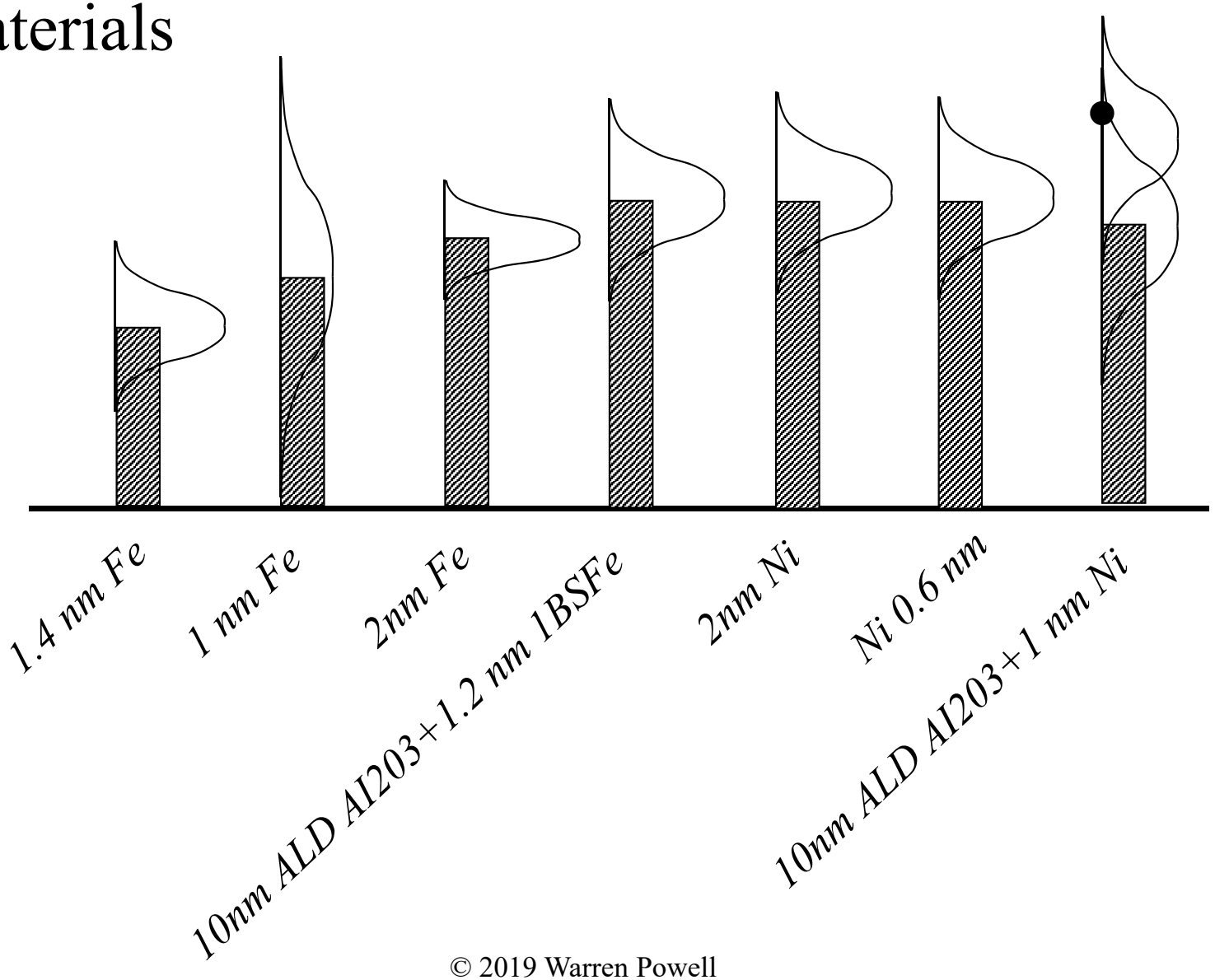
The learning challenge

- Correlations
  - » Simple belief model assumes independence
  - » Similar material properties between catalysts
  - » Scientists using domain knowledge can estimate correlations in experiments between similar catalysts.

	1.4 nm Fe	1 nm Fe	2nm Fe	10nm ALD Al2O3+1.2 nm IBS Fe	2 nm Ni	Ni 0.6 nm	10nm ALD Al2O3+1 nm Ni
1.4 nm Fe	1	0.7	0.7	0.6	0.4	0.4	0.2
1 nm Fe	0.7	1	0.7	0.6	0.4	0.4	0.2
2nm Fe	0.7	0.7	1	0.6	0.4	0.4	0.2
10nm ALD Al2O3+1.2 nm IBS Fe	0.6	0.6	0.6	1	1	0.3	0
2 nm Ni	0.4	0.4	0.4	1	1	0.7	0.6
Ni 0.6 nm	0.4	0.4	0.4	0.3	0.7	1	0.6
10nm ALD Al2O3+1 nm Ni	0.2	0.2	0.2	0	0.6	0.6	1

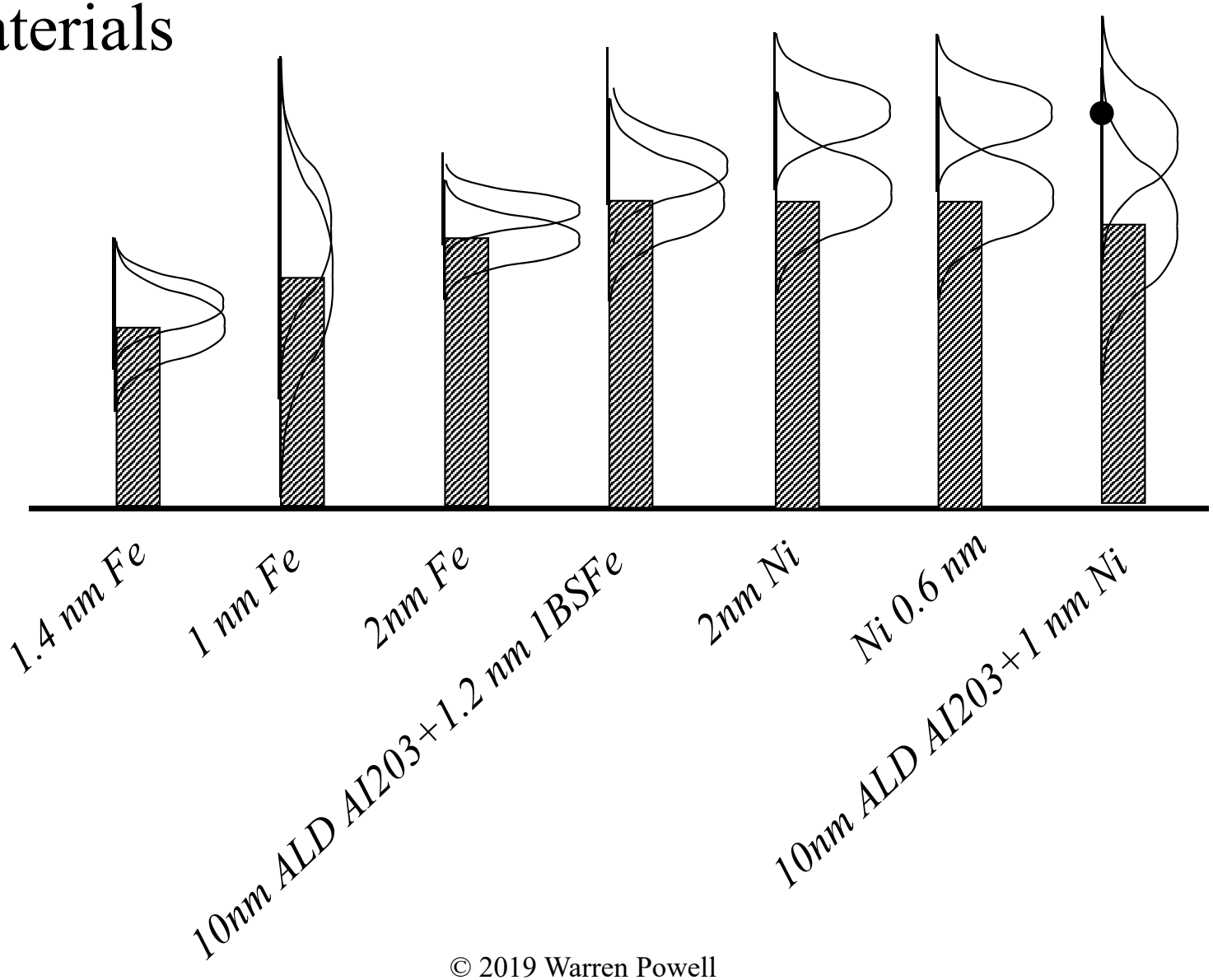
The learning challenge

- Testing one material teaches us about other materials



The learning challenge

- Testing one material teaches us about other materials



The learning challenge

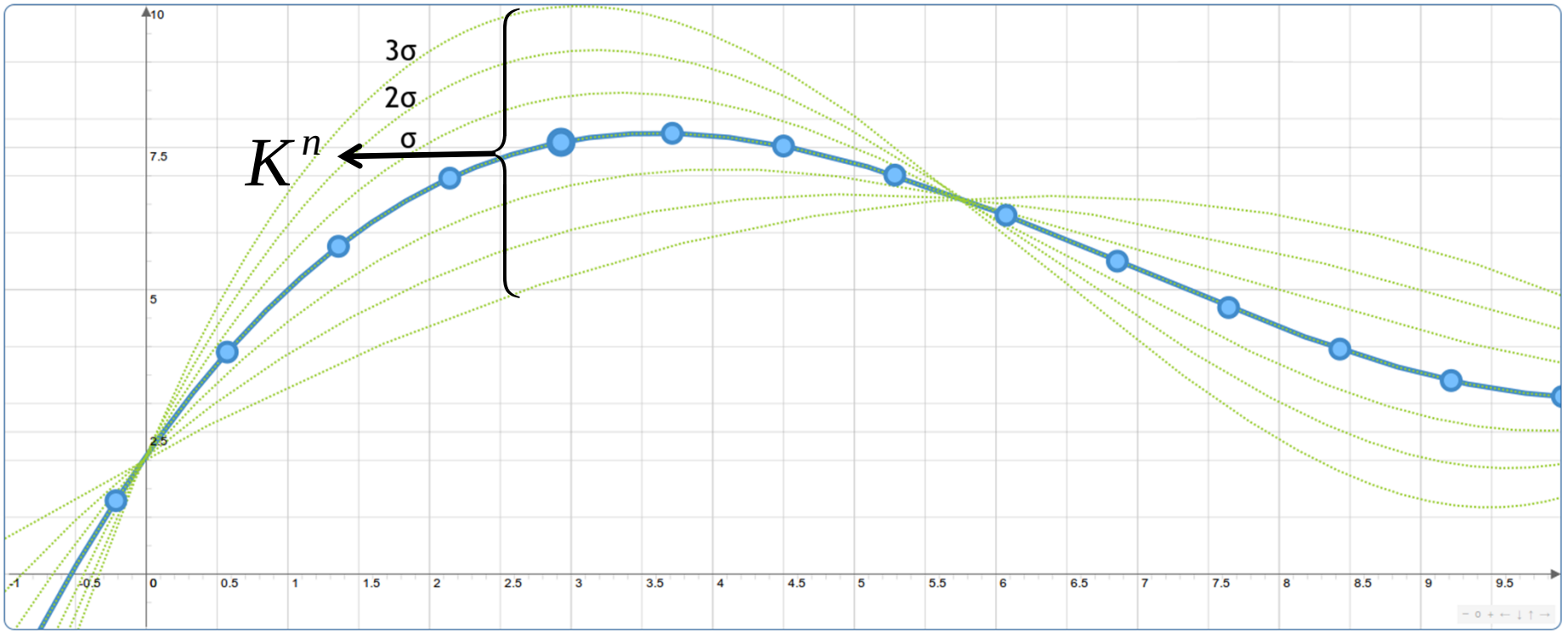
- Nonlinear, parametric belief model:
  - » For example, the following model describes the length of nanotubes in low temperatures:

$$L(t) = \frac{F_{c1}}{z}\left[\frac{k_{sb}\alpha S_0 n_m}{k_{cl}F_{c1}}\left(1-\exp\left(-\frac{F_{c1}k_{cl}}{\alpha S_0 n_m k_{sb}}t\right)\right)\right.$$
$$\left.-\frac{k_{sb}}{k_t(k_t-k_{sb})}\exp(-k_t t)+\frac{k_t}{k_{sb}(k_t-k_{sb})}\exp(-k_{sb}t)\right]$$

  - » We might enumerate a number of potential sets of values for all the parameters (known as "discrete priors")

The learning challenge

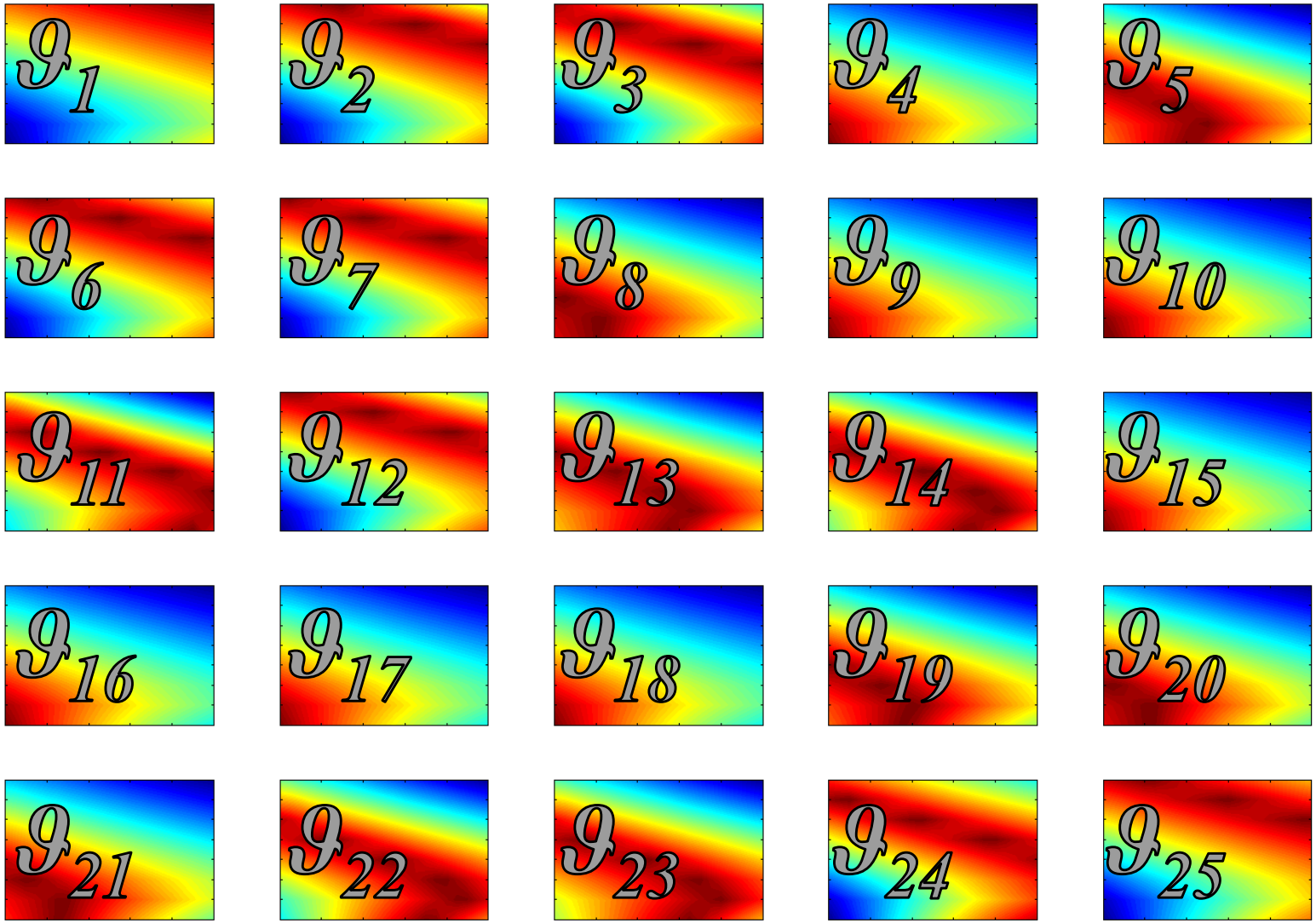
- We start by specifying our state of knowledge (say, after n experiments):



» Our state of knowledge may be a series of functions.

The learning challenge

- Set of possible truths:



Building a belief model

- Temperature dependence on chemical reaction rate k

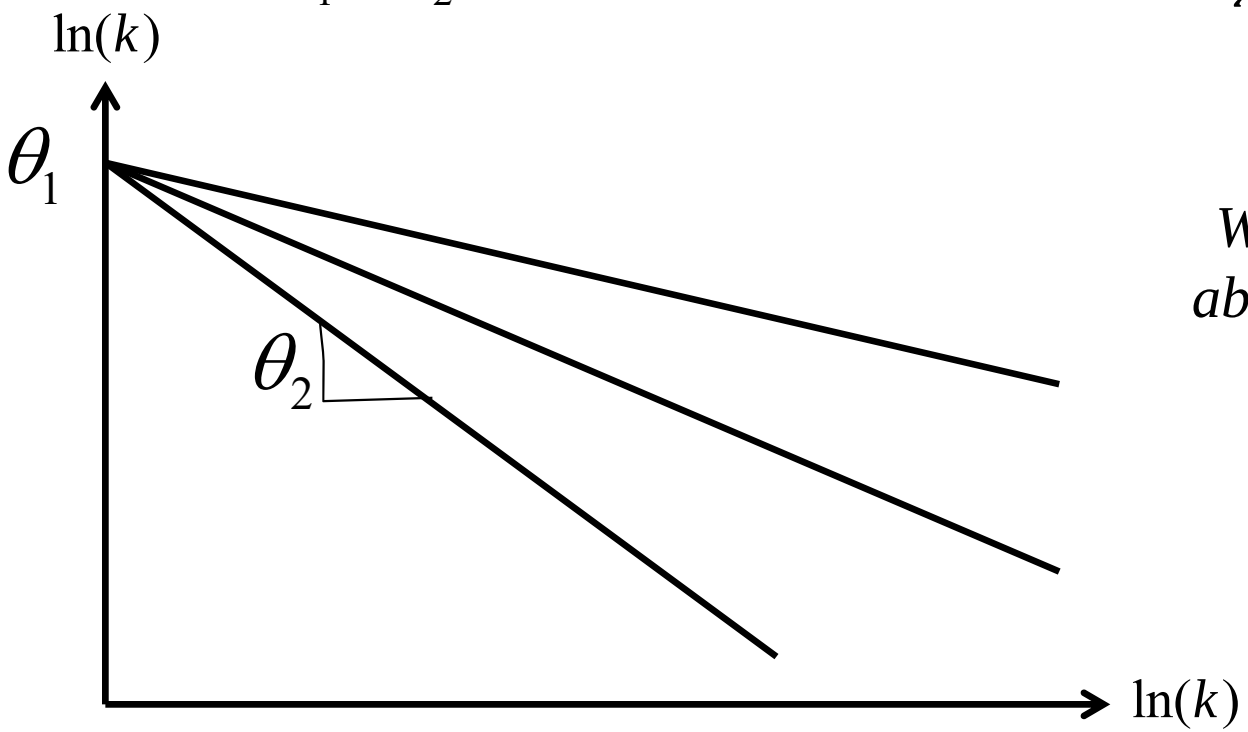
$$\ln(k) = \ln(A) - \frac{E_a}{k_B}\frac{1}{T}$$

$$= \theta_1 - \theta_2 x$$

$$\theta_1 = \ln(A)$$

$$\theta_2 = \frac{E_a}{k_b}$$

$$x = \frac{1}{T}$$



We might have beliefs about different slopes…

Building a belief model

- Temperature dependence on chemical reaction rate k

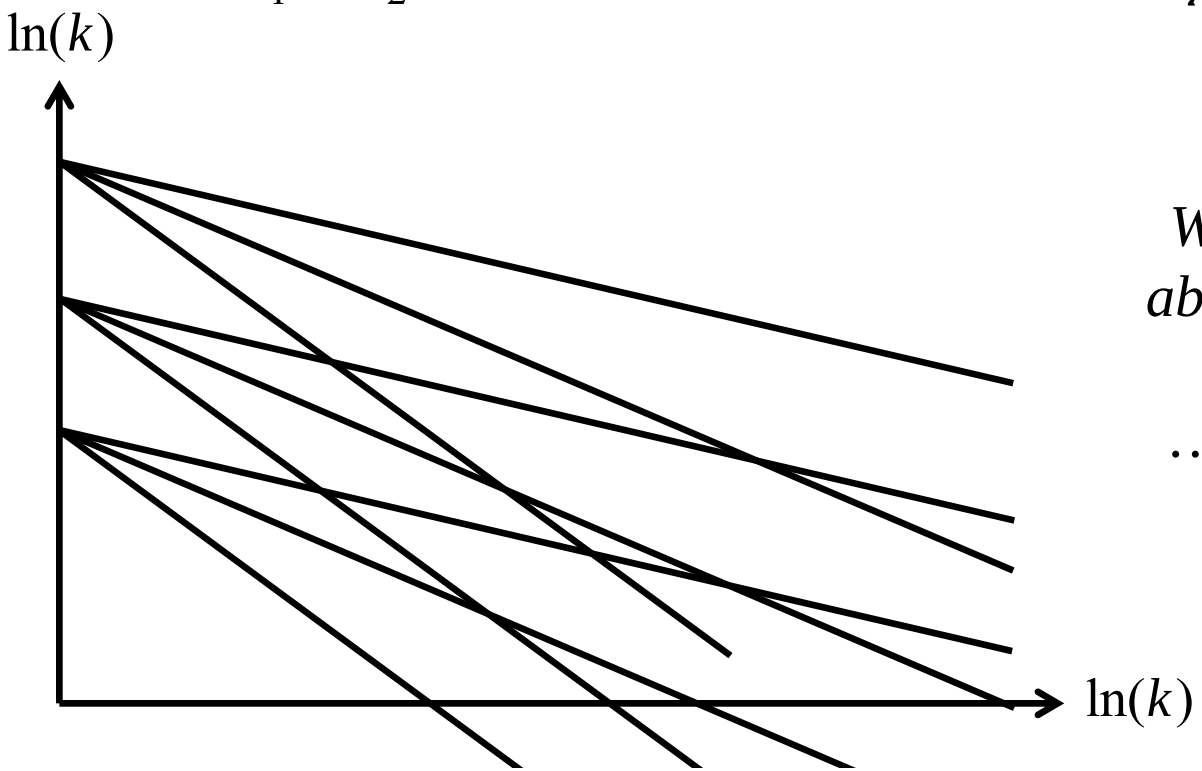
$$\ln(k) = \ln(A) - \frac{E_a}{k_B}\frac{1}{T}$$

$$= \theta_1 - \theta_2 x$$

$$\theta_1 = \ln(A)$$

$$\theta_2 = \frac{E_a}{k_b}$$

$$x = \frac{1}{T}$$



We might have beliefs about different slopes…

… as well as different intercepts.

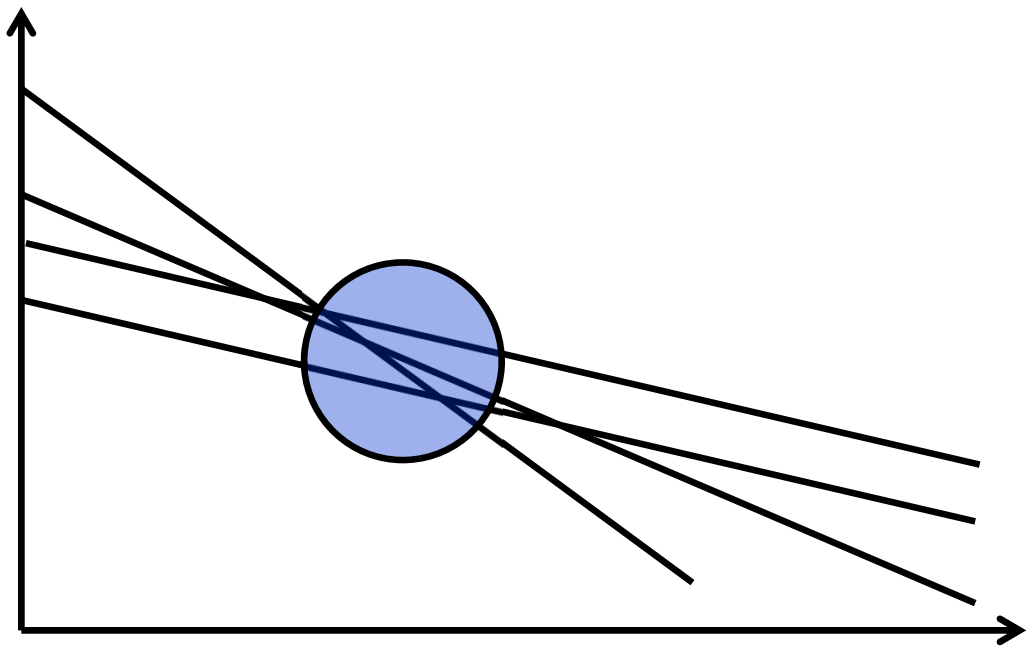
Building a belief model

- Temperature dependence on chemical reaction rate k

$$\ln(k) = \ln(A) - \frac{E_a}{k_B}\frac{1}{T}$$
$$= \theta_1 - \theta_2 x$$

$$\theta^0 = \begin{bmatrix} \theta_1^0 \\ \theta_2^0 \end{bmatrix}$$

$$\Sigma^{\theta,0} = \begin{bmatrix} \sigma_1^2 & \sigma_{12}^2 \\ \sigma_{12}^2 & \sigma_2^2 \end{bmatrix}$$



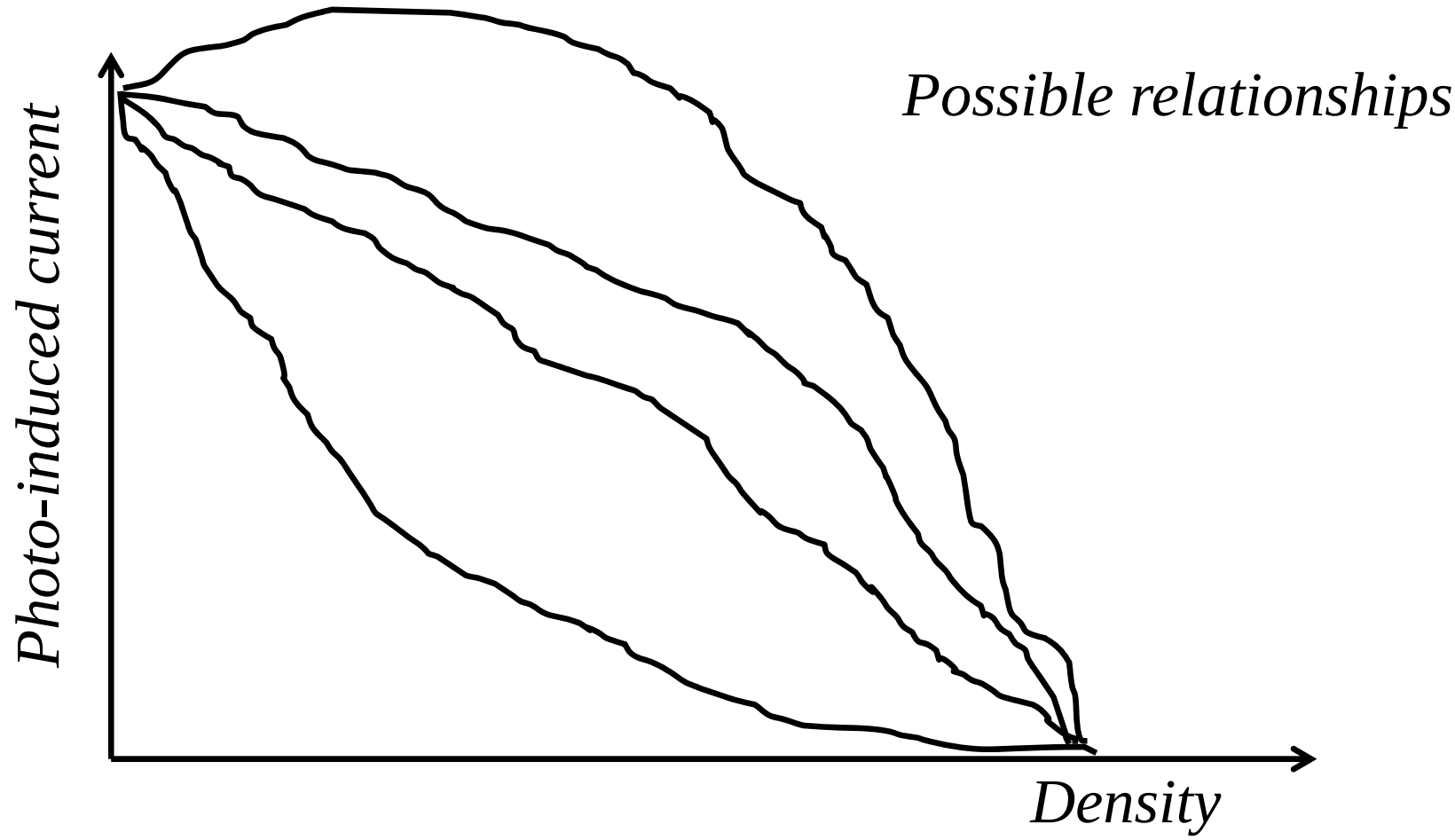
We might have beliefs about different slopes…

… as well as different intercepts.

But they are likely to be correlated.

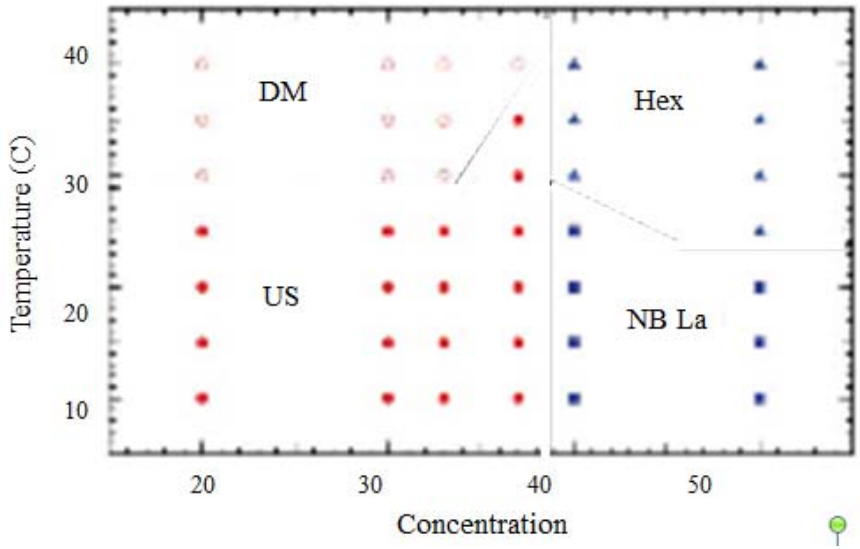
Building a belief model

- A prior can consist of a series of hand-drawn curves:



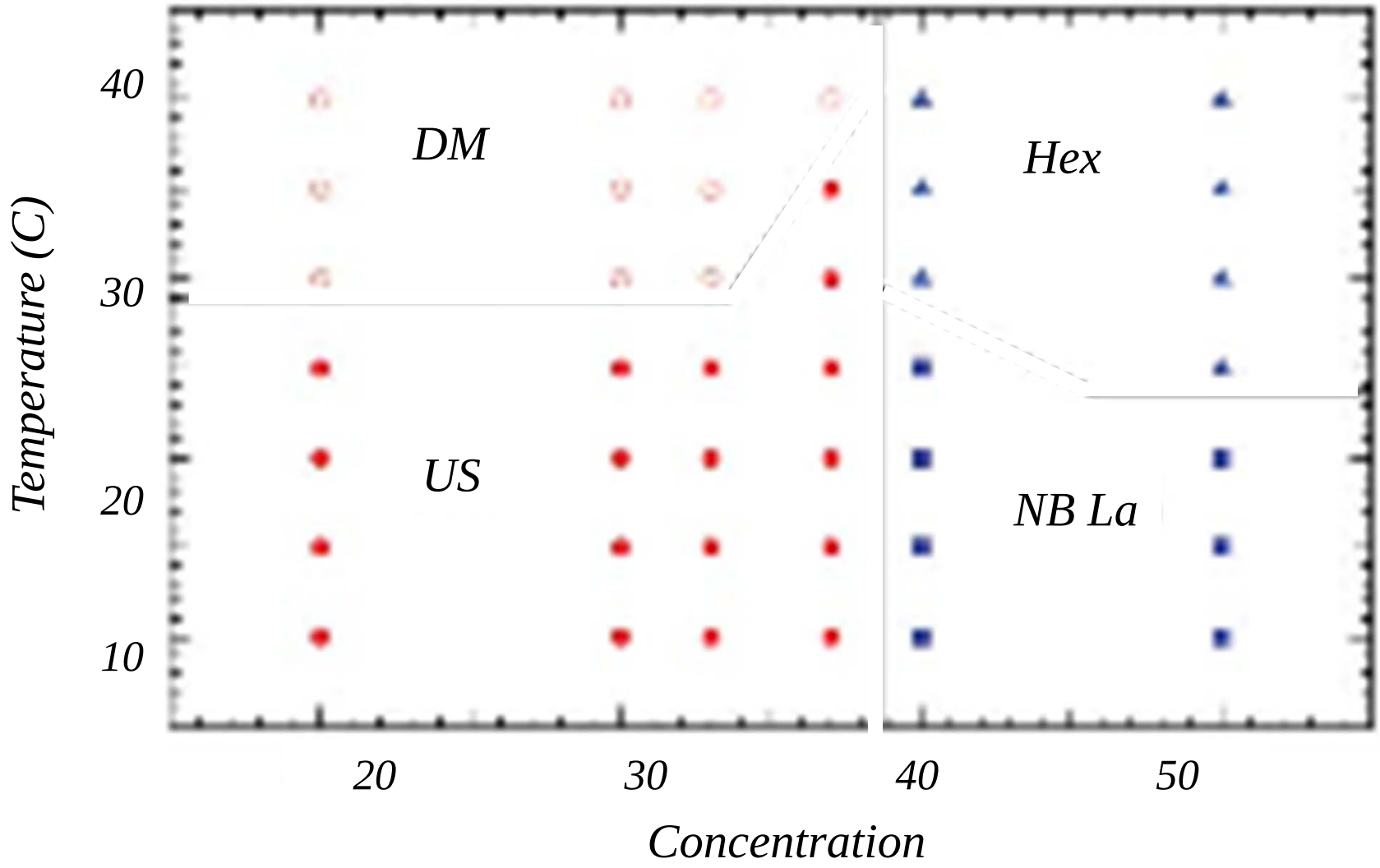
Building a belief model

- Classifying temperature/concentration regions
  - » Different regions produce different materials
  - » Each combination of temperature and concentration is a very expensive experiment.
  - » How do we minimize the number of experiments to come up with a good classification?



Building a belief model

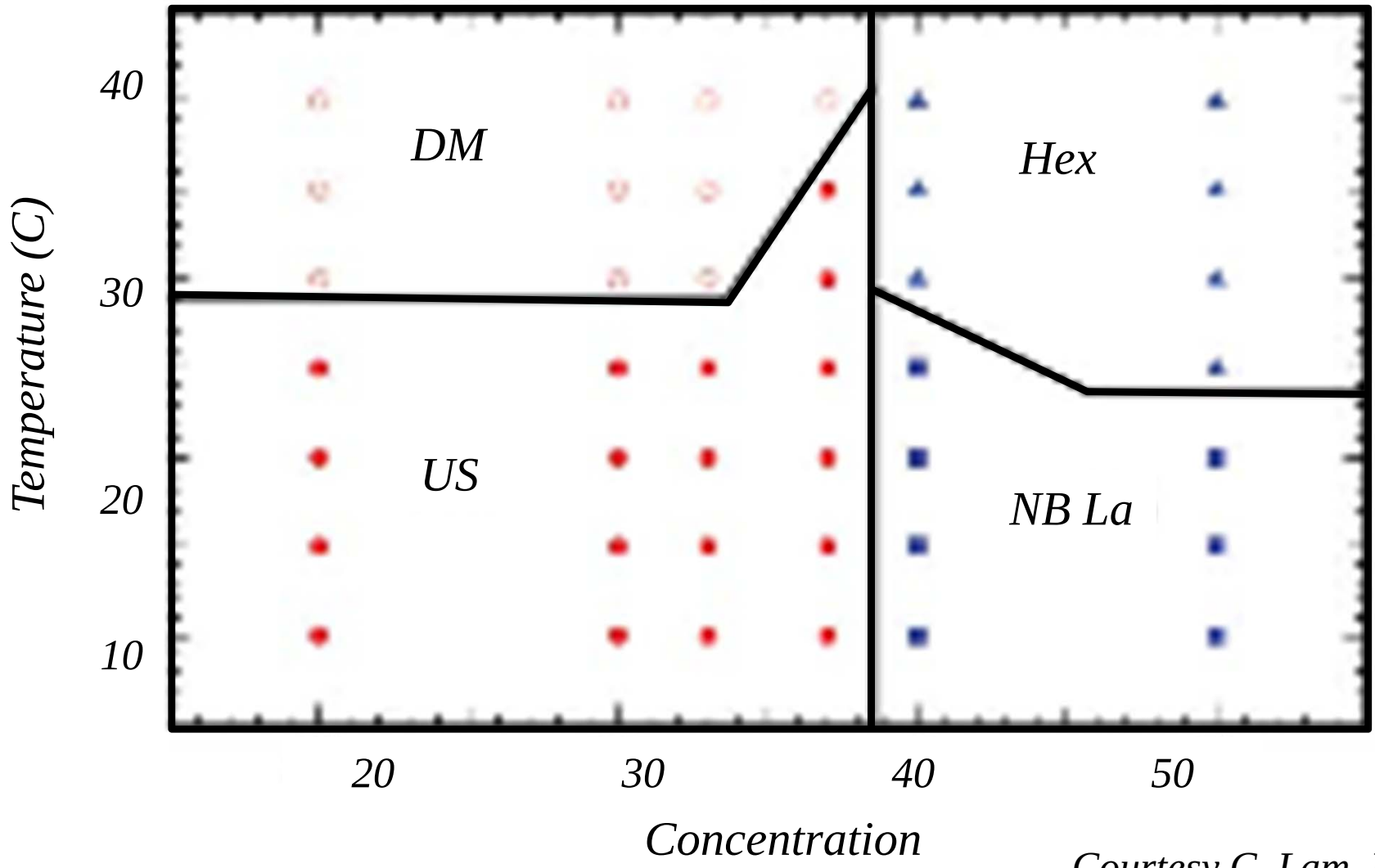
- Possible combinations we might run:



Courtesy C. Lam, R. Olsen

Building a belief model

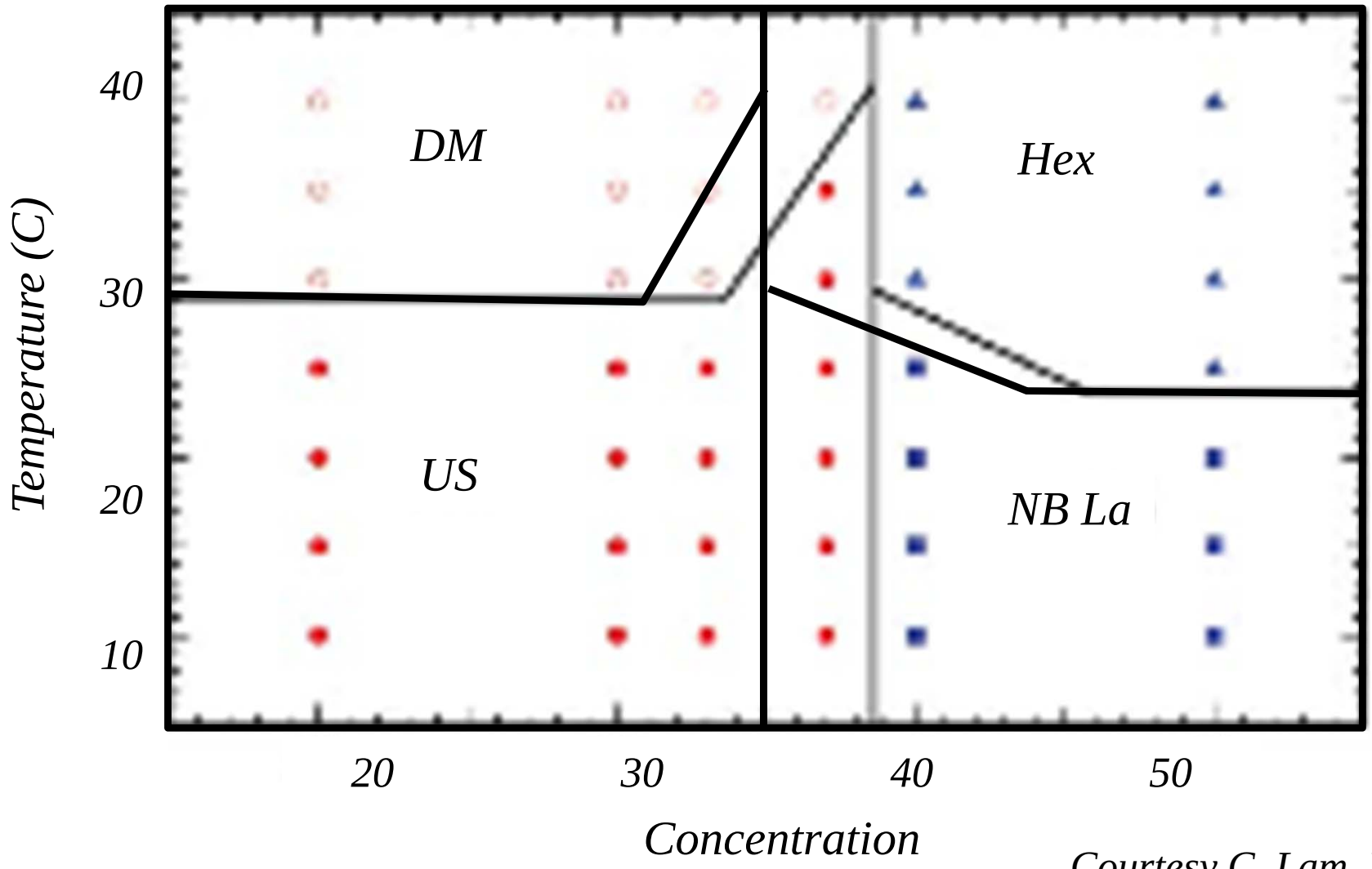
- One possible clustering:



Courtesy C. Lam, B. Olsen

Building a belief model

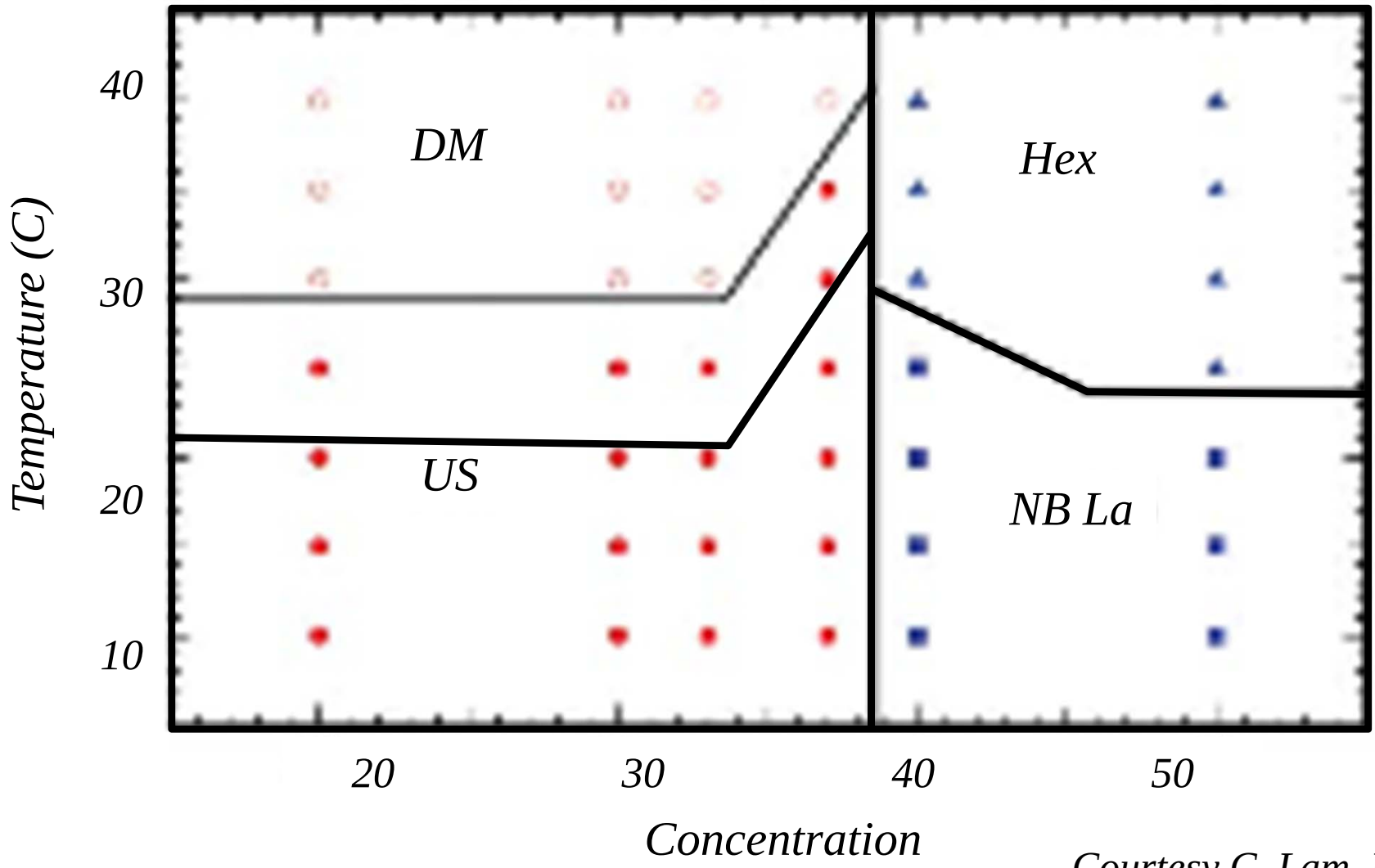
- Other clusterings:



Courtesy C. Lam, B. Olsen

Building a belief model

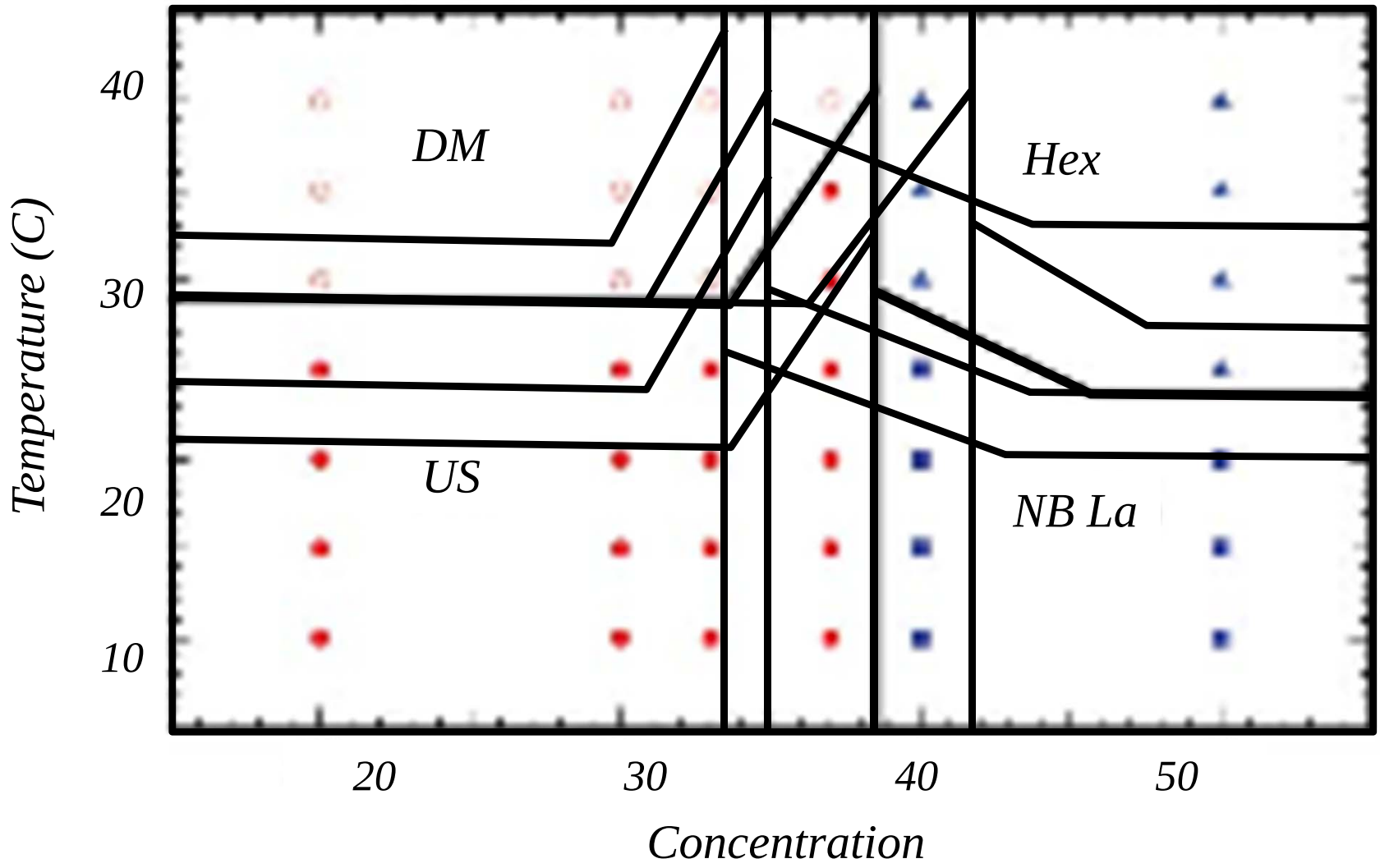
- Other clusterings:



Courtesy C. Lam, B. Olsen

Building a belief model

- Other clusterings:



Tutorial article in preparation:

Optimal Learning for Sequential Decisions in Laboratory Experimentation

Warren B. Powell, Nana Aboagye, Si Chen, Peter Frazier
Weidong Han, Xinyu He, Yan Li, Kris Reyes, Yingfei Wang

Department of Operations Research and Financial Engineering
Princeton University

Prepared for SIAM Review, June 2016

May 28, 2016