Foundations and Trends® in Technology,

Information and Operations Management

Sequential Decision Analytics and Modeling:

Modeling with Python

Suggested Citation: Warren Powell (2022), "Sequential Decision Analytics and Modeling:", Foundations and Trends® in Technology, Information and Operations Management: Vol. xx, No. xx, pp 1–. DOI: /XXXXXXXXX.

Warren B. Powell
wbpowell328@gmail.com

January 29, 2022

now
the essence of knowledge
Boston — Delft

Contents

Sequential Decision Analytics and Modeling:

Warren Powell[1]

[1] *Princeton University, Optimal Dynamics*

ABSTRACT

Sequential decision problems arise in virtually every human process, spanning finance, energy, transportation, health, e-commerce and supply chains, to name a few. An important dimension of every problem setting is the need to make decisions in the presence of different forms of uncertainty, and evolving information processes. This book uses a teach-by-example style to illustrate a modeling framework that can represent *any* sequential decision problem. A major challenge is, then, designing methods (called policies) for making decisions. We describe four classes of policies that are universal, in that they span any method that might be used, whether from the academic literature or heuristics used in practice. While this does not mean that we can immediately solve any problem, the framework helps us avoid the tendency in the academic literature of focusing on narrow classes of methods.

Warren Powell (2022), "Sequential Decision Analytics and Modeling:", Foundations and Trends® in Technology, Information and Operations Management: Vol. xx, No. xx, pp 1–. DOI: /XXXXXXXXX.

This is a work in progress. It is undergoing a major revision from an initial draft used to teach an undergraduate course at Princeton University. This edition has been updated up through chapter 1.

Please enter any comments at https://tinyurl.com/sdacomments. Be sure to include the date for the version that you are using.

I hope to submit the book in May, 2022, to NOW in their series: Foundations and Trends® in Technology, Information and Operations Management. However, the book will always be available as a free download in pdf form.

Warren Powell

January 29, 2022

1

Modeling sequential decision problems

Sequential decision problems can always be written as

$$decision,\ information,\ decision,\ information,\ decision, \ldots$$

Each time we make a decision we incur a cost or receive a contribution or reward (there are many ways to measure performance). Decisions have to be made before knowing the information that arrives in the future, and needs to consider the decisions that might be made as time progresses. We want to make good decisions over time, possibly with the ultimate goal of designing a system, so that it works well in the presence of different forms of uncertainty.

Sequential decision problems are ubiquitous, arising in virtually every human process. Table 1.1 provides a sample list of fields, with examples of some of the decisions that might arise. Most of these fields probably have many different types of decisions, ranging in complexity from when to sell an asset or to adopt a new web design, to choosing the best drug, material, or facility to design, to managing complex supply chains or managing a fleet of trucks.

Even more challenging than listing all the types of decisions is identifying the different sources of uncertainty that arise in many applications. Human behavior, markets, physical processes, transportation networks,

energy systems and the broad array of uncertainties that arise in health hint at the diversity of different sources of uncertainty.

As this book is being written, humanity is struggling with the spread of variations of COVID-19. Dealing with this pandemic has been described as "complex," but this is really a byproduct of a failure to think about the problem in a structured way. We are going to show the reader how to break down problems into a series of basic components lead to practical solutions.

Our approach starts by identifying some core elements such as performance metrics, decisions, and sources of uncertainty, which then lead to the creation of a mathematical model of the problem. The next step is usually (but not always) to implement the model on the computer, but there are going to be many problems where the process of building a computer model is impractical for any of a number of reasons. For this reason, we are also going to consider problems where we have to test and evaluate ideas in the field. To improve performance, we need to first learn how to make good decisions over time (this is how we control the system). Then, we turn to the design of the system.

At this time, the academic community has not adopted a standard modeling process for sequential decision problems. This is in sharp contrast with the arena of static, deterministic optimization problems which have followed a strict framework since the 1950s. Our modeling process is based on the presentation in **PowellRLSO2020**, which is a book aimed at a technical audience that is primarily interested in developing and implementing models on the computer.

By contrast, this book is aimed at a broader audience that is first and foremost interested in learning how to *think* about sequential decision problems. It uses a "teach-by-example" style that focuses on communicating the modeling process which we feel can be useful even without ultimately creating computer models. Central to our approach is the creation of a mathematical model that eliminates the ambiguity when describing problems in plain English. For readers who are interested in developing computer models, notation is the stepping stone to writing software. However, we are going to primarily use mathematical notation to create clarity when describing a problem, even if the reader never intends to write a line of code.

Field	Questions
Business	What products should we sell, with what features? Which supplies should you use? What price should you charge?
Economics	What interest rate should the Federal Reserve charge given the state of the economy? What levels of market liquidity should be provided?
Finance	What stocks should a portfolio invest in? How should a trader hedge a contract for potential downside?
Internet	What ads should we display to maximize adclicks? Which movies attract the most attention? When/how should mass notices be sent?
Engineering	How to design devices from aerosol cans to electric vehicles, bridges to transportation systems, transistors to computers?
Public health	How should we run testing to estimate the progression of a disease? How should vaccines be allocated? Which population groups should be targeted?
Medical research	What molecular configuration will produce the drug which kills the most cancer cells? What set of steps are required to produce single-walled nanotubes?
Supply chain management	When should we place an order for inventory from China? Which supplier should be used?
Freight transportation	Which driver should move a load? What loads should a truckload carrier commit to move? Where should drivers be domiciled?
Information collection	Where should we send a drone to collect information on wildfires or invasive species? What drug should we test to combat a disease?
Multiagent systems	How should a large company in an oligopolistic market bid on contracts, anticipating the response of its competitors?
Algorithms	What stepsize rule should we use in a search algorithm? How do we determine the next point to evaluate an expensive function?

Table 1.1: A sample of different fields and choices that need to be made within each field (adapted from **PowellRLSO2020**).

1.1 The modeling process

Modeling is an art, but it is art guided by a mathematical framework that ensures that we get a well-defined problem that we can put on the computer and solve. This can be viewed as building a bridge from a messy, poorly defined real-world problem to something with the clarity a computer can understand, even if your end goal is not to put it on the computer.

In this section, we are going to provide a very compact version of our modeling framework. Then (in section 1.2) we are going to illustrate the framework, initially using a very simple inventory problem (in section 1.2.1) but then introducing some modest extensions (in section 1.2.2). After presenting these examples, we are going to return in section 1.3 to a more detailed summary of our modeling framework.

The remaining chapters in this book use a series of applications to further develop the link between problem and mathematical model using a teach-by-example style. Our problem settings will grow in complexity, and while they are necessarily streamlined, we will be laying the foundation for modeling very complex problems, such as energy systems, vaccine distribution, trucking companies and supply chains.

1.1.1 A compact presentation of a dynamic model

We begin by observing that we can model any sequential decision problem using the sequence

$$(S_0, x_0, W_1, S_1, x_1, W_2, \ldots, S_t, x_t, W_{t+1}, \ldots, S_T),$$

where:

- S_t is the *state variable* that captures what we know at time t (it is best to think of S_t as the state of information or, more generally, the state of knowledge, at time t). State variables can capture the vaccines in inventory, the location of a car, weather forecasts, and beliefs about how the market will respond to price.

- x_t represents *decision variables* which capture the elements that we control, such as the whether to sell a house, the path through

a network, the choice of drug for a treatment, the price at which to sell a product, or the assignment which trucks should move different loads of freight.

- W_{t+1} is the information that arrives after we make the decision x_t, which might be the final selling price of a house, the travel times through the network, how a patient responds to a drug, the sales of a product at a price, and the loads of freight called in after we make initial assignments. The information in W_{t+1} comes from outside of our system, so we refer to it as *exogenous information.*

The decision x_t is determined by some method that we refer to as a *policy*, which we denote $X^\pi(S_t)$. We assume we have a *transition function* that takes as input the state S_t, decision x_t, and the exogenous information W_{t+1} and gives us the updated state S_{t+1}.

We incur a contribution (or cost) $C(S_t, x_t)$ when we make the decision $x_t = X^\pi(S_t)$. Our goal is to find the policy that maximizes some objective that depends on the contributions $C(S_t, x_t)$ where x_t.

There are many applications where it is more natural to index decisions using a counter n. In this case we would write

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

Here, n might be the n^{th} experiment, the n^{th} arrival of a customer, or the n^{th} iteration of an algorithm.

Finally, there are settings where we use both, as in $(S_t^n, x_t^n, W_{t+1}^n)$ to capture, for example, decisions in the n^{th} week at hour t.

This is a very compact description of a sequential decision problem. We next describe a set of steps to follow in the modeling process.

1.1.2 A six step modeling process

It is possible to divide the entire modeling process into six steps (for our purposes):

Step 1. The narrative - This will be a plain English discussion of the problem, often stepping through time (or events) to provide the reader with an advance description. The narrative will not

provide all the information needed to create a mathematical model; rather, it is a first step that should give the modeler the big picture without getting lost in notation.

Step 2. Identifying the core elements of the problem, with special emphasis on three dimensions of any sequential decision problem. These elements are described without using mathematics:

- What metrics are we trying to impact? Individual fields (such as supply chain management, health, energy, finance) will each be characterized by a number of metrics that might be described using terms such as costs, revenues, profits, rewards, gains, losses, performance and risk.
- What decisions are being made? Identifying decisions is quite easy for many problems such as computer games, but if we address a complex problem such as responding to a public health process, reducing the carbon footprint, or managing a supply chain, then identifying all the decisions can be quite challenging.
- What are the different sources of uncertainty? What are we uncertain about before we start? What information arrives exogenously over time (that is, arrives after we make a decision)?

Step 3. The mathematical model - Here we build off the first three elements from Step 2, but now we have to create a mathematical model that consists of five dimensions that apply to every sequential decision problem:

- State variables S_t - The state variable captures everything you need to know at time t to make a decision at time t, compute costs and constraints, and if necessary, simulate your way to time $t + 1$. State variables can include information about physical resources (inventories, the location of a vehicle), other information (prices, weather, forecasts), and beliefs about quantities and parameters we do not know perfectly.

- Decision variables x_t - These describe how we are going to design or control our system. Decisions have to satisfy constraints that we write as $x_t \in \mathcal{X}$ where $\mathcal{X}$ could be a set of discrete choices, or a set of linear equations. Decisions will be determined by *policies* that are functions (or rules) that we designate by $X^\pi(S_t)$ that determine x_t given what is in the state variable. Policies can be very simple (buy low, sell high) or quite complex. The index π carries information about the type of function $f \in \mathcal{F}$, and any tunable parameters $\theta \in \Theta^f$. We introduce the policy $X^\pi(S_t)$, but defer until later designing the policy.
- Exogenous information W_{t+1} - This is new information that arrives over time such as how much we sell after setting a price, or the time to complete the path we chose. When we make a decision at time t, the information in W_{t+1} is unknown, so we treat it as a random variable when we are choosing x_t.
- The transition function $S^M(S_t, x_t, W_{t+1})$ - These are the equations that describe how the state variables evolve over time. For many real problems, transition functions capture all the physics of the problem, and can be quite complex. In some cases, we do not even know the equations, and have to depend on just the state variables that we can observe. We write the evolution of the state variables S_t using our transition function as

  $$S_{t+1} = S^M(S_t, x_t, W_{t+1})$$

  where $S^M(\cdot)$ is known as the state (or system) transition model (hence the M in the superscript). The transition function describes how every element of the state variable changes given the decisions x_t and exogenous information W_{t+1}. In complex problems, the transition function may require thousands of lines of code to implement. This is where all the physics of a problem can be found.
- The objective function - This captures the performance metrics we use to evaluate our performance, and provides the

basis for searching over policies. We let

$C(S_t, x_t)$ = the contribution (if maximizing) or cost (if minimizing) of decision x_t which may depend on the information in S_t. There will be times when the contribution function itself depends on time, in which case we would write it as $C_t(S_t, x_t)$.

Our objective is to find the best policy $X^\pi(S_t)$ to optimize some metric such as:

- Maximize the expected sum of contributions over some horizon.
- Maximize the expected performance of a final design that we learned over a number of experiments or observations.
- Minimize the risk associated with a final design.

The transition from the real problem (guided by the narrative) to the elements of the mathematical model is perhaps the most difficult step, as it often involves soliciting information from a non-technical source.

Step 4. The uncertainty model - This is how we model the different types of uncertainty. We have three ways of modeling the exogenous information process:

- Create a mathematical model of $W_1, W_2, \ldots, W_T$.
- Use observations from history, such as past prices, sales, or weather events.
- Run the system in the field, observing W_t as they happen.

Step 5. Designing policies - Policies are functions, so we have to search for the best function. We do this by identifying two core strategies for designing policies:

- Search over a family of functions to find the one that works best, on average, over time.

- Create a policy by estimating the immediate cost or contribution of a decision x_t, plus an approximation of future costs or contributions.

Step 6. Evaluating policies - Finding the best policy means evaluating policies to determine which is best. Most often we assume policies can be evaluated in a simulator, but simulators can be complex and difficult to build, and are still subject to modeling approximations. For this reason, the vast majority of practical problems encountered in practice tend to involve testing in the field, which introduces its own set of issues.

The only way to become comfortable with a mathematical model is to see it applied in practice. It helps to use familiar examples, so we are going to start with a universal problem that we all encounter in every day life: managing inventories.

1.2 Some inventory problems

We are going to illustrate our modeling framework using two variations of a classic inventory problem, which is widely used as an application for illustrating certain methods for solving sequential decision problems. We are going to start with a simple inventory example that gets across the core elements of our modeling framework, but allows us to ignore many of the complexities that we will be exploring in the remainder of the book.

Then, we are going to transition to a *slightly* more complicated inventory problem that will allow us to illustrate some modeling principles. Throughout the book, we are also going to use the idea of starting with a basic version of a problem, and then introduce extensions that hint at the types of complications that can arise in real applications.

1.2.1 A simple inventory problem

One of the most familiar sequential decision problems that we all experience each time we visit a store is an inventory problem. We are going to use a simple version of this problem to illustrate the six steps of our modeling process that we introduced above:

Step 1: Narrative - A pizza restaurant has to decide how many pounds of sausage to order from its food distributor. The restaurant has to make the decision at the end of day t, communicate the order which then arrives the following morning to meet tomorrow's orders. If there is sausage left over, it can be held to the following day. The cost of the sausage, and the price that it will be sold for the next day, is known in advance, but the demand is not.

Step 2: Core elements - These are:

- Metrics - We want to maximize profits given by the sales of sausage minus the cost of purchasing the sausage.
- Decisions - We have to decide how much to order at the end of one day, arriving at the beginning of the next.
- Sources of uncertainty - The only source of uncertainty in this simple model is the demand for sausage the next day.

Step 3: - Mathematical model - This consists of five elements.

1) The state variable is our inventory which we are going to call R_t^{inv}. For now, this is the only element of the state variable, so $S_t = R_t^{inv}$ (later we are going to introduce additional elements to our state variable).

2) The decision variable x_t is how much we order at time t, which we assume (for now) arrives right away. We make our decisions with a policy $X^\pi(S_t)$ which we design later.

3) The exogenous information is the random demand for our product which we are going to denote $\hat{D}_{t+1}$, so $W_{t+1} = \hat{D}_{t+1}$.

4) Our transition function captures how the inventory R_t evolves over time, which is given by

$$R_{t+1}^{inv} = \max\{0, R_t^{inv} + x_t - \hat{D}_{t+1}\}. \tag{1.1}$$

5) Our objective function. For our inventory problem, it is most natural to compute the contribution including the purchase cost of product x_t and the revenue from satisfying the demand $\hat{D}_{t+1}$, which means that our single-period contribution

function would be written

$$C(S_t, x_t, \hat{D}_{t+1}) = -cx_t + p \min\{R_t + x_t, \hat{D}_{t+1}\},$$

where $x_t = X^\pi(S_t)$. Given a sequence of demands $\hat{D}_1, \ldots, \hat{D}_T$, the performance of a policy $\hat{F}^\pi$ would be

$$\hat{F}^\pi = \sum_{t=0}^{T} C(S_t, X^\pi(S_t), \hat{D}_{t+1}).$$

Our profits $\hat{F}^\pi$ are random because it depends on a particular sequence of random demands $\hat{D}_1, \ldots, \hat{D}_T$. We average over these random demands by taking an average, which we write by taking is *expectation*:

$$F^\pi = \mathbb{E}\left\{\sum_{t=0}^{T} C(S_t, X^\pi(S_t), \hat{D}_{t+1})|S_0\right\}. \tag{1.2}$$

Step 4. The uncertainty model - The simplest approach to modeling uncertainty is to just use historical data. The problem we might encounter is that if we run out of sausage, we might not observe the full demand for sausage that day. If we are able to capture this lost demand, then this is a reasonable approach.

An alternative is to build a mathematical model. We might assume that our demand is normally distributed with some mean $\bar{D}$ and standard deviation $\bar{\sigma}$. If we assume that both of these are known, we can write our demand as

$$\hat{D}_{t+1} \sim N(\bar{D}, \bar{\sigma}^2),$$

and take advantage of packages that can sample from the normal distribution (for example, in Excel this is called `Norm.inv`$(Rand(), \bar{D}, \bar{\sigma})$ to generate a random observation with mean $\bar{D}$ and standard deviation $\bar{\sigma}$.

Using this model, we can create a set of demands $(\hat{D}_1, \hat{D}_2, \ldots, \hat{D}_T)$. Then, we can repeat this N times to create N sequences of T demands, giving us the sequence $(\hat{D}_1^n, \hat{D}_2^n, \ldots, \hat{D}_T^n)$ for $n = 1, \ldots, N$ that we need to estimate the value of the policy (we use this below in Step 6).

Step 5. Designing policies - Next we have to design a method for determining our orders. A commonly used strategy for inventory problems is known as an "order-up-to" policy that looks like

$$X^{\pi}(S_t|\theta) = \begin{cases} \theta^{max} - R_t & \text{if } R_t < \theta^{min}, \\ 0 & \text{otherwise,} \end{cases} \tag{1.3}$$

where $\theta = (\theta^{min}, \theta^{max})$ is a set of parameters that need to be tuned

Step 6. Evaluating policies - There are a variety of strategies we might use. In practice, we cannot compute the expectation in the objective function in equation (1.2), so we take a series of samples of demands. Let $\hat{D}_1^n, \ldots, \hat{D}_T^n$ be one sample. Now we can estimate our expected profits from policy $X^{\pi}(S_t)$ by averaging over the samples for $n = 1, \ldots, N$, which is computed using

$$\overline{F}^{\pi}(\theta) = \frac{1}{N} \sum_{n=1}^{N} \sum_{t=0}^{T} C(S_t, X^{\pi}(S_t|\theta), \hat{D}_{t+1}^n).$$

In plain English, we are simulating the policy $X^{\pi}(S_t|\theta)$ N times using the simulated (or observed from history) samples of demands $\hat{D}_1^n, \ldots, \hat{D}_T^n$, and then averaging the performance to get $\overline{F}^{\pi}(\theta)$. We then face the problem of finding the best value of θ. A simple strategy would be to generate K possible values $\theta_1, \ldots, \theta_K$, simulating each one to find $\overline{F}^{\pi}(\theta_k)$ for each k, and then pick the value of θ_k that works the best. This is nowhere near an ideal strategy, but it provides a simple, practical starting point.

1.2.2 A slightly more complicated inventory problem

The simple inventory problem above is a classic setting for demonstrating a particular method for solving sequential decision problems known as dynamic programming, which depends on having a simple state variable that is a) discrete and b) does not have too many possible values. In our slightly more complicated inventory problem, we are going to illustrate three different flavors of state variables which would represent a serious complication for one popular method for solving sequential decision problems, but has no effect on several others.

Step 1: Narrative - We again have our pizza restaurant that has to order sausage, but we are going to allow the price we pay for sausage to vary from day to day, where we assume the price on one day is independent of the price on the previous day. Then, we are also going to assume that while the demand for sausage tomorrow is random, we are going to be given a forecast of tomorrow's demand that, while not perfect, is better than not having a forecast. Otherwise, everything about our more complicated problem is the same as it was before.

Step 2: Core elements - These are:

- Metrics - We want to maximize profits given by the sales of sausage minus the cost of purchasing the sausage, where the cost varies from day to day.
- Decisions - As with our simpler inventory problem, we have to decide how much to order at the end of one day, arriving at the beginning of the next.
- Sources of uncertainty - There are now three sources of uncertainty: the difference between the actual demand and the forecast, the evolution of the forecasts from one day to the next, and the price we pay for the sausage.

Step 3: - Mathematical model - We still have the same five elements, but now the problem is a bit richer.

1) To construct the state variable, we need to list the information (specifically, information that evolves over time) in three different places:

1) The objective function.
2) The policy for making decisions (which includes the constraints).
3) The transition function.

Of course, we have not yet introduced any of these functions, so you have to read forward, and verify that our state variable contains all the information needed to compute each of these

functions. Think of this as a dictionary of the information we will need.

We start with the initial state S_0 which consists of constant parameters, and initial values of quantities and parameters that change over time. These are:

- Price - We assume that we sell our sausage at a fixed price p.
- Initial inventory - We start with an initial inventory R_0.
- Initial forecast - We assume that our first forecast $f^D_{0,1}$ is given.
- Initial estimate of the variance of the demand - $\bar{\sigma}^D_0$.
- Initial estimate of the variance of the forecast - $\bar{\sigma}^f_0$.

We then have the information that evolves over time which makes up our dynamic state variable S_t:

- Current inventory R^{inv}_t - The inventory for the beginning of time interval $(t, t+1)$.
- Purchase cost c_t - This is the cost of sausage purchased at time t which is given to us at time t.
- Demand forecast $f^D_{t,t+1}$ - This is the forecast of $\hat{D}_{t+1}$.
- Current estimate of the variance of the demand - $\bar{\sigma}^D_t$.
- Current estimate of the variance of the forecast - $\bar{\sigma}^f_t$.

2) The decision variable x_t is how much we order at time t, which we assume (for now) arrives right away. We make our decisions with a policy $X^\pi(S_t)$ which we design later.

3) The exogenous information now consists of:

- Purchase costs $\hat{c}_t$ - This is the purchase cost of sausage on day t which is specified exogenously.
- Forecasts - Each time period we are given a new forecast. Let ε^f_{t+1} be the change in the forecast between time t and $t+1$.
- Demands - Finally, we assume that the actual demand is a random deviation from the forecast, which we could

write

$$\hat{D}_{t+1} = f^D_{t,t+1} + \varepsilon^D_{t+1}.$$

Our complete set of exogenous information variables can now be written

$$W_{t+1} = (\hat{c}_{t+1}, \varepsilon^f_{t+1}, \varepsilon^D_{t+1}).$$

4) Transition function - This specifies how each of the (dynamic) state variables S_t evolve over time. We update our inventory using:

$$R^{inv}_{t+1} = \max\{0, R^{inv}_t + x_t - \hat{D}_{t+1}\}. \tag{1.4}$$

The demand is the forecasted demand plus the deviation ε^D_{t+1} from the forecast, giving us the equation:

$$\hat{D}_{t+1} = f^D_{t,t+1} + \varepsilon^D_{t+1}. \tag{1.5}$$

We assume that our forecast is updated using

$$f^D_{t+1,t+2} = (1-\alpha)\hat{D}^t + \alpha f^D_{t,t+1} + \varepsilon^f_{t+1}. \tag{1.6}$$

Next, we are going to adaptively estimate the variance in the demand and the demand forecast:

$$\bar{\sigma}^D_{t+1} = (1-\alpha)\bar{\sigma}^D_{t+1} + (1-\alpha)(f^D_{t,t+1} - \hat{D}_{t+1})^2, \tag{1.7}$$

$$\bar{\sigma}^f_{t+1} = (1-\alpha)\bar{\sigma}^f_{t+1} + (1-\alpha)(f^D_{t,t+1} - f^D_{t+1,t+2})^2. \tag{1.8}$$

Finally, we update the cost c_{t+1} with the "observed cost" $\hat{c}_{t+1}$ which we write simply as

$$c_{t+1} = \hat{c}_{t+1}. \tag{1.9}$$

Our transition function

$$S_{t+1} = S^M(S_t, x_t, W_{t+1})$$

consists of the equations (1.4) - (1.9).

5) Finally, our single period contribution function would now be written

$$C(S_t, x_t, \hat{D}_{t+1}) = -c_t x_t + p \min\{R_t + x_t, \hat{D}_{t+1}\},$$

where the only difference with the simpler inventory problem is that the cost c is now time-dependent c_t. We now state our objective function formally as

$$\max_{\pi} \mathbb{E}\left\{\sum_{t=0}^{T} C(S_t, X^{\pi}(S_t), \hat{D}_{t+1})|S_0\right\}. \tag{1.10}$$

Step 4. The uncertainty model - We are going to assume that the exogenous changes ε_{t+1}^{D} and ε_{t+1}^{f} are described by normal distributions with mean 0 and variances $\bar{\sigma}_t^D$ and $\bar{\sigma}_t^f$, which we express by writing

$$\begin{aligned}\varepsilon_t^D &\sim N(0, \bar{\sigma}_t^D),\\ \varepsilon_t^f &\sim N(0, \bar{\sigma}_t^f).\end{aligned}$$

Uncertainty models can become quite complex, but this will serve as an illustration.

Step 5. Designing policies - Next we have to design a method for determining our orders. Instead of the order-up-to policy of our simpler model, we are going to suggest the idea of ordering enough to meet the expected demand for tomorrow, with an adjustment. We could write this as

$$X^{\pi}(S_t|\theta) = \max\{0, f_{t,t+1}^{D} - R_t\} + \theta. \tag{1.11}$$

If we had a perfect forecast, then all we have to order would be $f_{t,t+1}^{D}$ (our forecast of $\hat{D}_{t+1}$) minus the on-hand inventory. However, because of uncertainty we are going to add an adjustment (presumably $\theta > 0$, but this depends on the sales price relative to the purchase cost).

Step 6. Evaluating policies - This time we have to generate samples of all the random variables in the sequence $W_1, W_2, \ldots, W_T$. Again

we might generate n samples of the entire sequence so we can estimate the performance of a policy using

$$\overline{F}^{\pi}(\theta) = \frac{1}{N}\sum_{n=1}^{N}\sum_{t=0}^{T} C(S_t, X^{\pi}(S_t|\theta), \hat{D}^n_{t+1}).$$

We again face the problem of finding the best value of θ, but we return to that challenge later.

1.3 The mathematical modeling framework

Section 1.1 provided a streamlined summary of the modeling process, but did not provide some of the details needed to tackle more complex problems than the simple illustrations in section 1.2. Below we provide a more detailed summary of the elements of a sequential decision problem.

State variables - The state S_t of the system at time t has all the that is necessary and sufficient to model our system from time t onward. This means it has to capture the information needed to compute costs/contributions, constraints on decisions, and the evolution of this information over time in the transition function. In fact, it is often best to write out the objective function (with costs/contributions), the constraints for the decisions, and the transition function, and then return to define the state variables.

There will be many settings where it makes more sense to use a counter n rather than time. In this case, we let S^n be the state after n observations (these may be experiments, customer arrives, iterations of an algorithm). We will use time t as our default index.

We need to distinguish between the initial state S_0 and subsequent states S_t for $t > 0$:

S_0 - The initial state S_0 captures i) deterministic parameters that never change, ii) initial values of quantities or parameters that do change (possibly due to decisions), and iii) beliefs about quantities or parameters that we do not know perfectly (this might be the parameters of a probability distribution) such as how we respond to a vaccine or how the market will respond to price.

S_t - This is all the information we need at time t from history to model the system from time t onward.

There are three types of information in S_t:

- The physical state, R_t, which in most (but not all) applications is the state variable that is being controlled. R_t may be a scalar, or a vector with element R_{ti} where i could be a type of resource (e.g. a blood type) or the amount of inventory at location i. In our inventory problems, the physical state variables would be the inventory R_t and the demands $\hat{D}_t$.
- Other information, I_t, which is any information that is known deterministically not included in R_t, such as prices or demand forecasts. The information state often evolves exogenously, but may depend on decisions.
- The belief state, B_t, which contains information about a probability distribution describing one or more unknown parameters. Thus, B_t could capture the mean and variance of a normal distribution (as with our demand forecast above). Alternatively, it could be a vector of probabilities that evolve over time.

We note that the distinction of "physical state" R_t versus "other information" I_t is somewhat imprecise, but there are many problems that involve the management of physical (and financial) resources where the state of the physical resources (inventories, location of a vehicle, location and speed of a robot) is a natural "state variable" (in fact, a common error is to assume that the state of physical resources is the only state variable). By contrast, the "other information" I_t is literally any other information that does not seem to belong in R_t.

For example, R_t might be the amount of money in a cash account, while I_t might be the current state of the stock and bond markets. If we are traveling over a dynamic network, R_t might be our location on the network, while I_t could be what we know about the travel times over each link. If we plan a path and then wish to

penalize deviations from the plan, then the plan would be included in the state variable through I_t.

State variables typically are not obvious. They emerge during the modeling process, rather than something you can just immediately write out. Just because we write it first does not mean that you will always be able to list all the elements of the state variable right away. But in the end, this is where you store all the information you need to model your system from time t onward.

Decision variables - Different communities use different notations for decision, such as a_t for a (typically discrete) action or u_t for a (typically continuous) control in engineering. We use x_t as our default since it is widely used by the math programming community.

Decisions may be binary (e.g. for modeling when to sell an asset, or for A/B testing of different web designs), discrete (e.g. choice of drug, which product to advertise), continuous (prices, temperatures, concentrations), vectors (discrete or continuous such as allocations of blood supplies among hospitals), and categorical (e.g. what features to highlight in a product advertisement). We note that entire fields of research are sometimes distinguished by the nature of the decision variable.

We assume that decisions are made with a policy, which we might denote $X^\pi(S_t)$ if we use x_t as our decision. We assume that a decision $x_t = X^\pi(S_t)$ is feasible at time t, which means $x_t \in \mathcal{X}_t$ for some set (or region) $\mathcal{X}_t$, which may depend on S_t.

We let "π" carry the information about the type of function $f \in \mathcal{F}$ (for example, a linear model with specific explanatory variables, or a particular nonlinear model), and any tunable parameters $\theta \in \Theta^f$.

Exogenous information - We let W_{t+1} be any new information that first becomes known at time $t+1$ (that is, between t and $t+1$), where the source of the information is from outside of our system (which is why it is "exogenous"). When modeling specific variables,

we use "hats" to indicate exogenous information. Thus, $\hat{D}_{t+1}$ could be the demand that arises between t and $t+1$, or we could let $\hat{p}_{t+1}$ be the change in the price between t and $t+1$.

The exogenous information process may be stationary or nonstationary, purely exogenous or state (and possibly action) dependent (if we decide to sell a lot of stock, it could push prices down). We let ω represent a sample path $W_1, \ldots, W_T$, which represents a sequence of outcomes of each W_t. Often, we will create a set Ω with a set of discrete samples that represent a particular sequence of the outcomes of our W_t process which we could write as $W_1(\omega), \ldots, W_T(\omega)$. If we have 20 sample paths, we can think of ω as consisting of a number between 1 and 20, which allows us to look up the sample path.

Transition function - We denote the transition function by

$$S_{t+1} = S^M(S_t, x_t, W_{t+1}), \tag{1.12}$$

where $S^M(\cdot)$ is also known by names such as system model, plant model, plant equation, state equation, and transfer function. Equation (1.12) is the classical form of a transition function which gives the equations from the pre-decision state S_t to pre-decision state S_{t+1}. Equation (1.1) was the only transition equation for our simple inventory example, while equations (1.4) - (1.9) made up the transition function for our more complicated example.

The transition function might capture any of the following types of updates:

- Changes in physical resources such as adding inventory, moving people, or modifying equipment.
- Updates to information such as changes in prices and weather.
- Updates to our beliefs about uncertain quantities or parameters.

The transition function may be a known set of equations, or unknown, such as when we describe human behavior or the evolution of CO2 in the atmosphere. When the equations are unknown

the problem is often described as "model free" or "data driven" which means we can only observe changes in a variable, rather than using a physical model. Equation (1.9), where we "observe" the cost $c_{t+1} = \hat{c}_{t+1}$, with no idea how we evolved from c_t, is an example of a model free transition.

Transition functions may be linear, continuous nonlinear or step functions. When the state S_t includes a belief state B_t, then the transition function has to include the updating equations (we illustrate this later in the book).

Given a policy $X^\pi(S_t)$, an exogenous process W_{t+1} and a transition function, we can write our sequence of states, decisions, and information as

$$(S_0, x_0, W_1, S_1, x_1, W_2, \ldots, x_{T-1}, W_T, S_T).$$

Objective functions - There are a number of ways to write objective functions. We will write our generic objective function as

$$\max_\pi \mathbb{E}\left\{\sum_{t=0}^{T} C_t(S_t, X_t^\pi(S_t))|S_0\right\}, \tag{1.13}$$

where

$$S_{t+1} = S^M(S_t, X_t^\pi(S_t), W_{t+1}). \tag{1.14}$$

The model is fully specified when we also have a model of the initial state S_0, and a model of the exogenous process $W_1, W_2, \ldots$. We write all the exogenous information as

$$(S_0, W_1, W_2, \ldots, W_T). \tag{1.15}$$

Equations (1.13), (1.14) and (1.15) constitute a model of a sequential decision problem.

Note that the objective function depends on the initial state S_0. As described earlier, the initial state includes any deterministic parameters, initial values of dynamic parameters (such as inventory or the initial location of a drone), or the initial values of a probability distribution

describing an uncertain quantity (such as the health of a patient). Whatever we compute using the objective function in (1.13) depends, at least implicitly, on the information in S_0.

At this point, we have a model, but we are not done. We still have two major steps remaining:

- Modeling uncertainty - Typically we describe the elements of the initial state S_0, and the exogenous information process $W_1, \ldots, W_T$, but we have to find some way to generate samples from the underlying probability distributions.
- Designing policies - This will be a focal point of this book. When we described our inventory problems, we suggested very simple policies just to provide an idea of a policy. However, our standard method is to "model first, then solve" which means to specify the model using the function $X^\pi(S_t)$ as a placeholder. Then, once we have created our model (including a method for drawing samples from our initial state S_0 and the exogenous information process $W_1, \ldots, W_T$), we turn to the challenge of designing policies.

1.4 Modeling uncertainty

For many complex problems (supply chains, energy systems, and public health are just a few), identifying and modeling the different forms of uncertainty can be a rich and complex exercise. We are going to hint at the issues that arise, but we are not going to attempt a thorough discussion of this dimension.

Uncertainty is communicated to our model through two mechanisms: the initial state S_0, which is where we would model the parameters of probability distributions describing quantities and parameters that we do not know perfectly, and the exogenous information process $W_1, \ldots, W_T$.

1.4.1 The initial state variables S_0

The initial state variable may contain deterministic parameters or initial values of dynamically varying quantities and parameters. If this is all that is in the initial state, then it is not capturing any form of uncertainty.

There are many problems where we do not know some quantities or parameters, but can represent what we do know through the parameters of a probability distribution. Some examples are:

- The response of a patient to a new medication.
- How a market will respond to a change in price.
- How many salable heads of lettuce we have in inventory (an uncertain number may have wilted and are no longer salable).
- The time at which a box of inventory previously ordered from China will arrive.
- The amount of deposits to a mutual fund vary randomly around a mean λ, but we do not know what λ is.

These are a small sample of ways that we can initialize a model with uncertainty in some of the inputs.

An initial probabilistic belief may come from subjective judgment, or from previous observations or experiments.

1.4.2 The exogenous information process

The second way uncertainty enters our model is through the exogenous information process. The variable W_t contains information that is not known until time period t. This means we have to make a decision x_t at time t before we know the outcome of $W_{t+1}, W_{t+2}, \ldots, W_T$.

Below is a list of examples of W_{t+1} that are revealed after a decision x_t is made:

- We choose a path, and then observe the travel time on the path.
- We choose a drug, and then observe how the patient responds.
- We choose a catalyst, and then observe the strength of the material that it is used to produce.
- We choose a product to advertise in an online market, and then observe the sales.

- We select a web interface design, and then observe the number of clicks that it can generate.
- We allocate funds into an investment, and then observe the change in the price of the investment.

In each case, the information we observe after we make the decision affects the performance of the decision (and which decision would have been best).

We note that the information W_{t+1} helps to determine the state S_{t+1}, which in turn is used to make the following decision x_{t+1}. This means that the decision x_{t+1} is uncertain at time t when we are making our decision x_t.

1.4.3 Testing policies

To test the value of a policy, we need a way to generate samples of the exogenous information process. We let ω be a sample path, where $W_1(\omega), \ldots, W_T(\omega)$ represents a particular sample path. Table 1.2 illustrates 10 sample paths of prices that are indexed ω^1 to ω^10. If we choose ω^6, then

$$W_7(\omega^6) = 44.16.$$

	$t=1$	$t=2$	$t=3$	$t=4$	$t=5$	$t=6$	$t=7$	$t=8$
ω^n	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8
ω^1	45.00	45.53	47.07	47.56	47.80	48.43	46.93	46.57
ω^2	45.00	43.15	42.51	40.51	41.50	41.00	39.16	41.11
ω^3	45.00	45.16	45.37	44.30	45.35	47.23	47.35	46.30
ω^4	45.00	45.67	46.18	46.22	45.69	44.24	43.77	43.57
ω^5	45.00	46.32	46.14	46.53	44.84	45.17	44.92	46.09
ω^6	45.00	44.70	43.05	43.77	42.61	44.32	44.16	45.29
ω^7	45.00	43.67	43.14	44.78	43.12	42.36	41.60	40.83
ω^8	45.00	44.98	44.53	45.42	46.43	47.67	47.68	49.03
ω^9	45.00	44.57	45.99	47.38	45.51	46.27	46.02	45.09
ω^{10}	45.00	45.01	46.73	46.08	47.40	49.14	49.03	48.74

Table 1.2: Illustration of a set of sample paths for prices all starting at $45.00.

The question is: how do we create a sample of observations such as those depicted in table 1.2? There are three typical strategies:

- Create samples from historical data. Since there is only one outcome at any point in time, we can create multiple sample paths by combining observations from different periods of time. We might pick prices from different years, or demands from different months, or observed travel times on different days.

- We can test an idea in the field, using observations as they actually occur. The advantage of this is that we are working with real data (history may not be the same as the future). The disadvantage is that it takes a day to observe a day of new data (and we may need much more than a single day of observations).
  item Simulating from a mathematical model. This approach offers the advantage of being able to generate large samples to get statistically reliable estimates of the performance of a policy. These models can be quite sophisticated, but it is quite easy to create models (even sophisticated ones) that do not replicate the behavior of real data. The biggest challenge is capturing correlations, either over time or between samples (say the demands of different products, the prices of different stocks, or the wind speed in different locations).

We refer the reader to **PowellRLSO2020**[Chapter 10] for a more in-depth discussion of uncertainty modeling.

1.5 Designing policies

Throughout the applications in the chapters that follow, we will always be trying to find a good policy, which might be the best in a class, or sometimes the best over all (the true optimal policy). Figure 1.1 shows the front covers of books representing roughly 15 distinct fields that all deal with sequential decisions under uncertainty. These books are largely distinguished by how they approach creating policies, along differences in problem characteristics.

We are going to build on the idea that there are two fundamental strategies for creating policies, each of which can then be further divided into two classes, creating four classes of policies:

Figure 1.1: A sampling of major books representing different fields in stochastic optimization.

Policy search - This is where you search across methods (functions) for making decisions, simulating their performance (as we do in equation (1.13)), to find the method that works best on average over time. This may involve searching over different classes of methods, as well as any tunable parameters for a given method. This idea opens up two classes of policies:

1) Policy function approximations (PFAs) - These are analytical functions of a state that directly specify an action. The order-up-to policy in equation (1.3) is a good example, along with our policy of using an adjusted forecast in equation (1.11).

2) Cost function approximations (CFAs) -

Lookahead policies - We can build effective policies by optimizing across the contribution (or cost) of a decision, plus an approximation of the downstream contributions (or cost) resulting from the decision made now. Again, we can divide these into two more classes of policies:

3) Value function approximations (VFAs) - If we are in a state S_t, make a decision x_t and then observe new information W_{t+1}, it will take us to a new state S_{t+1} according to our transition function

$$S_{t+1} = S^M(S_t, x_t, W_{t+1}). \tag{1.16}$$

Now, let's assume that we have a way of approximating all the downstream contributions (or costs) that will happen from the decisions we make, and new information that we observe, starting at time $t+1$ when we are in state S_{t+1}, which is a random variable when we are sitting at time t because we do not yet know the information W_{t+1} that arrives immediately after we choose decision x_t.

Let's call our estimate of all these future contributions the value of being in state S_{t+1}, and we designate it with the function $\overline{V}_{t+1}(S_{t+1})$. We see from the transition function in (1.16) that the state S_{t+1} depends on S_t, the decision x_t and the new information W_{t+1}. We would then make our decision by solving

$$X^{\pi}(S_t) = \arg\max_{x_t \in \mathcal{X}_t} \left(C(S_t, x_t) + \mathbb{E}\{\overline{V}_{t+1}(S_{t+1})|S_t, x_t\}\right). \quad (1.17)$$

Here, the expectation $\mathbb{E}$ is over any uncertain quantity, that includes the information that has not yet arrived, W_{t+1} and any uncertain quantities in our state variable S_t (such as the uncertainty in our forecast).

This class of policy falls under headings such as approximate dynamic programming and, most often, reinforcement learning. While a powerful idea, it is not easy to apply and depends on our ability to create an approximation $\overline{V}_{t+1}(S_{t+1})$

4) Direct lookahead approximations (DLAs) - There are many problems where we simply cannot develop effective policies using any of the first three classes, and when this happens, we have to turn to direct lookahead approximations. We will write this out in its full mathematical form later, but for now, we are going to describe DLAs as making a decision now while optimizing over a (typically approximate) model that extends over some planning horizon.

A common DLA is to create an approximate model that is deterministic. This is what we are doing when we use a navigation system that finds the shortest path to the destination assuming that we know the travel time along each link of the

network. As a general rule, solving an accurate stochastic model of the future is almost always impossible, so we are going to investigate different strategies for approximating the problem.

We illustrated our modeling framework in section 1.2 using two inventory problems, and suggested two simple policies (forms of PFAs) with equations (1.3) and (1.11), but we did this just to have a concrete example of a policy. However, there is no chance that we would be able to take these policies to other problems in the vast collection of sequential decision problems.

By contrast, we are going to claim that the four classes of policies we just outlined (PFAs, CFAs, VFAs and DLAs) is universal, in that these cover *any* method that we might use to solve *any* sequential decision problem. To be clear, these are meta-classes. That is, if we think a problem lends itself to one particular class, we are not done, since we still have to design the specific policy within the class. Just the same, we feel that these four classes provides a roadmap to guide the process for designing policies.

1.6 Next steps

The next five chapters of the book are going to apply our modeling framework to five different problems:

- Chapter 2 - An asset selling problem
- Chapter 3 - Adaptive market planning
- Chapter 4 - Learning the best diabetes medication
- Chapter 5 - Stochastic shortest path problems - Static
- Chapter 6 - Stochastic shortest path problems - Dynamic

Each of these chapters will follow the same outline as we used in section 1.2 to describe the two inventory problems. This outline consists of:

- Narrative - A plain English description of the problem.

- Model - This model will follow our format of describing the five elements of a sequential decision problem: state variables, decision variables, exogenous information variables, the transition function and the objective function.
- Uncertainty model - Here we will provide a possible model of any uncertainties in the problem.
- Designing policies - We are going to suggest possible policies for making decisions. We have chosen our problems so that the five application settings will give us a tour through all four classes of policies. For now, we are going to let the reader try to recognize which of the four classes we are choosing.
- Extension - Finally we may suggest one or more possible extensions of our basic problem that may require changing the policy.

We then return to the four classes of policies in chapter 7 and discuss our general modeling framework, using the problems in chapters 2 - 6 to illustrate different modeling ideas.

After this discussion, we return to our pattern of teach-by-example chapters, but using more complex problems. Our remaining chapters cover the following problems:

- Chapter 8 - Energy storage I
- Chapter 9 - Energy storage II
- Chapter 10 - Supply chain management I: The two-agent newsvendor
- Chapter 11 - Supply chain management II: The beer game
- Chapter 12 - Ad-click optimization
- Chapter 13 - Blood management problem
- Chapter 14 - Optimizing clinical trials

2

An asset selling problem

One of the most basic sequential decision problems is determining when to sell an asset. We use this simple problem to illustrate our modeling style, with a simple class of policies known as *policy function approximations* (or PFAs). We begin with a simple narrative that describes the problem in plain English. We then present a mathematical model which consists of the following components:

State variables S_t - This captures all the information we need (and only the information we need) from history to model the system from time t onward. It can be helpful in many settings to divide state information into one of three categories: physical state variables R_t (inventories, location of a vehicle, attributes of a patient), other information I_t (prices, weather), and information that describes the parameters of probability distribution (we use B_t for belief state).

Decision variables x_t - This is how we control the system. The initial decision x_0 might be a design decision, while x_t for $t > 0$ represents control variables.

Exogenous information W_t - The variable W_t captures information

that first becomes available between $t-1$ and t from outside of our system.

Transition function - This is the set of equations that governs how the state variables evolve from S_t to S_{t+1}. Our canonical form is

$$S_{t+1} = S^M(S_t, x_t, W_{t+1}).$$

Objective function - We assume we wish to minimize or maximize some cost or contribution. When there are elements of the *problem* that depend on the state variables, we represent costs or contributions as $C(S_t, x_t)$ or $C(S_t, x_t, W_{t+1})$. There is an important set of *learning problems* where we are trying to learn about a function which is not itself a function of the state variable (this means the state variable consists purely of a belief state B_t).

After presenting the model, we will provide further discussion about modeling uncertainty (an often overlooked topic) and then move to the design and testing of policies.

Finally, we will follow each model with extensions to bring out different modeling issues and their effect on the solution of the model.

2.1 Narrative

We are holding a block of shares of stock, looking for an opportune time to sell. We start by assuming that we are a small player which means it does not matter how many shares we sell, so we are going to assume that we have just one share. If we sell at time t, we receive a price that varies according to some random process over time, although we do not believe prices are trending up or down. Once we sell the stock, the process stops.

2.2 Basic model

2.2.1 State variables

Our process has two state variables: the "physical state," which captures whether or not we are still holding the asset, and an "informational state" which for this problem is the price of the stock.

Our "physical state" is given by

$$R_t = \begin{cases} 1 & \text{If we are holding the stock at time } t, \\ 0 & \text{If we are no longer holding the stock at time } t. \end{cases}$$

If we sell the stock, we receive the price per share of p_t. This means our state variable is

$$S_t = (R_t, p_t).$$

2.2.2 Decision variables

The decision variable is whether to hold or sell the stock. We write this using

$$x_t = \begin{cases} 1 & \text{If we sell the stock at time } t, \\ 0 & \text{If do not sell the stock at time } t. \end{cases}$$

Of course, we can only sell the stock if we are holding the stock, so we might write a constraint as

$$x_t \leq R_t.$$

We are going to define our policy $X^\pi(S_t)$ which is going to define how we make decisions. For example, a simple policy might be to sell if the price drops below some limit point that we think suggests that it is starting a big decline. Thus, we could write this policy as

$$X^{sell-low}(S_t|\theta^{low}) = \begin{cases} 1 & \text{If } p_t < \theta^{low} \text{ and } R_t = 1 \\ 1 & \text{If } t = T \text{ and } R_t = 1 \\ 0 & \text{Otherwise} \end{cases} \tag{2.1}$$

We would of course need to tune θ^{low} to get the best performance within this class of policy. A separate question is whether this is a good structure of a policy, since it is easy to design others.

2.2.3 Exogenous information

The only random process in our basic model is the change in price. There are two ways to write this. One is to assume that the exogenous information is the change in price. We can write this as

$$\hat{p}_{t+1} = p_{t+1} - p_t.$$

This means that our price process is evolving according to

$$p_{t+1} = p_t + \hat{p}_{t+1}.$$

We would then write our exogenous information W_{t+1} as

$$W_{t+1} = (\hat{p}_{t+1}).$$

An alternative way that some will find more natural is to simply let W_t be the new price, in which we would write

$$W_{t+1} = (p_{t+1}).$$

Note that with this style, p_{t+1} is the exogenous information, but we are not writing it with a hat. The value of the hat notation is that it tells us that a variable is determined exogenously, rather than being a variable that depends on other random variables. We could let $\hat{p}_{t+1}$ be the price that arrives exogenously at time $t+1$, but this then requires us to either write $p_{t+1} = \hat{p}_{t+1}$, or just use $\hat{p}_{t+1}$ as the price (rather than the more compact p_{t+1}). Both styles are correct and just depend on the preference of the modeler.

2.2.4 Transition function

The transition function is the equations that describe how the state evolves. The transition equation for R_t is given by

$$R_{t+1} = R_t - x_t, \tag{2.2}$$

where we have the constraint that $x_t \leq R_t$ to ensure that we do not sell the asset when we no longer own it.

Next we have to write how the price process evolves over time. If we use the $\hat{p}_t$ notation, the transition function for the price p_t would be given by

$$p_{t+1} = p_t + \hat{p}_{t+1}. \tag{2.3}$$

Equations (2.2) and (2.3) make up what we call our *transition function* that we write as

$$S_{t+1} = S^M(S_t, X^\pi(S_t), W_{t+1}). \tag{2.4}$$

If we use our policy $X^\pi(S_t)$ to make decisions, and if we choose sample path ω that determines the sequence $W_1, W_2, \ldots, W_T$, then we can write a simulation of our process as

$$(S_0, x_0 = X^\pi(S_0), W_1(\omega), S_1, x_1 = X^\pi(S_1), W_2(\omega), \ldots, x_{T-1}, W_T(\omega), S_T).$$

Note that as we write the sequence, we index variables by their information content. For example, S_0 is an initial state, and x_0 depends only on S_0. By contrast, any variable indexed by t is allowed to "see" any of the outcomes of our exogenous process $W_1, \ldots, W_t$, but are not allowed to see W_{t+1}.

2.2.5 Objective function

We close with the problem of determining the best policy. To start, we have to have some performance metric, which for this problem would be how much we earn from selling our stock. We can define a generic contribution function that we write as $C(S_t, x_t)$, which would be given by

$$C(S_t, x_t) = p_t x_t.$$

In our problem, $x_t = 0$ until we sell our asset, at which point we would have $x_t = 1$, which is going to happen just once over our horizon. We write the dependence of $C(S_t, x_t)$ to capture the dependence on the state, which is because of the presence of the price p_t.

We now want to formulate our optimization problem. If the prices were given to us in advance, we would write

$$\max_{x_0,\ldots,x_{T-1}} \sum_{t=0}^{T-1} p_t x_t, \tag{2.5}$$

where we would impose the constraints

$$\sum_{t=0}^{T-1} x_t = 1, \tag{2.6}$$

$$x_t \leq 1, \tag{2.7}$$

$$x_t \geq 0. \tag{2.8}$$

This is fine when the problem is deterministic, but how do we model the problem to handle the uncertainty in the prices? What we do is to imagine that we are simulating a policy following a sample path ω of prices $p_1(\omega), p_2(\omega), \ldots$. Using a policy π, we would then generate a series of states using

$$S_{t+1}(\omega) = S^M(S_t(\omega), X^\pi(S_t(\omega)), W_{t+1}(\omega)).$$

We write $S_t(\omega)$ to express the dependence on the sample path. We could also have written $S_t^\pi(\omega)$ to express the dependence on the policy π, but we tend to suppress this for simplicity.

If we follow policy π along this sample path, we can compute the performance using

$$\hat{F}^\pi(\omega) = \sum_{t=0}^{T-1} p_t(\omega) X^\pi(S_t(\omega)).$$

This is for one sample path. We can simulate over a sample of N samples $\omega^1, \ldots, \omega^n, \ldots, \omega^N$) and take an average using

$$\overline{F}^\pi = \frac{1}{N} \sum_{n=1}^{N} \hat{F}^\pi(\omega^n). \tag{2.9}$$

Finally, we write out optimization problem in terms of finding the best policy, which we can write

$$\max_{\pi \in \Pi} \overline{F}^\pi. \tag{2.10}$$

In practice we are typically using averages such as in equation (2.9) for our optimization problem. However, this is just an approximation of taking an actual expectation, which we will write as

$$F^\pi = \mathbb{E}\hat{F}^\pi. \tag{2.11}$$

By convention, when we write the expectation, we drop the indexing on ω and instead view $\hat{F}^\pi$ as a random variable, whereas $\hat{F}^\pi(\omega)$ is treated as a sample realization (this is standard convention, so just get used to it). Using our expectation operator, we would write our objective function as

$$\max_{\pi \in \Pi} F^\pi. \tag{2.12}$$

The form in equation (2.12) (or (2.11)) is nice and compact. You just have to remember that it is rarely the case that we can actually compute the expectation, so we generally depend on simulation.

Now we are left with the problem of searching over policies. We are going to return to this below, but for the moment, we can think of this search in two ways. First, we have to think of different types of functions for making decisions. One example of a function is given by the policy in 2.1, but it is not hard to think of others (for example, selling if the price drops by too much, or goes over some upper limit). Then, we have to recognize that functions such as (2.1) may depend on parameters such as θ^{sell} that have to be tuned. To evaluate the policy in (2.1) would require that we first find the best value of θ^{sell}.

We return to this question in section 2.4.

In the models in the remaining chapters in this book, we are going to use the style in (2.12) as our default way of writing an objective, where we will defer until later the design of the policies, and the method for evaluating policies (since we can never compute the expectation).

2.3 Modeling uncertainty

We are going to need some way to sample observations of W_t. One way is to draw samples from history. Imagine that we are interested in running our simulation over a period of a year. We could think of getting 10 years of past data, and scaling it so the processes all start at the same point. This is a popular and widely used method for creating samples, since it avoids making any simplifying modeling assumptions. The drawback is that this only gives us 10 sample paths, which may represent patterns from history that are no longer relevant.

The second strategy, which we will often use, is to estimate a statistical model. For our basic model, we might assume

$$p_{t+1} = p_t + \hat{p}_{t+1}, \tag{2.13}$$

where $\hat{p}_{t+1}$ is normally distributed with mean 0 and some variance σ^2. In addition to assuming that $\hat{p}_{t+1}$ is normally distributed, we are also going to assume that the sequence $\hat{p}_1, \ldots, \hat{p}_t, \ldots$ are also independent.

To simulate our process, we need to assume a probabilistic model for $\hat{p}_{t+1}$. A simple model would be to assume that $\hat{p}_{t+1}$ is normally distributed with mean 0 and variance σ^2. Later we are going to have to simulate observations of W_{t+1} (or $\hat{p}_{t+1}$). There are a number of ways to do this, but most computer languages have provided functions for doing this. For example, Excel provides the function $\texttt{Norm.inv}(p, \mu, \sigma)$, which returns the value w of a random variable W with mean μ and standard deviation σ where $P[W \leq w] = p$. If we set $\mu = 0$ and $\sigma = 1$, then if $p = .5$ we would obtain $Norm.inv(.5, 0, 1) = 0$. Similarly, if $p = 0.975$, we would obtain $\texttt{Norm.inv}(.975, 0, 1) = 1.96$.

The real power of functions such as $\texttt{Norm.inv}(p, \mu, \sigma)$ is that it allows us to generate samples of our random variable W. To do this, we use the presence of a function (again, available in all computer packages) to generate a random variable that we call U that is uniformally distributed between 0 and 1. In Excel, this function is called `Rand()`. We can then generate random observations of our normally distributed random variable W using

$$W = Norm.inv(U, \mu, \sigma).$$

Table 2.1 illustrates ten observations of random variables U that are uniformly distributed between 0 and 1, and the corresponding samples of normally distributed random variables with mean 0 and variance 1.

Equation (2.13) is a pretty basic price model, but it will help illustrate our modeling framework. Below, we are going to introduce some extensions which include a richer model.

2.4 Designing policies

We have already illustrated one policy in equation (2.1) which we called our "sell-low" policy, which is parameterized by a selling point that we called $\theta^{sell-low}$. Clearly we can think of other policies. One might be a "high-low" selling policy, where we want to sell if the price jumps too high or too low. Let $\theta^{high-low} = (\theta^{low}, \theta^{high})$. This might be written

$$X^{high-low}(S_t|\theta^{high-low}) = \begin{cases} 1 & \text{If } p_t < \theta^{low} \text{ or } p_t > \theta^{high} \\ 1 & \text{If } t = T \text{ and } R_t = 1 \\ 0 & \text{Otherwise} \end{cases} \quad (2.14)$$

U	W
0.8287	0.9491
0.6257	0.3206
0.9343	1.5086
0.4879	-0.0303
0.3736	-0.3223
0.8145	0.8947
0.0385	-1.7685
0.0089	-2.3698
0.9430	1.5808
0.3693	-0.3336

Table 2.1: Ten uniform random variables U, and ten corresponding samples of normally distributed random variables with mean 0 and variance 1.

A possible objection to this policy could be that it prematurely sells a rising stock. Perhaps we just want to sell when the stock rises above a tracking signal. First create a smoothed estimate of the price using

$$\bar{p}_t = (1-\alpha)\bar{p}_{t-1} + \alpha\hat{p}_t. \tag{2.15}$$

Now consider a tracking policy that we might write as

$$X^{track}(S_t|\theta^{track}) = \begin{cases} 1 & \text{If } p_t \geq \bar{p}_t + \theta^{track} \\ 1 & \text{If } t = T \text{ and } R_t = 1 \\ 0 & \text{Otherwise} \end{cases} \tag{2.16}$$

For this policy, we are going to need to tweak our model, because we now need $\bar{p}_t$ in order to make a decision. This means we would now write our state as

$$S_t = (R_t, p_t, \bar{p}_t).$$

We can write our classes of policies as the set $\mathcal{F} = \{\text{“}sell - low\text{”}, \text{“}high - low\text{”}, \text{“}track\text{”}\}$. For each of these classes, we have a set of parameters that we can write as θ^f for $f \in \mathcal{F}$. For the “sell-low” and “track” there is a single parameter, while $\theta^{high-low}$ has two parameters.

Now we can write our search over policies $\pi \in \Pi$ in a more practical way as searching over function classes $f \in \mathcal{F}$, and then searching over

	$t=1$	$t=2$	$t=3$	$t=4$	$t=5$	$t=6$	$t=7$	$t=8$
ω^n	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8
ω^1	42.67	45.53	47.07	47.56	47.80	48.43	46.93	46.57
ω^2	46.35	43.15	42.51	40.51	41.50	41.00	39.16	41.11
ω^3	43.17	45.16	45.37	44.30	45.35	47.23	47.35	46.30
ω^4	45.24	45.67	46.18	46.22	45.69	44.24	43.77	43.57
ω^5	47.68	46.32	46.14	41.53	44.84	45.17	44.92	46.09
ω^6	47.83	44.70	43.05	43.77	42.61	44.32	44.16	45.29
ω^7	45.11	43.67	43.14	44.78	43.12	42.36	41.60	40.83
ω^8	46.78	44.98	44.53	45.42	46.43	47.67	47.68	49.03
ω^9	43.16	44.57	45.99	47.38	45.51	46.27	46.02	45.09
ω^{10}	46.57	45.01	46.73	46.08	47.40	49.14	49.03	48.74

Table 2.2: Illustration of a set of price paths.

parameters $\theta^f \in \Theta^f$, where Θ^f tells us the range of possible values (capturing at the same time the dimensionality of θ^f).

2.5 Policy evaluation

We indicated above that we can evaluate a policy by simulating it using

$$\hat{F}^{\pi}(\omega) = \sum_{t=0}^{T-1} p_t(\omega) X^{\pi}(S_t(\omega)).$$

where ω is used to represent a sample path of realizations of whatever exogenous random variables are used in the model. Table 2.2 illustrates a series of sample paths of prices. For example, imagine that we are using the "sell-low" policy with $\theta^{sell-low} = \$42$. Now consider testing it on sample path ω^5. The result would be

$$\hat{F}^{sell-low}(\omega^5) = \$41.53,$$

since \$41.53 is the first price that falls below \$42. If none of the prices fall below our sell point, then all of our policies are designed to sell at the end.

We can then evaluate each policy (both the class of policy, and the parameters for that class) by doing repeated simulations and taking an average. We write this as

$$\overline{F}^{\pi} = \frac{1}{N}\sum_{n=1}^{N} \hat{F}^{\pi}(\omega^n).$$

Sometimes we need to express a confidence interval, since $\overline{F}^{\pi}$ is nothing more than a statistical estimate. We would first compute an estimate of the variance of our random variable $\hat{F}^{\pi}$, which we do using

$$(\hat{\sigma}^{\pi})^2 = \frac{1}{N-1}\sum_{n=1}^{N}(\hat{F}^{\pi}(\omega^n) - \overline{F}^{\pi})^2.$$

We then get our estimate of the variance of our average $\overline{F}^{\pi}$ using

$$(\bar{\sigma}^{\pi})^2 = \frac{1}{N}(\hat{\sigma}^{\pi})^2.$$

From this, we can construct a confidence interval to compare two policies which we might call π^A and π^B. Let μ^{π} be the true performance of policy π, where $\overline{F}^{\pi}$ is our statistical estimate of μ^{π}. We would like to get a confidence interval for the difference $\mu^{\pi^A} - \mu^{\pi^B}$. Our best estimate of this difference is $(\overline{F}^{\pi^A} - \overline{F}^{\pi^B})$. The variance of this difference is

$$Var(\overline{F}^{\pi^A} - \overline{F}^{\pi^B}) = (\bar{\sigma}^{\pi^A})^2 + (\bar{\sigma}^{\pi^B})^2),$$

where we assume that the estimates $\overline{F}^{\pi^A}$ and $\overline{F}^{\pi^B}$ are independent, which means that each policy is being tested on a different random sample of prices. When this is the case, we would compute our confidence interval using

$$\mu^{\pi^A} - \mu^{\pi^B} \in \left(\overline{F}^{\pi^A} - \overline{F}^{\pi^B} + z_{\alpha}\sqrt{(\bar{\sigma}^{\pi^A})^2 + (\bar{\sigma}^{\pi^B})^2}\right).$$

A better approach is to use the same samples to evaluate each policy. For example, we could test each policy on the same sample path ω chosen from table 2.2). Testing our policies this way, we would get $\hat{F}^{\pi^A}(\omega)$ and $\hat{F}^{\pi^B}(\omega)$ (using the same set of prices $p_t(\omega)$), and then compute the difference

$$\delta\hat{F}^{A-B}(\omega) = \hat{F}^{\pi^A}(\omega) - \hat{F}^{\pi^B}(\omega).$$

Now we compute the average difference

$$\delta\overline{F}^{A-B} = \frac{1}{N}\sum_{n=1}^{N}\hat{F}^{A-B}(\omega^n),$$

and the variance

$$(\delta\bar{\sigma}^{A-B})^2 = \frac{1}{N}\left(\frac{1}{N-1}\sum_{n=1}^{N}(\delta\hat{F}^{A-B}(\omega^n) - \delta\overline{F}^{A-B})^2\right).$$

Note that the variance $\delta\bar{\sigma}^{A-B,2}$ will be smaller than the variance when independent samples are used. The confidence interval for the difference would then be

$$\delta\mu^{A-B} \in (\delta\overline{F}^{A-B} - z_\alpha\sqrt{\delta\bar{\sigma}^{A-B,2}}, \delta\overline{F}^{A-B} + z_\alpha\sqrt{\delta\bar{\sigma}^{A-B,2}}).$$

Computing confidence intervals may be useful when comparing different classes of policies. We would not use confidence intervals to decide the best parameter θ that governs a policy.

2.6 Extensions

2.6.1 Time series price processes

Imagine that we want a somewhat more realistic price process that captures autocorrelation over time. We might propose that

$$p_{t+1} = \theta_0 p_t + \theta_1 p_{t-1} + \theta_2 p_{t-2} + \varepsilon_{t+1}, \tag{2.17}$$

where we still assume that the random noise ε_t is independent (and identically distributed) over time. We are also going to assume for the moment that we know the coefficients $\theta = (\theta_0, \theta_1, \theta_2)$.

This change requires a subtle change in our model, specifically the state variable. We are going to replace our old transition equation for prices, (2.3), with our new time series model given in (2.17). To compute p_{t+1}, it is no longer enough to know p_t, we now also need to know p_{t-1} and p_{t-2}. Our state variable would now be given by

$$S_t = (R_t, p_t, p_{t-1}, p_{t-2}).$$

For the policies we have considered above, this does not complicate our model very much. Later, we are going to introduce policies where the additional variables represent a major complication.

2.6.2 Basket of assets

Another twist arises when we are considering a basket of assets. Let p_{ti} be the price of asset i. Assume for the moment that each price evolves according to the basic process

$$p_{t+1,i} = p_{ti} + \varepsilon_{t+1,i}.$$

We could assume that the noise terms $\varepsilon_{t+1,i}$ are independent across the assets $i \in \mathcal{I}$, but a more realistic model would be to assume that the prices of different assets are correlated. Let

$$\sigma_{ij} = Cov_t(p_{t+1,i}, p_{t+1,j})$$

be the covariance of the random prices $p_{t+l,i}$ and $p_{t+1,j}$ for assets i and j given what we know at time t. Assume for the moment that we know the covariance matrix Σ, perhaps by using a historical datset to estimate it (but holding it fixed once it is estimated).

We can use the covariance matrix to generate sample realizations of correlated prices using a technique called Cholesky decomposition. It proceeds by creating what we call the "square root" of the covariance matrix Σ which we store in a lower triangular matrix L. In python, using the NumPy package, we would use the python command

$$L = \texttt{scipy.linalg.cholesky}(\Sigma^X, lower = True)$$

The matrix L allows us to obtain the matrix Σ using

$$\Sigma = L^T L.$$

Now let Z be a vector of random variables, one for each asset, where $Z_i \sim N(0,1)$ (virtually every computer language has routines for creating random samples from normal distributions with mean 0, variance 1). Let p_t, p_{t+1} and Z be column vectors (dimensioned by the number of assets). We first create a sample $\hat{Z}$ by sampling from $N(0,1)$ $|\mathcal{I}|$ times. Our sample of prices p_{t+1} is then given by

$$p_{t+1} = p_t + L\hat{Z}.$$

For this problem, our state variable is given by

$$S_t = (R_t, p_t)$$

where $R_t = (R_{ti})_{i \in \mathcal{I}}$ captures how many shares of each asset we own, while p_t is our current vector of prices. The covariance matrix Σ is not in the state variable because we assume it is static (which means we put it in S_0). The answer changes if we were to update the covariance matrix with each new observation, in which case we would write the covariance matrix as Σ_t to capture its dependence on time. Since it now varies dynamically, the state variable would be

$$S_t = (R_t, p_t, \Sigma_t).$$

3

Adaptive market planning

3.1 Narrative

There is a broad class of problems that involve allocating some resource to meet an uncertain (and sometimes unobservable) demand. Examples include

- Stocking a perishable inventory (e.g. fresh fish) to meet a demand where leftover inventory cannot be held for the future.
- Stocking parts for high-technology manufacturing (e.g. jet engines) where we need to order parts to meet known demand, but where the parts may not meet required specifications and have to be discarded. So, we may need to order eight parts to meet the demand of five because several of the parts may not meed required engineering specifications.
- We have to allocate time to complete a task (such as driving to work, or allocating time to complete a project).
- We have to allocate annual budgets for activities such as marketing. Left-over funds are returned to the company.

The simplest problem involves making these decisions to meet an uncertain demand with a known distribution, but the most common applications involve distributions that are unknown and need to be learned. There may be other information such as the availability of forecasts of the demand, as well as dynamic information such as the market price for the fresh fish (which may be known or unknown before the resource decision is made).

This problem has been widely studied since the 1950's, originally known as the "single period inventory problem" but currently is primarily identified as the "newsvendor problem." It is widely used as the canonical problem in optimization under uncertainty.

The newsvendor problem is typically formulated in the form

$$\max_x \mathbb{E}F(x,W) = \mathbb{E}\{p\min\{x,W\} - cx\}, \tag{3.1}$$

where x is our decision variable that determines the amount of resource to meet the task, and where W is the uncertain demand for the resource. We assume that we "purchase" our resource at a unit cost of c, and we sell the smaller of x and W at a price p (which we assume is greater than c). The objective function given in equation (3.1) is called the *asymptotic* form of the newsvendor problem.

If W was deterministic (and if $p > c$), then the solution is easily verified to be $x = W$. Now imagine that W is a random variable with probability distribution $f^W(w)$ (W may be discrete or continuous). Let $F^W(w) = Prob[W \leq w]$ be the cumulative distribution function. If W is continuous, and if we could compute $F(x) = \mathbb{E}F(x,W)$, then the optimal solution x^* would satisfy

$$\left.\frac{dF(x)}{dx}\right|_{x=x^*} = 0.$$

Now consider the stochastic gradient, where we take the derivative of $F(x,W)$ assuming we know W, which is given by

$$\frac{dF(x,W)}{dx} = \begin{cases} p-c & x \leq W \\ -c & x > W \end{cases} \tag{3.2}$$

Taking expectations of both sides give

$$\begin{aligned}\mathbb{E}\frac{dF(x,W)}{dx} &= (p-c)Prob[x \leq W] - cProb[x > W] \\ &= (p-c)(1-F^W(x)) - cF^W(x) \\ &= (p-c) - pF^W \\ &= 0 \quad \text{For } x = x^*.\end{aligned}$$

We can now solve for $F^W(x^*)$ giving

$$F^W(x^*) = \frac{p-c}{p}.$$

Thus, as c decreases to 0, we want to order an amount x^* that will satisfy demand with probability 1. As c approaches p, then the optimal order quantity will satisfy demand with a probability approaching 0.

This means that we compute $(p-c)/p$, which is a number between 0 and 1, and then find the quantity x^* that corresponds to the order quantity where the probability that the random demand is less than x^* is equal to $(p-c)/p$.

We have just seen two situations where we can find the order quantity exactly: when we know W in advance (we might call this the perfect forecast) or when we know the distribution of W. This result has been known since the 1950's, sparking a number of papers for estimating the distribution of W from observed data, handling the situation where we cannot observe W directly when $x < W$ (that is, we only observe sales, rather than demand, a situation known as "censored demands").

There is a wide range of problems that we can solve when there is no uncertainty, but this is a rare special case where we can find an optimal solution when the distribution is known. For example, we can create a multidimensional newsvendor problem with different types of products and where a customer might shift to another product if the preferred product runs out.

In this chapter, we are going to start by addressing the single-dimensional newsvendor problem, where we assume that the demand distribution W is unknown, but can be observed. This work can be applied to the case where x and W are vectors (for the multiproduct problem), but where we do know the distribution of W.

3.2 Basic model

We are going to solve our problem using a classical method, first developed in 1951 by Robbins and Monro, called a stochastic gradient algorithm. We introduced the stochastic gradient in equation (3.2) above. Here, we are going to use the more general notation $\nabla_x F(x, W)$ to be the derivative (or gradient, if x is a vector) of $F(x, W)$ after we have observed W. In an actual algorithm, we are going to pick x^n, then observe W^{n+1}, and then compute $\nabla_x F(x^n, W^{n+1})$. Our algorithm is then given by

$$x^{n+1} = x^n + \alpha_n \nabla_x F(x^n, W^{n+1}), \tag{3.3}$$

where α_n is known as a *stepsize*. This algorithm has been widely studied for its asymptotic behavior, where it has been shown that we obtain asymptotic optimality

$$\lim_{n\to\infty} x^n = x^*,$$

if the stepsize α_n satisfies

$$\alpha_n > 0, \tag{3.4}$$

$$\sum_{n=1}^{\infty} \alpha_n = \infty, \tag{3.5}$$

$$\sum_{n=1}^{\infty} (\alpha_n)^2 < \infty. \tag{3.6}$$

Equation (3.5) ensures that the stepsizes do not shrink so quickly that we stall out on the way to the optimum, while (3.6) has the effect of insuring that the variance of our estimate of x^* does shrink to zero.

The stochastic gradient algorithm in (3.3) is elegant and simple, primarily because stochastic gradients are often easy to compute. We can handle constraints $x \in \mathcal{X}$ (such as the requirement that $x \geq 0$) by introducing a projection operation, which is typically written

$$x^{n+1} = \Pi_{\mathcal{X}}[x^n + \alpha_n \nabla_x F(x^n, W^{n+1})]. \tag{3.7}$$

For example, if we have the constraint $x \geq 0$, the projection operator would be just

$$\Pi_{\mathcal{X}}[x^n + \alpha_n \nabla_x F(x^n, W^{n+1})] = \max\{0, x^n + \alpha_n \nabla_x F(x^n, W^{n+1})\}.$$

The challenge with stochastic gradient algorithms has been the selection of stepsizes. For example, a stepsize that satisfies (3.4) - (3.6) is $\alpha_n = 1/n$. The problem with this rule is that it often does not work well in practice since it may shrink to zero too quickly, producing slow convergence. Complicating the choice of stepsize is that there are special situations where $\alpha_n = 1/n$ may be the best stepsize formula.

Methods for choosing stepsizes are typically known as stepsize rules, but in our framework we are going to view these as policies, and we will return to them later.

Our stochastic gradient algorithm is a form of sequential decision problem. While not widely recognized as such in the literature, we can model the algorithm using our standard framework. We are going to model the problem where we use Kesten's stepsize rule for the stepsize. We introduce this perspective early in our modeling framework because stochastic gradient algorithms will represent one of our most powerful solution procedures in sequential decisions.

This is one of those problems where it is useful to skip past the state variable initially, and return to it after looking at the rest of the model.

3.2.1 State variables

The state variable captures the information we have at time n that we need, along with the policy and the exogenous information, to compute the state at time $n+1$. For our stochastic gradient algorithm, our state variable is given by

$$S^n = (x^n).$$

3.2.2 Decision variables

The trick with this problem is recognizing the decision variable. It is tempting to think that x^n is the decision, but in the context of this algorithm, the real decision is the stepsize α_n (note: while we generally index functions and variables using counter n in the superscript, we make an exception with stepsizes, since α^n is easily confused with raising α to the power of n). As with all of our sequential decision problems, the decision (that is, the stepsize) is determined by what is typically

referred to as a stepsize rule, but is sometimes called a stepsize policy that we denote by $\alpha^\pi(S^n)$.

Normally we introduce policies later, but to help with the understanding the model, we are going to start with a basic stepsize policy called a *harmonic stepsize rule* given by

$$\alpha^{harmonic}(S^n|\theta^{step}) = \frac{\theta^{step}}{\theta^{step}+n-1}.$$

This is a simple deterministic stepsize rule, which means that we know in advance the stepsize α_n once we know n. Below we introduce a more interesting stochastic stepsize policy that requires a richer state variable.

We are also going to let $X^\pi(S^n)$ be the value of x^n determined by stepsize policy $\alpha^\pi(S^n)$.

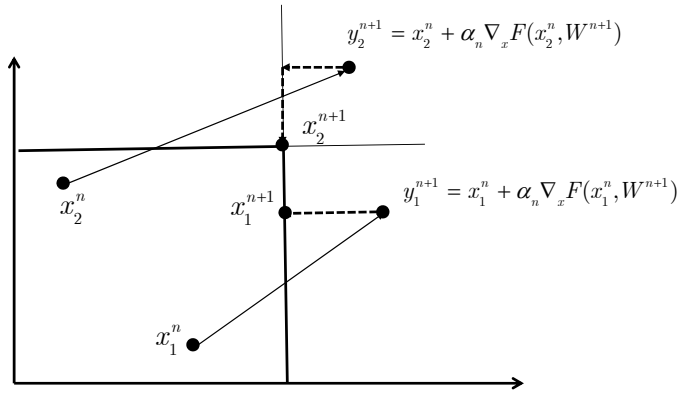


Figure 3.1: Illustration of projection mapping of two points, y_1 and y_2, back onto box constraints.

3.2.3 Exogenous information

The exogenous information is the random demand W^{n+1} for the resource (product, time or money) that we are trying to meet with our supply of product x^n. We may assume that we observe W^{n+1} directly, or we may just observe whether $x^n \leq W^{n+1}$, or $x^n > W^{n+1}$.

3.2.4 Transition function

The transition equation, for the setting where x is unconstrained, is given by

$$x^{n+1} = x^n + \alpha_n \nabla_x F(x^n, W^{n+1}).. \tag{3.8}$$

If we require that x^n be in a set $\mathcal{X}$, we write the transition equation as

$$x^{n+1} = \Pi_{\mathcal{X}}[x^n + \alpha_n \nabla_x F(x^n, W^{n+1})]. \tag{3.9}$$

where $\Pi_{\mathcal{X}}[x]$ is the projection operator that maps x into the point in $\mathcal{X}$ that is closest to x.

Figure 3.1 illustrates what happens when we step from points x_1 and x_2 to points y_1 and y_2 (respectively) to points outside of a set of box constraints of the form $x \leq u$. The point y_1 just violates one upper bound constraint, so we map back back to that constraint. The point

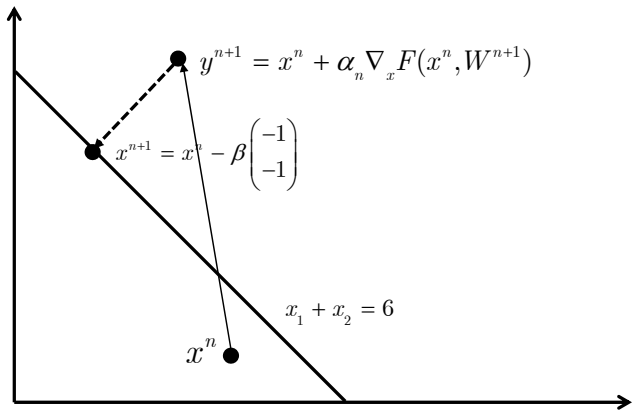


Figure 3.2: Illustration of projection mapping of the point y back onto a budget constraint $x_1 + x_2 \leq 6$.

y_2 violates the upper bound constraint in both dimensions. We fix this by mapping each dimension of y in order, giving us the corner point.

Figure 3.2 illustrates the process of mapping back onto a budget constraint of the form $x_1 + x_2 \leq B$. When we first step to a point y^{n+1} where $y_1^{n+1} + y_2^{n+1} > B$, we perform the projection by subtracting a constraint β from each dimension. We then find β that brings us back to the budget. This is very easy to do. If we have two dimensions, we first compute the gap $y_1^{n+1} + y_2^{n+1} - B$, then divide by the number of dimensions (two in this example). We then subtract this number from each dimension of y to get a vector whose sum equals B.

The process of subtracting β from each dimension might result in one or more dimensions of the updated y becoming negative. If this happens, just set each of the elements of the updated y to zero, which then gives us another update that is now larger than B. Lock the dimensions of the updated y that are now zero so they are no longer updated, and repeat the process.

3.2.5 Objective function

In our market setting, at each iteration we receive a net benefit given by

$$F(x^n, W^{n+1}) \quad = \quad p \min\{x^n, W^{n+1}\} - cx^n.$$

Since we are experiencing our decisions in a market setting, we want to maximize the total reward over some horizon. This means we need to find the best policy (which in this setting means the best stepsize rule) by solving

$$\max_{\pi} \mathbb{E}\left\{\sum_{n=0}^{N-1} F(X^{\pi}(S^n), W^{n+1}) | S^0\right\}. \tag{3.10}$$

where $S^{n+1} = S^M(S^n, X^{\pi}(S^n), W^{n+1})$ describes the evolution of the algorithm. Here, π refers to the type of stepsize rule (e.g harmonic vs. Kesten), and any tunable parameters (such as θ^{step}).

We note in passing that the newsvendor problem has been studied since the 1950's using the asymptotic form given in equation (3.1), which we restate here as

$$\max_{x} \mathbb{E}F(x, W) = \mathbb{E}\{p \min\{x, W\} - cx\}.$$

Here, we are trying to find the x that maximize profits on average, but this ignores the reality that we are learning this as we go.

3.3 Uncertainty modeling

Let $f^W(w)$ be the distribution of W (this might be discrete or continuous), with cumulative distribution function $F^W(w) = Prob[W \leq w]$. We might assume that the distribution is known with an unknown parameter. For example, imagine that W follows a Poisson distribution with mean μ given by

$$f^W(w) = \frac{\mu^w e^{-\mu}}{w!}, \quad w = 0, 1, 2, \ldots.$$

We may assume that we know μ, in which case we could solve this problem using the analytical solution given at the beginning of the chapter. Assume, instead, that μ is unknown, but with a known distribution

$$p_k^\mu = Prob[\mu = \mu_k],$$

Note that this distribution $p^\mu = (p_k^\mu)_{k=1}^K$ would be modeled in our initial state S^0.

3.4 Designing policies

We have already introduced two choices of stepsize policies which we write as $\alpha^\pi(S^n)$ to mimic our style elsewhere for writing policies.

A wide range of stepsize policies (often called stepsize rules) have been suggested in the literature. One of the simplest and most popular is the harmonic stepsize policy given by

$$\alpha^{harmonic}(S^n|\theta^{step}) = \frac{\theta^{step}}{\theta^{step} + n - 1}.$$

Figure 3.3 illustrates the behavior of the harmonic stepsize rule for different values of θ^{step}.

The harmonic stepsize policy is also known as a deterministic policy, because we know its value for a given n. The challenge with deterministic policies is that they are not allowed to adapt to the data. For this reason, it is often useful to use a stochastic rule. One of the earliest and simplest examples is Kesten's rule

$$\alpha^{kesten}(S^n|\theta^{step}) = \frac{\theta^{step}}{\theta^{step} + K^n - 1},$$

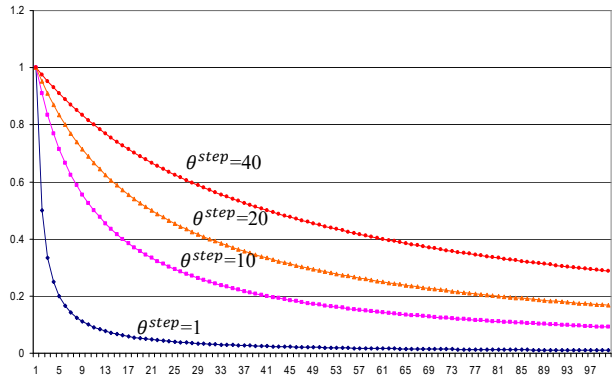


Figure 3.3: Harmonic stepsizes for different values of θ^{step}.

where K^n is a counter that counts how many times the gradient has switched direction. We determine this by asking if the product (or inner product, if x is a vector) $(\nabla^x F(x^n, W^{n+1}))^T \nabla^x F(x^{n-1}, W^n) < 0$. If the gradient is switching directions, then it means that we are in the vicinity of the optimum and are stepping past it, so we need to reduce the stepsize. This formula is written

$$K^n = \begin{cases} K^n + 1 & \text{If } (\nabla^x F(x^n, W^{n+1}))^T \nabla^x F(x^{n-1}, W^n) < 0, \\ K^n & \text{Otherwise} \end{cases} \tag{3.11}$$

Now we have a stepsize that depends on a random variable K^n, which is why we call it a stochastic stepsize rule. If we use Kesten's rule, we have to modify our state variable to include K^n, giving us

$$S^n = (x^n, K^n).$$

We also have to add equation (3.11) to our transition function.

Another stepsize rule, known as AdaGrad, is particularly well suited when x is a vector with element $x_i,\ i = 1, \ldots, I$. To simplify the notation a bit, let the stochastic gradient with respect to element x_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

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

We then set a stepsize for the ith dimension using

$$\alpha_{ni} = \frac{\eta}{(G^n_{ii})^2 + \epsilon}, \tag{3.12}$$

where ϵ is a small number (e.g. 10^{-8} to avoid the possibility of dividing by zero).

3.5 Policy evaluation

We illustrate policy iteration for both forms of the objective function given above: cumulative reward, where we add up rewards incurred as we are trying each answer x^n, and final reward, where we only care about the performance of the final design, $x^{\pi,N}$.

3.5.1 Cumulative reward

For a given stepsize policy $\alpha^\pi(S^n)$, we can simulate the value of the policy. If we assume that the true mean μ is known, we just simulate a sample path $\omega = (W^1(\omega), W^2(\omega), \ldots, W^N(\omega))$. If we assume that μ is unknown but where $Prob[\mu = \mu^k] = p^\mu_k$, then we would randomly sample a truth $\mu(\omega)$ from the distribution p^μ. So, when we refer to a sample path ω, we mean a sample of anything random, whether it is μ or the realizations $W^n(\omega)$.

The performance of the policy from a single simulation is given by

$$F^\pi(\omega) = \sum_{n=1}^{N} F(X^\pi(S^n(\omega)), W^{n+1}).$$

We then will run K of these simulations, where ω^k is the kth sample path, and take an average, giving us

$$\overline{F}^\pi = \frac{1}{K} F^\pi(\omega^k).$$

3.5.2 Final reward

We might pose the problem that we are able to simulate this problem in the computer using observations of W drawn from some distribution which we would have to assume is known (in order to draw sample

realizations). Even in this setting, we cannot run an infinite number of iterations. Assume we have a budget of N experiments, and let $x^{\pi,N}$ be the solution that our algorithm produces using stepsize rule π. The solution $x^{\pi,N}$ is a random variable, because it depends on $W^1, \ldots, W^N$. Once we obtain $x^{\pi,N}$, we need to test it, and we let $\widehat{W}$ be the random variable we use for testing. The problem of finding the best policy (the best stepsize rule) would then be stated

$$\max_{\pi} \mathbb{E}\{F(x, W)|S_0\} = \mathbb{E}_{S_0}\mathbb{E}_{W^1,\ldots,W^N|S_0}\mathbb{E}_{\widehat{W}|S_0}F(x^{\pi,N}, \widehat{W}). \quad (3.13)$$

Equation (3.13) starts with an expectation over the initial state S_0 only because we might have a distribution of belief about something. For example, we might assume that the demand W is given by

$$W_t = \mu + \varepsilon_t$$

where ε_t is a random variable with some distribution. But what if we do not know μ? We might assume that μ is some fixed number that is somewhere between 10 and 20. We might assume that μ takes on one of a set of values in $(\mu_1, \ldots, \mu_K)$, each with probability $p^\mu = (p^\mu_k)_{k=1}^K$.

$$\begin{aligned}
\mathbb{E}\{F(x, W)|S_0\} &= \mathbb{E}_{S_0}\mathbb{E}_{W^1,\ldots,W^N|S^0}\mathbb{E}_{\widehat{W}|S^0}F(x^{\pi,N}, \widehat{W}) \\
\mathbb{E}\{F(x, W)|S_0\} &= \mathbb{E}_{\mu}\mathbb{E}_{W^1,\ldots,W^N|\mu}\mathbb{E}_{\widehat{W}|\mu}F(x^{\pi,N}, \widehat{W}) \\
&= \sum_{k=1}^{K} p^\mu_k \mathbb{E}_{W^1,\ldots,W^N|\mu=\mu_k}\mathbb{E}_{\widehat{W}|\mu=\mu_k}F(x^{\pi,N}, \widehat{W})
\end{aligned}$$

Figure 3.4 illustrates different rates of convergence for different stepsize rules, showing $F(x^n, W^{n+1})$ as a function of the number of iterations. If we were optimizing the final reward in (3.13), we could just pick the line that is highest, which depends on the budget N. If we are optimizing the cumulative reward in equation (3.10), then we have to focus on the area under the curve, which favors rapid initial convergence.

3.6 Extensions

1) Imagine that we do not know μ, but let's assume that μ can take on one of the values $(\mu_1, \mu_2, \ldots, \mu_K)$. Let H^n be the history of

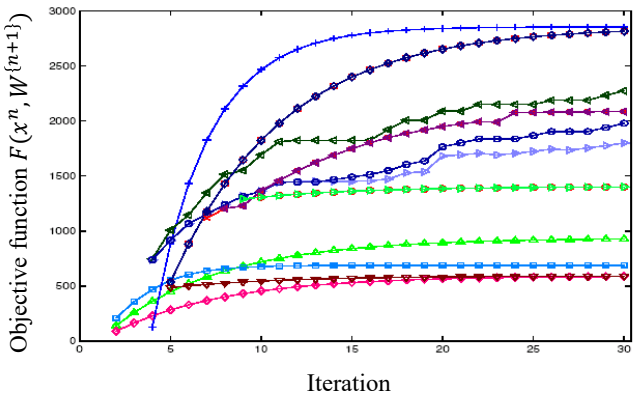


Figure 3.4: Plot of $F(x^n, W^{n+1})$ for different stepsize rules, illustrating different rates of convergence.

observations up through the nth experiment, and let H^0 be the initial empty history. We assume we start with an initial prior probability on μ that we write as

$$p_k^0 = Prob[\mu = \mu_k | H^0].$$

After we have observed $W^1, \ldots, W^n$, we would write our updated distribution as

$$p_k^n = Prob[\mu = \mu_k | H^n].$$

We can update $p^n = (p_k^n)_{k=1}^K$ using Bayes theorem

$$\begin{aligned} p_k^{n+1} &= Prob\mu = \mu_k | W^{n+1} = w, H^n & (3.14) \\ &= \frac{Prob[W^{n+1} = w | \mu = \mu_k, H^n] Prob[\mu = \mu_k | H^n]}{Prob[W^{n+1} = w | H^n]} & (3.15) \\ &= \frac{Prob[W^{n+1} = w | \mu = \mu_k] p_k^n}{Prob[W^{n+1} = w | H^n]} & (3.16) \end{aligned}$$

where

$$Prob[W^{n+1} = w | H^n] = \sum_{k=1}^{K} Prob[W^{n+1} = w | \mu = \mu_k] p_k^n.$$

With this extension to the basic model, we have two probability distributions: the belief on the true mean μ, and the random demand W given μ. To include this extension, we would have to insert p^n into our state variable, so we would write

$$S^n = (x^n, p^n).$$

2) Imagine that our problem is to purchase a commodity (such as oil or natural gas) in month n to be used during month $n+1$. We purchase the commodity at a unit cost c, and sell it up to an unknown demand D^{n+1} at an unknown price p^{n+1}. We would write our objective for this problem as

$$F^n(x^n, W^{n+1}) = p^{n+1} \min(x^n, D^{n+1}) - cx^n. \quad (3.17)$$

For now, assume that p^{n+1} is independent of p^n, and that D^{n+1} is independent of D^n.

a) For the model in equation (3.17), give the state variable S^n and the exogenous information variable W^n.

b) Give the stochastic gradient algorithm for this problem, and show that it is basically the same as the one when price was constant.

3) Now assume that our objective is to optimize

$$F^n(x^n, W^{n+1}) = p^n \min(x^n, D^{n+1}) - cx^n. \quad (3.18)$$

The only difference between equations (3.18) and (3.17) is that now we get to see the price p^n *before* we choose our decision x^n. We know this by how the price is indexed.

a) For the model in equation (3.18), give the state variable S^n and the exogenous information variable W^n.

b) Give the stochastic gradient algorithm for this problem. Unlike the previous problem, this gradient will be a function of p^n.

The situation where the gradient depends on the price p^n is a fairly significant complication. What is happening here is that instead of trying to find an optimal solution x^* (or more precisely, $x^{\pi,N}$), we are trying to find a function $x^{\pi,N}(p)$.

The trick here is to pick a functional form for $x^{\pi,N}(p)$. We suggest two alternatives:

Lookup table - Even if p is continuous, we can discretize it into a series of discrete prices $p_1, \ldots, p_K$, where we pick the value p_k closest to a price p^n. Call this price p_k^n. Now think of a stochastic gradient algorithm indexed by whatever p_k is nearest to p^n. We then use the stochastic gradient to update $x^n(p_k^n)$ using

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

Of course, we do not want to discretize p too finely. If we discretize prices into, say, 100 ranges, then this means we are trying to find 100 order quantities $x^{\pi,N}(p)$, which would be quite slow.

Parametric model - Now imagine that we think we can represent the order quantity $x^{\pi,N}(p)$ as a parametric function

$$x^{\pi,N}(p|\theta) = \theta_0 + \theta_1 p + \theta_2 p^{\theta_3}. \quad (3.19)$$

When we use a parametric function such as this, we are no longer trying to find the order quantity $x^{\pi,N}$; instead, we are trying to find θ that determines the function (in this case, (3.19)). Our stochastic gradient algorithm now becomes

$$\begin{aligned}\theta^{n+1} &= \theta^n + \alpha_n \frac{dF^n(x^{\pi,N}(p^n|\theta^n), W^{n+1})}{d\theta}, \\ &= \theta^n + \alpha_n \frac{dF^n(x^{\pi,N}(p^n|\theta^n), W^{n+1})}{dx} \frac{dx^{\pi,N}(p^n|\theta^n)}{d\theta},\end{aligned}$$

Remember that θ^n is a four-element column vector, while x^n is a scalar. The first derivative is our original stochastic gradient

$$\frac{dF^n(x^{\pi,N}(p^n|\theta^n), W^{n+1})}{dx} = \begin{cases} p - c & x \le W \\ -c & x > W \end{cases}$$

The second derivative is computed directly from the policy (3.19), which is given by

$$\begin{aligned}\frac{dx^{\pi,N}(p^n|\theta^n)}{d\theta} &= \begin{pmatrix} \frac{dx^{\pi,N}(p^n|\theta^n)}{d\theta_0} \\ \frac{dx^{\pi,N}(p^n|\theta^n)}{d\theta_1} \\ \frac{dx^{\pi,N}(p^n|\theta^n)}{d\theta_2} \\ \frac{dx^{\pi,N}(p^n|\theta^n)}{d\theta_3} \end{pmatrix} \\ &= \begin{pmatrix} 1 \\ p^n \\ (p^n)^{\theta_3} \\ \theta_2 (p^n)^{\theta_3} \ln p^n \end{pmatrix}.\end{aligned}$$

Using the parametric model can be very effective if the parametric form matches the true form of the function $x^{\pi,N}(p)$. The lookup table representation is more general, which can be a feature, but

if the discretization is too fine, then it will require a much larger number of iterations to solve.

With these strategies in mind, consider the next three extensions:

4) Return to the objective function in equation (3.17) where the price is only revealed after we make the order decision, but now p^{n+1} depends on history, as in

$$p^{n+1} = p^n + \varepsilon^{n+1}.$$

Discuss how you might approach this problem, given what we presented above.

5) Repeat (4), but now assume that

$$p^{n+1} = 0.5p^n + 0.5p^{n-1} + \varepsilon^{n+1}.$$

6) Repeat (4), but now the quantity x^n is chosen subject to the constraint $0 \leq x \leq R^n$ where

$$R^{n+1} = \max\{0, R^n + x^n - W^{n+1}\},$$

and where the price $p = p^n$ is revealed before we make a decision. With this transition, our problem becomes a traditional inventory problem.

4

Learning the best diabetes medication

4.1 Narrative

When people find they have high blood sugar, typically evaluated using a metric called the "A1C" level, there are several dozen drugs that fall into four major groups:

- Sensitizers - These target liver, muscle, and fat cells to directly increase insulin sensitivity, but may cause fluid retention and therefore should not be used for patients with a history of kidney failure.
- Secretagoges - These drugs increase insulin sensitivity by targeting the pancreas but often causes hypoglycemia and weight gain.
- Alpha-glucosidase inhibitors - These slow the rate of starch metabolism in the intestine, but can cause digestive problems.
- Peptide analogs - These mimic natural hormones in the body that stimulate insulin production.

The most popular drug is a type of sensitizer called metformin, which is almost always the first medication that is prescribed for a new diabetic,

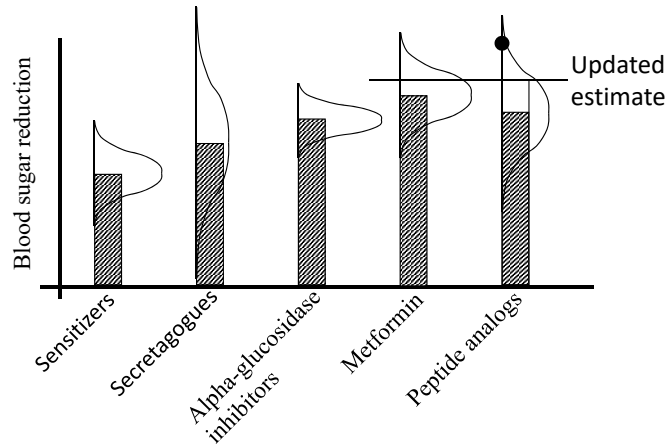


Figure 4.1: Beliefs about the potential that each drug might have on the reduction of blood sugar.

but this does not always work. Prior to working with a particular patient, a physician may have a belief about the potential of metformin, and drugs from each of the four groups, to reduce blood sugar that is illustrated in figure 4.1.

A physician will typically start with metformin, but this only works for about 70 percent of patients. Often, patients simply cannot tolerate a medication (it may cause severe digestive problems). When this is the case, physicians have to begin experimenting with different drugs. This is a slow process, since it takes several weeks before it is possible to assess the effect a drug is having on a patient. After testing a drug on a patient for a period of time, we observe the reduction in the A1C level, and then use this observation to update our estimate of how well the drug works on the patient.

4.2 Basic model

For our basic model, we are going to assume that we have five choices of medications: metformin, or a drug (other than metformin) drawn from one of the four major drug groups. Let $\mathcal{X} = \{x_1, x_2, x_3, x_4, x_5\}$ be the five choices. From observing the performance of each drug over many

Drug	A1C reduction	Std. dev.
Metformin	0.32	0.12
Sensitizers	0.28	0.09
Secretagoges	0.30	0.17
Alpha-glucosidase inhibitors	0.26	0.15
Peptide analogs	0.21	0.11

Table 4.1: Metformin and the four drug classes and the average reduction over the full population.

(that is, millions) of patients, it is possible to construct a probability distribution of the reduction in A1C levels across all patients. The results of this analysis are shown in table 4.1 which reports the average reduction and the standard deviation across all patients. We assume that the distribution of reductions in A1C across the population is normally distributed, with means and standard deviations as given in the table.

To create a model, let

$$\begin{aligned} \bar{\mu}_x^0 &= \text{The mean reduction in the A1C for drug choice } x \text{ across the population,} \\ \bar{\sigma}_x^0 &= \text{The standard deviation in the reduction in A1C for drug } x. \end{aligned}$$

Our interest is learning the best drug for a particular individual. Although we can describe the patient using a set of attributes, for now we are only going to assume that the characteristics of the patient do not change our belief about the performance of each drug for an individual patient.

We do not know the reduction we can expect from each drug, so we represent it as a random variable μ_x, where we assume that μ_x is normally distributed, which we write as

$$\mu_x \sim N(\bar{\mu}_x^0, \bar{\sigma}_x^0).$$

We refer to the normal distribution $N(\bar{\mu}_x^0, \bar{\sigma}_x^0)$ as the *prior distribution of belief* about μ_x.

We index each iteration of prescribing a medication by n which starts at 0. Assume that we always observe a patient for a fixed period of time (say, a month). If we try a drug x on a patient, we make a noisy observation of the truth value μ_x. Assume we make a choice of drug x^n using what we know after n trials, after which we observe the outcome of the $n+1st$ trial, which we denote W^{n+1} (this is the reduction in the A1C level). This can be written

$$W^{n+1} = \mu_x + \varepsilon^{n+1}.$$

Remember that we do not know μ_x; this is a random variable, where $\bar{\mu}_x^n$ is our current estimate of the mean of μ_x.

4.2.1 State variables

Our state variable is our belief about the random variable μ_x which is the true effect of each drug on a particular patient after n trials. S^0 is the initial state, which we write as

$$S^0 = (\bar{\mu}_x^0, \bar{\sigma}_x^0)_{x\in\mathcal{X}},$$

where we also include in S^0 the assumption of normality, which remains through all the experiments. After n experiments, the state is

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

where we no longer include the normality assumption because it is captured in our initial state (the distribution is static, so by convention we do not include it in the dynamic state variable).

Later, we are going to find it useful to work with the *precision* of our belief, which is given by

$$\beta_x^n = \frac{1}{(\bar{\sigma}_x^n)^2}.$$

We can then write our state variable as

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

We are using what is known as a *Bayesian belief model*. In this model, we treat the unknown value of a drug, μ_x, as a random variable with initial prior distribution given by S^0. After n experiments with different drugs, we obtain the *posterior* distribution of belief S^n.

4.2.2 Decision variables

The decision is the choice of medication to try for a month, which we write as

$$\begin{aligned} x^n &= \text{The choice of medication,} \\ &\in \mathcal{X} = \{x_1, \ldots, x_M\}. \end{aligned}$$

We are going to determine x^n using a policy $X^\pi(S^n)$ that depends only on the state variable S^n (along with the assumption of the normal distribution in S^0).

4.2.3 Exogenous information

After making the decision x^n, we observe

$$W_x^{n+1} = \text{The reduction in the A1C level resulting from the drug } x = x^n \text{ we prescribed for the } n+1st \text{ trial.}$$

A reader might wonder why we write the information learned from the decision x^n is written W_x^{n+1} rather than W_x^n. We do this to capture the information available in each variable. Thus, the decision x^0 depends only on the initial state S^0. The state S^n for $n \geq 1$ depends on S^0 along with the observations $W_{x^0}^1, \ldots, W_{x^{n-1}}^n$, but not $W_{x^n}^{n+1}$, since we have not yet made the decision x^n. By letting $W_{x^n}^{n+1}$ be the result of the prescription x^n, we know that x^n cannot depend on W^{n+1}, which would be like seeing into the future.

4.2.4 Transition function

The transition function captures how the observed reduction in A1C, W_x^{n+1}, affects our belief state S^n. Although it takes a little algebra, it is possible to show that if we try drug $x = x^n$ and observe W_x^{n+1}, we can update our estimate of the mean and precision using

$$\bar{\mu}_x^{n+1} = \frac{\beta_x^n \bar{\mu}_x^n + \beta_x^W W_x^{n+1}}{\beta_x^n + \beta_x^W}, \tag{4.1}$$

$$\beta_x^{n+1} = \beta_x^n + \beta^W. \tag{4.2}$$

For all $x \neq x^n$, $\bar{\mu}_x^n$ and β_x^n remain unchanged.

The transition function which we earlier wrote as a generic function

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

in equation (1.12), is given by equations (4.1) - (4.2).

4.2.5 Objective function

Each time we prescribe a drug $x = x^n$, we observe the reduction in the A1C represented by $W_{x^n}^{n+1}$. We want to find a policy that chooses a drug $x^n = X^\pi(S^n)$ that maximizes the expected total reduction in A1C. Our canonical model used $C(S^n, x^n, W^{n+1})$ as our performance metric. For this problem, this would be

$$C(S^n, x^n, W^{n+1}) = W_{x^n}^{n+1}.$$

We write the problem of finding the best policy as

$$\max_\pi \mathbb{E}\left\{\sum_{n=0}^{N-1} W_{x^n}^{n+1} | S_0\right\}, \tag{4.3}$$

where $x^n = X^\pi(S^n)$, and $S^{n+1} = S^M(S^n, x^n, W^{n+1})$. Here, the conditioning on S_0 is particularly important because it carries the prior distribution of belief.

4.3 Modeling uncertainty

Sampling random outcomes for our asset selling problem was relatively simple. For our medical setting, generating outcomes of the random variables $W^1, \ldots, W^n, \ldots$ is a bit more involved.

With the asset selling problem, we were generating random variables with mean 0 and a given variance which we assumed was known. In this medical application, the reduction in the A1C from a particular drug is a noisy observation of the true mean μ_x (for a particular patient) which we can write as

$$W^{n+1} = \mu_x + \varepsilon^{n+1},$$

where ε^{n+1} is normally distributed with mean 0 and a variance (which we assume is known) given by $(\sigma^W)^2$. The real issue is that we do not

know μ_x. Given what we know after n experiments with different drugs, we assume that μ_x is normally distributed with mean $\bar{\mu}_x^n$ and precision β_x^n. We write this as

$$\mu_x|S^n \quad \sim \quad N(\bar{\mu}_x^n, \beta_x^n)$$

where we use the precision β_x^n instead of the more customary variance when we write our normal distribution. We then write the distribution of W^{n+1} as conditioned on μ_x using

$$W^{n+1}|\mu_x \quad \sim \quad N(\mu_x, \beta_x^W).$$

This means that we have to simulate two random variables: the true performance of drug x on our patient, given by μ_x (given our beliefs after n experiments), and then the noise ε^{n+1} when we try to observe μ_x. This just means that instead of generating one normally distributed random variable, as we did in our asset selling problem, we have to generate two.

4.4 Designing policies

A popular class of policies for this problem class is organized under the general name of *upper confidence bounding*. One of the first UCB policies had the form

$$X^{UCB}(S^n) \quad = \quad \arg\max_{x\in\mathcal{X}} \left(\bar{\mu}_x^n + 4\sigma^W\sqrt{\frac{\log n}{N_x^n}}\right), \tag{4.4}$$

where N_x^n is the number of times we have tried drug x. It is standard practice to replace the coefficient $4\sigma^W$ by a tunable parameter, giving us

$$X^{UCB}(S^n|\theta^{UCB}) \quad = \quad \arg\max_{x\in\mathcal{X}} \left(\bar{\mu}_x^n + \theta^{UCB}\sqrt{\frac{\log n}{N_x^n}}\right), \tag{4.5}$$

A popular variant that we have found works surprisingly well was originally introduced under the name *interval estimation*, which his given by

$$X^{IE}(S^n|\theta^{IE}) \quad = \quad \arg\max_{x\in\mathcal{X}} \left(\bar{\mu}_x^n + \theta^{IE}\bar{\sigma}_x^n\right), \tag{4.6}$$

4.5 Policy evaluation

We originally wrote our objective function as

$$\max_{\pi} \mathbb{E}\left\{\sum_{n=0}^{N-1} W_{x^n}^{n+1}|S_0\right\},$$

but writing the expectation in this way is a bit vague. Recall that we have two sets of random variables: the true values of μ_x for all $x \in \mathcal{X}$, and the observations $W^1, \ldots, W^N$ (or more precisely, the noise when we try to observe μ_x). We can express this nested dependence by writing the objective function as

$$\max_{\pi} \mathbb{E}_{\mu}\mathbb{E}_{W^1,\ldots,W^N|\mu}\left\{\sum_{n=0}^{N-1} W_{x^n}^{n+1}|S_0\right\}, \tag{4.7}$$

There are two ways to simulate the value of a policy:

Nested sampling First we simulate the value of the truth μ_x for all $x \in \mathcal{X}$ where we let $\psi \in \Psi$ be a sample realization of μ, which we write as $\mu(\psi)$. We then simulation the observations W, where we let $\omega \in \Omega$ be a sample realization of $W^1(\omega), \ldots, W^N(\omega)$, which means that ω is an outcome of all possible observations over all possible drugs $x \in \mathcal{X}$, over all experiments $n = 1, \ldots, N$.

Simultaneous sampling Here, we let ω be a sample realization of both μ_x and the observations $W^1, \ldots, W^N$.

If we use nested sampling, assume that we generate K samples of the true values $\mu(\psi_k)$, and L samples of the errors $\varepsilon^1(\omega_\ell), \ldots, \varepsilon^N(\omega_\ell)$. For the sampled truth $\mu(\psi_k)$ and noise $\varepsilon^n(\omega_\ell)$, the performance of drug x^n in the $n+1st$ experiment would be

$$W_{x^n}^{n+1}(\psi_k, \omega_\ell) \quad = \quad \mu(\psi_k) + \varepsilon^n(\omega_\ell).$$

We can then compute a simulated estimate of the expected performance of a policy using

$$\overline{F}^{\pi} \quad = \quad \frac{1}{K}\sum_{k=1}^{K}\left(\frac{1}{L}\sum_{\ell=1}^{L}\sum_{n=0}^{N-1} W_{x^n}^{n+1}(\psi_k, \omega_\ell)\right),$$

where $x^n = X^\pi(S^n)$ and $S^{n+1}(\psi_k, \omega_\ell) = S^M(S^n(\psi_k, \omega_\ell), X^\pi(S^n(\psi_k, \omega_\ell)), W^{n+1}(\psi_k, \omega_\ell))$.

If we use simultaneous sampling, then a sample ω determines both the truth $\mu(\omega)$ and the noise $\varepsilon(\omega)$, allowing us to write a sampled estimate of our observation $W_{x^n}^{n+1}$ as

$$W_{x^n}^{n+1}(\omega_\ell) \quad = \quad \mu(\omega_\ell) + \varepsilon^n(\omega_\ell).$$

The estimated value of a policy is given given by

$$\overline{F}^\pi \quad = \quad \frac{1}{L}\sum_{\ell=1}^{L}\sum_{n=0}^{N-1} W_{x^n}^{n+1}(\omega_\ell),$$

where $x^n = X^\pi(S^n)$ and $S^{n+1}(\omega_\ell) = S^M(S^n(\omega_\ell), X^\pi(S^n(\omega_\ell)), W^{n+1}(\omega_\ell))$.

If we use one of our parameterized policies where θ is the tunable parameter, we might write the expected performance as $\overline{F}^\pi(\theta)$ (technically, the parameter θ) as $\overline{F}^\pi(\theta)$. Then, the optimization problem would be

$$\max_\theta \overline{F}^\pi(\theta), \tag{4.8}$$

using any tunable policy.

4.6 Extensions

We have been describing a problem that applies to a single patient. This means that we would have to solve this problem from scratch for each patient. If we have a million diabetes patients, then we would have a million models.

Imagine that we would like to use information from different patients to learn a single model. We can do this by characterizing each patient using a set of attributes $a = (a_1, \ldots, a_K)$. Assume for the moment that each element a_k is discrete (e.g. gender) or discretized (e.g. age, divided into ranges). In fact, we are going to start by assuming that there is a single attribute, gender. Let G^n the gender of the nth patient. Now we have two forms of exogenous information: the gender G^n of the nth patient, and the outcome W^n of the treatment of the nth patient.

We start with a knowledge state K^0 that is our vector $(\bar{\mu}^0, \beta^0)$ introduced earlier in the chapter. The first patient will have gender

G^1, which means our state variable (that is, everything we know) after the first patient arrives is $S^1 = (K^0, G^1)$. We then make a decision x^1 regarding the treatment of patient 1, after which we observe an outcome W^1 describing how the treatment worked. We use this information to obtain an updated knowledge state K^1, after which the process repeats.

$$(K^0, G^1, S^1 = (K^0, G^1), x^1, W^1, K^1, G^2, S^2 = (K^1, G^1), \ldots,$$
$$K^{n-1}, G^n, S^n = (K^{n-1}, G^n), x^n, W^n, K^n, G^{n+1}, \ldots)$$

We pause for a moment and note that our indexing is different from what we used in the basic model. In our basic model, the index n referred to visits by a patient. We make a decision x^n *after* the nth visit using what is known from the first n visits. We let W^{n+1} be the outcome of this treatment, incrementing n to $n+1$ to emphasize that x^n was computed without knowing W^{n+1}.

With our new model, however, n refers to a patient. It makes more sense to let G^n be the gender of the nth patient, at which point we make a decision for the nth patient, and let W^n be the outcome of the treatment for the nth patient. We do not increment n until we see the $n+1st$ patient, at which point we see the gender of the $n+1st$ patient.

Ex. 4.1 — Show how to adapt the policy presented earlier to our problem where gender is the only patient attribute by using a lookup table representation, which means that instead of learning $\bar{\mu}_x^n$, we learn $\bar{\mu}_{a,x}^n$ where $a =$ gender. So, instead of learning an estimate $\bar{\mu}_x^n$ for each treatment x, we have to learn an estimate $\bar{\mu}_{a,x}^n$ for each combination of gender $a = G^n$ and treatment $x = x^n$.

Ex. 4.2 — Now imagine that instead of just gender, we capture age by decade $(0-9, 10-19, ..., 80^+)$, smoker or not, and race (assume eight categories of ethnicity), giving us an attribute vector $a = (a_{gender}, a_{age}, a_{smoker}, a_{race})$. If $a \in \mathcal{A}$, how many elements does $\mathcal{A}$ have? How would this impact your proposed solution in exercise 4.1?

Ex. 4.3 — Imagine that each element a_k in the attribute vector a has L possible values, and that a has K elements, which means that $\mathcal{A}$ has L^K elements. If $L = 10$, what is the largest value of K so that learning our attribute-based model is easier than learning a model for each of 7 million diabetes patients.

Ex. 4.4 — Now imagine that our attribute space $\mathcal{A}$ is simply too large to be practical. What we have done up to now is a lookup table representation where we find an estimate $\bar{\mu}_{a,x}^n$, which becomes problematic when the number of possible values of a becomes large. An alternative approach is to use a parametric model. The simplest would be a linear model where we would write

$$\bar{\mu}_{a,x} = \sum_{f \in \mathcal{F}} \theta_f \phi_f(a, x),$$

where $\phi_f(a, x)$ for $f \in \mathcal{F}$ is a set of features that we (as analysts) would have to define. For example, one feature might just be an indicator of gender, or age range, or race. In this case, there would be a feature for each possible gender, each possible age range, and so on.

- If there are L possible values of each K attributes, what is the minimum number of features we would need?
- Suggest more complex features other than just those that indicate the value of each attribute.

- Contrast the strengths and weaknesses of a lookup table representation versus our linear model.

5

Stochastic shortest path problems - Static

Shortest path problems over graphs are both an important application area (arising in transportation, logistics, and communication), but are also a fundamental problem class that arise in many settings. The most familiar shortest path problem is the classical deterministic problem illustrated in figure 5.1, where we have to find the best path from node 1 to node 11, where the cost of traversing each arc is known in advance.

Shortest path problems build on a fundamental dynamic programming recursion. Let

$$\begin{aligned}
\mathcal{N} &= \text{The set of all nodes in the network (here, this is the nodes } 1, 2, \ldots, 11), \\
\mathcal{N}_i^+ &= \text{The set of all nodes that can be reached directly from node } i, \\
\mathcal{N}_j^- &= \text{The set of all nodes that are connected to node } j, \\
\mathcal{L} &= \text{Set of all links } (i, j) \text{ in the network}, \\
c_{ij} &= \text{The cost of traversing link } (i, j), \text{ where } j \text{ is assumed to be in the set } \mathcal{N}_i^+.
\end{aligned}$$

Now let v_i be the minimum cost to get from node i to the destination

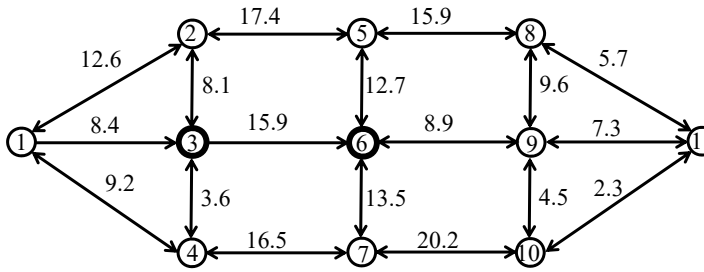


Figure 5.1: Network for a deterministic shortest path problem.

node 11. The values v_i for all nodes $i \in \mathcal{N}$ should satisfy

$$v_i = \min_{j \in \mathcal{N}_i^+} (c_{ij} + v_j). \tag{5.1}$$

We can execute equation (5.1) by initializing v_{11} to zero, and setting all other values to some large number. If we loop over every node i and compute v_i using equation (5.1) repeatedly, the values v_i will converge to the optimal value. This is a very inefficient version of a shortest path algorithm.

Another way to view our network is to assume that each node i is a state S, and let $V_t(S_t)$ be the value of being in state S_t at "time" t. In our shortest path problem, we are going to use t to index the number of links we have traversed on our way from node 1 to the node represented by S_t.

From a state (node) S_t, assume we make a decision we call "x" which would be a decision to traverse a link emanating from the node corresponding to state S_t. We can write this set of decisions as $\mathcal{X}_s$ representing the decisions x that are available to use when we are in state $S_t = s$. Next let $C(s, x)$ be the cost of being in state s and choosing decision x, which would correspond to our link cost c_{ij} in our network above. Finally, we are going to use a "state transition function" that we denote by $S^M(s, x)$ that tells us what state we transition to if we are in state s and take action $x \in \mathcal{X}_s$.

Using this notation, we can rewrite equation (5.1) as

$$V_t(s) = \min_{x \in \mathcal{X}_s} \big(C(s, x) + V_{t+1}(S_{t+1})\big). \tag{5.2}$$

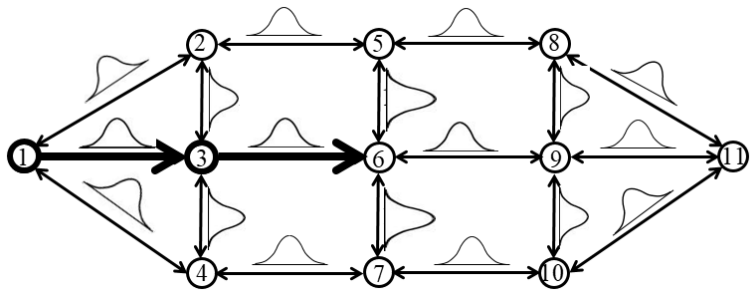


Figure 5.2: Network for a stochastic shortest path problem where distributions are known, but costs are not observed until after decisions are made.

where $S_{t+1} = S^M(s, x)$. We can execute equation (5.2) by setting $V_T(s) = 0$ for a large enough value of T (that is, the largest number of links that we might traverse in a path). Since we might set T too large, we have to add to the set of choices in $\mathcal{X}_s$ the ability to stay at the destination node at time T. We then set $t = T - 1$ and run (5.2) for all states s. We keep repeating this until we get to $t = 0$. When we run the system this way, the time index t is really a counter for how many links we have traversed.

Equation (5.2) is a deterministic version of what is known as Bellman's equation. In the remainder of this exercise, we are going to show how to use Bellman's equation to handle uncertainty in our shortest path problem.

5.1 Narrative

You are trying to create a navigation system that will guide a driverless vehicle to a destination over a congested network. We assume that our system has access to both historical and real-time link costs, from which we can create estimates of the mean and variance of the cost of traversing a link. We can think of this as a shortest path problem where we see distributions rather than actual costs, as depicted in figure 5.2.

We are going to start by assuming that we have to make decisions of which link to traverse based on these distributions. After we traverse a link from i to j, we then experience a sample realization from the

distribution. We want to choose a path that minimizes the expected costs.

5.2 Basic model

We are going to assume that we are trying to traverse the network in figure 5.2 starting at a node q and ending at a destination r.

5.2.1 State variables

In this basic problem, the state S_t is the node where we are located after t link traversals. For the basic problem, it is more natural to say that we are in "state" (node) i, but the state variable will become a lot more interesting in the extensions.

5.2.2 Decision variables

We are modeling decision as the node j that we traverse to given that we are at node i. There is a large community that works on problems that fit this class where the decision is represented as an action a, where a takes on one of a set of discrete values in the set $\mathcal{A}_s$ when we are in state s.

A convenient way to represent decisions is to define

$$x_{tij} = \begin{cases} 1 & \text{If we traverse link } i \text{ to } j \text{ when we are at } i \text{ after } t \text{ traversals,} \\ 0 & \text{Otherwise.} \end{cases}$$

This notation will prove useful when we write our objective function.

5.2.3 Exogenous information

After traversing the link (i, j), we observe

$\hat{c}_{ij}$ = The cost we experience traversing from i to j (which we only observe after traversing the link).

For the moment, we are going to assume that the new observation $\hat{c}_{ij}$ is stored in a very large database. Over time, these observations may have the effect of changing $\bar{c}_{ij}$. Later, we are going to introduce problems where the estimates $\bar{c}_{ij}$ may be just rough approximations of the actual

average link costs, where the decision to traverse link (i, j) will provide better estimates, but that is for later.

5.2.4 Transition function

For our basic graph problem, if we make decision $x_{tij} = 1$, the state $S_t = i$ evolves to state $S_{t+1} = j$.

5.2.5 Objective function

We can model our costs using

$\hat{c}_{ij}$ = A random variable giving the cost to traverse from node i to node j,

$\bar{c}_{ij}$ = The expected value of c_{ij} computed by averaging over our database of past travel costs,

= $\mathbb{E}\hat{c}_{ij}$,

$\bar{\sigma}_{ij}$ = Our estimate of the standard deviation of c_{ij} computed using historical data.

We write $\bar{c}_{ij} = \mathbb{E}\hat{c}_{ij}$ as the expected value of the random variable $\hat{c}_{ij}$ which is the actual link cost, although in practice this is really just an average over past observations.

We assume that we have to make the decision of which link to traverse out of a node i before seeing the actual value of the random cost $\hat{c}_{ij}$. This means that we have to make our decision using our best estimate of $\hat{c}_{ij}$, which would be $\bar{c}_{ij}$.

We could write our objective function using

$$\min_{x_{ij},(i,j)\in\mathcal{L}} \sum_{i\in\mathcal{N}} \sum_{j\in\mathcal{N}_i^+} \hat{c}_{ij} x_{ij}, \tag{5.3}$$

but this formulation would require that we know the realizations $\hat{c}_{ij}$. Instead, we are going to use the expectation, giving us

$$\min_{x_{ij},(i,j)\in\mathcal{L}} \sum_{i\in\mathcal{N}} \sum_{j\in\mathcal{N}_i^+} \bar{c}_{ij} x_{ij}, \tag{5.4}$$

The optimal solution of this problem would be to set all $x_{ij} = 0$, which means we do not get a path. For this reason, we have to introduce

constraints of the form:

$$\sum_{j \in \mathcal{N}_q^+} x_{qj} = 1, \tag{5.5}$$

$$\sum_{i \in \mathcal{N}_r^-} x_{ir} = 1, \tag{5.6}$$

$$\sum_{i \in \mathcal{N}_j^-} x_{ij} - \sum_{k \in \mathcal{N}_j^+} x_{jk} = 0, \quad \text{for } j \neq q, r. \tag{5.7}$$

$$x_{ij} \geq 0, \quad (i,j) \in \mathcal{L}. \tag{5.8}$$

Equations (5.4) - (5.8) represent a *linear program*, and there are powerful packages that can be used to solve this problem when written this way (or, we can use Bellman's equations for a deterministic network that we introduced above). However, this approach does not provide a path for handling uncertainty. Below, we describe how to solve the stochastic shortest path problem using our language of designing policies, which will provide a foundation for addressing uncertainty.

5.3 Modeling uncertainty

For our basic model we are just using the point estimates $\bar{c}_{ij}$ which we assume is just an average of prior observations collected, for example, using travel cost estimates drawn from GPS-enabled smartphones. When estimates are based on field observations, the method is called *data-driven*, which means we do not need a model of uncertainty - we just need to observe it.

For example, let $\bar{c}_{ij}$ be our current estimate of the average travel cost for link (i,j) and assume we just observed a cost of $\hat{c}_{ij}$. We might update our estimate using

$$\bar{c}_{ij} \leftarrow (1-\alpha)\bar{c}_{ij} + \alpha\hat{c}_{ij},$$

where α is a smoothing parameter (sometimes called a learning rate or a stepsize) that is less than 1. However, if we update the estimates $\bar{c}_{ij}$ in this way, then these estimates become dynamic, and would now have to be represented in our state variable.

5.4 Designing policies

Our "policy" for this deterministic problem is a function that maps the "state" (that is, what node we are at) to an action (which link we move over). We can solve this problem by optimizing the linear program represented by equations (5.4) - (5.8), which gives us the vector x^*_{ij} for all links (i, j). We can think of this as a function where given the state (node i) we can an action, which is the link (i, j) for which $x_{ij} = 1$. We can write this policy as a function $X^\pi(S_t)$ using

$$X^\pi(S_t = i) = j \quad \text{if } x_{ij} = 1.$$

Alternatively, we can solve Bellman's equation as we did initially for our deterministic shortest path problem using equation (5.1). This gives us a value v_i which is the minimum travel cost from each node i to the destination node r. Once these values are computed, we can make decisions using the following policy

$$X^\pi(i) = \arg\min_{j \in \mathcal{N}_i^+} (\bar{c}_{ij} + v_j). \tag{5.9}$$

This means that our "stochastic" shortest path problem can be solved just we solved our deterministic problem. Below we show that with a minor twist, the situation changes dramatically.

We can introduce a slight variant by assuming that our estimated costs depend on our "time" variable. Thus, instead of writing $\bar{c}_{ij}$, we would write $\bar{c}_{tij}$, in which case we would write Bellman's equation as

$$V_t(i) = \min_{j \in \mathcal{N}_i^+} (\bar{c}_{tij} + V_{t+1}(j)). \tag{5.10}$$

Here, we would pick a large time T and set $V_T(r) = 0$ for our destination node r. We would then set $V_T(i)$ to a large number for all other nodes i other than r. Finally, we simply execute equation (5.10) by stepping backward in time. This is not a very efficient shortest path algorithm, but it is easy to accelerate by limiting the loop over all nodes i at time t to nodes that can be reached from nodes j that we have already reached.

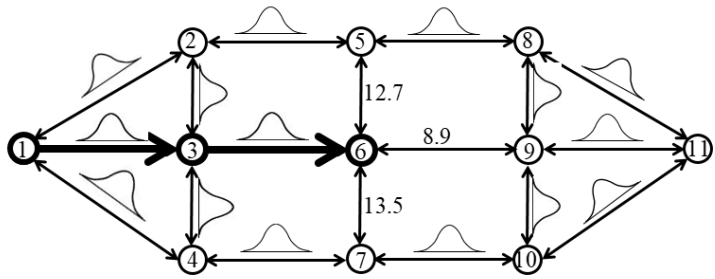


Figure 5.3: Network for a stochastic shortest path problem where travelers get to see the link costs before making a decision.

5.5 Policy evaluation

Policy evaluation for this problem is not required, because the policy is optimal. We will not see this very often.

5.6 Extensions

The stochastic shortest path problem is a foundational problem class for many applications. It should then not be surprising that we can introduce minor twists where the problem suddenly becomes much more complicated. In the process, we will lose our ability to find optimal policies using Bellman's equation.

5.6.1 Stochastic shortest path - Adaptive

We are going to change the information that we can use while making decisions. In our first stochastic shortest path problem, we assumed that we had to choose the next link to traverse *before* we see the actual travel cost over the link. Now assume that we get to see the travel cost *before* we make a decision, which means we make our decision using the actual cost $\hat{c}_{ij}$ rather than its expectation (or average) $\bar{c}_{ij}$. This is illustrated in figure 5.3.

If we pretend for the moment that someone can give us the values v_j which is the minimum travel cost from node j to our destination node r, an optimal policy for choosing the next downstream node would be

written

$$X^{\pi}(i) = \arg\min_{j \in \mathcal{N}_i^+} (\hat{c}_{ij} + v_j). \tag{5.11}$$

The problem here is that we cannot compute v_j as we did before using Bellman's equation (5.1). In fact, the switch to seeing the costs before we make a decision requires a fundamental change to our basic model.

In our deterministic model, or basic stochastic model, the state variable S_t after "t" transitions was the node i where the traveler was located. This was the only information we need at that point in time.

In our new stochastic model, just capturing the node where our traveler is located is no longer enough. Remember that above, we introduced a state variable as "all the information we need at time t (or after n iterations) from history to model the system from time t (or n) onward." While this description is accurate, it is useful to introduce a more complete definition:

Definition 5.6.1. A **state variable** is: A function of history that is necessary and sufficient to compute the cost/contribution function, the constraints, and any information required to model the evolution of information needed in the cost/contribution function and the constraints.

Our new stochastic shortest path problem introduces new information that is needed to make a decision: the costs out of the node where we are located. We are going to find it convenient to introduce two new state variables:

$$\begin{aligned} R_t &= \text{The physical state of the system, which is usually controlled directly by decisions,} \\ I_t &= \text{Other information that we need to make a decision.} \end{aligned}$$

In our network problem, our physical state would be the node where we are located, while the "other information" variable I_t would capture the costs on links out of the node where we are located. Assume that at time t that $R_t = i$. We are going to add time t to our index for link costs, which means we will replace $\hat{c}_{ij}$ with $\hat{c}_{tij}$ to refer to the cost when

we go from i to j at time t. We might then write

$$\begin{aligned} S_t &= (R_t, I_t) \\ &= \left(i, (\hat{c}_{tij})_{j \in \mathcal{N}_i^+}\right). \end{aligned}$$

To see what this does to our previous way of solving our shortest path problem, take a fresh look at Bellman's equation as we first introduced it in equation (5.2)

$$V_t(S_t) = \min_{x \in \mathcal{X}_s} \left(C(S_t, x) + V_{t+1}(S_{t+1})\right). \tag{5.12}$$

where S_{t+1} would be given by

$$\begin{aligned} S_{t+1} &= (R_{t+1}, I_{t+1}), \quad \text{where:} \\ R_{t+1} &= \text{The node produced by our decision } x, \\ &\quad \text{So if } x_{ij} = 1, \text{ then } R_{t+1} = j \\ I_{t+1} &= \text{The costs that are observed out of node } R_{t+1}, \text{ which depends on the decision } x. \text{ If } x \text{ sends us to node } j \text{ so that } R_{t+1} = j, \text{ then } I_{t+1} = (\hat{c}_{t+1,jk_1}, \hat{c}_{t+1,jk_2}, \hat{c}_{t+1,jk_3}) \text{ (assuming there are three links out of node } j). \end{aligned}$$

As we can see, even if we assume that the costs $\hat{c}_{t+1,j}$ are discrete, the state space just grew dramatically. Imagine that we have discretized costs into buckets of 20 values. If there are three links out of every node, our state space has grown from the number of nodes, to $20 \times 20 \times 20 = 8,000$ times larger.

Assume that $R_t = i$. The cost function $C(S_t, x)$ is given by

$$C(S_t, x) = \sum_{j \in \mathcal{N}_i^+} \hat{c}_{tij} x_{ij}.$$

Remember that S_t (where $R_t = i$) contains the costs $\hat{c}_{tij}$ for the links (i, j) out of i, so these are known.

So, we have a bit of a problem. At time t (which is when we are computing $V_t(S_t)$ in equation (5.12)), the link costs I_{t+1} out of downstream node R_{t+1} (which is determined by the decision x) are

not known. Said differently, I_{t+1} is a random variable at time t, which means we cannot even calculate $V_{t+1}(S_{t+1})$. We fix this by taking the expectation, which we write as

$$V_t(S_t) = \min_{x \in \mathcal{X}_s} \left(C(S_t, x) + \mathbb{E}\{V_{t+1}(S_{t+1}) | S_t, x\} \right). \tag{5.13}$$

The expectation operator $\mathbb{E}$ should be viewed as taking an average over the possible path costs that a traveler might encounter once they arrive at node R_{t+1} when making decision x (which determines R_{t+1}).

We can write this more explicitly. Assume that x sends us to node j (which means that $x_{ij} = 1$), and when we arrive we see $I_{t+1} = (\hat{c}_{t+1,jk_1}, \hat{c}_{t+1,jk_2}, \hat{c}_{t+1,jk_3})$. We learn these costs when we arrive to node j at time $t+1$, but they are random when we are at node i at time t thinking about what to do.

Now we face a new challenge: how do we compute the expectation $\mathbb{E}\{V_{t+1}(S_{t+1})|S_t, x\}$? Assume that each cost $\hat{c}_{t+1,jk}$ can take on values $c_1, c_2, \ldots, c_L$ with probabilities $p_{jk}(c_\ell)$. For example, c_1 might be 1 minute, c_2 might be 2 minutes, and so on. The probability $p_{jk}(c_\ell)$ is the probability that $\hat{c}_{t+1,jk} = c_\ell$. Assume that the decision x takes us to node j, after which we face a choice of traveling over links $(j, k_1), (j, k_2)$ or (j, k_3). We would compute our expectation using

$$\begin{aligned} \mathbb{E}\{V_{t+1}(S_{t+1})|S_t, x\} \quad = \quad & \sum_{\ell_1=1}^{L} p_{jk_1}(c_{\ell_1}) \sum_{\ell_2=1}^{L} p_{jk_2}(c_{\ell_2}) \sum_{\ell_3=1}^{L} p_{jk_3}(c_{\ell_3}) \\ & V_{t+1}(S_{t+1} = (j, (c_{\ell_1}, c_{\ell_2}, c_{\ell_3}))). \end{aligned} \tag{5.14}$$

To put it bluntly, equation (5.14) is pretty ugly. Those triple summations are going to be hard to compute.

The expectation is not even the worst of our problem. To use Bellman's equation in equation (5.13) (or (5.2)), we have to compute $V_t(S_t)$ for every possible state S_t. When the state was just a node, that is not too bad, even if there are thousands (even tens of thousands) of nodes. However, adding the information variable I_t to the state makes the problem dramatically harder. Imagine that there are 10 possible values of each cost variable $\hat{c}_{tij}$. That means there are 1,000 possible values of I_t. If our network has 10,000 nodes (that is, R_t can take on 10,000 values), then S_t can now take on $10,000 \times 1,000 = 10,000,000$ values.

This is our first peek at what happens when a state variable becomes a vector. The number of possible values of the state variable grows exponentially, a process known widely as the *curse of dimensionality.*

All is not lost for this problem. There is a trick we can use that allows us to overcome the curse of dimensionality for this particular problem. The main computational challenge with Bellman's equation in equation (5.13) is the expectation operator, which is easily the most dangerous piece of notation solving sequential decision problems.

We are going to use two powerful strategies to overcome this problem in this setting (and we are going to use these strategies for other settings). First, we introduce the idea of the *post decision state* which we designate S_t^x. The post-decision state is the state of the system *after* we make a decision.

To see pre- and post-decision states, return to figure 5.3. As we saw before, our pre-decision state (which we call "the state") S_t is

$$S_t = (6, (12.7, 8.9, 13.5)).$$

Once we have made a decision, we are still at node 6, but imagine that the decision we made was to go to node 9. We might describe our physical post-decision state $R_{t+1}^x = 9$, which we might alternatively state as "going to node 9." However, we no longer need those troublesome observations of costs on the links out of node 6, given by $(\hat{c}_{t+1,6,5}, \hat{c}_{t+1,6,9}, \hat{c}_{t+1,5,7}) = (12.7, 8.9, 13.5)$. This means that our post-decision state is

$$S_t^x = (9).$$

Using the post-decision state, we are going to break Bellman's equation into two steps. Instead of going from S_t to S_{t+1} to S_{t+2} as we are doing in Bellman's equation (5.13), we are going to first step from the pre-decision state S_t to post-decision state S_t^x which we do by rewriting equation (5.13) as

$$V_t(S_t) = \min_{x \in \mathcal{X}_s} \left(C(S_t, x) + V_t^x(S_t^x) \right), \tag{5.15}$$

where V_t^x is the value of being at post-decision state S_t^x. Note that we no longer have the expectation, because by construction the post-decision state involves a given decision x (e.g. "go to node 9") but no

new information (which is the random part). Note that in this case, the post-decision state S_t^x consists of just the node, which means it is much simpler than S_t.

We are not out of the woods. We still have to compute $V_t^x(S_t^x)$, which is done using

$$V_t^x(S_t^x) = \mathbb{E}\{V_{t+1}(S_{t+1})|S_t, x\}. \tag{5.16}$$

So, we still have to compute that expectation, and it has not gotten any easier. Assume that our decision x is to go to node j (which means that $x_{ij} = 1$), and let

$$\hat{c}_{t+1,j} = (\hat{c}_{t+1,jk},\ k \in \mathcal{N}_j^+)$$

be the set of link costs out of node j. Our next pre-decision state S_{t+1} would then be

$$S_{t+1} = (j, \hat{c}_{t+1,j}).$$

Now assume that we have a way of sampling possible values of $\hat{c}_{t+1,j}$. We might do this from a database of historical observations of link costs, or we might build a probability distribution from past data and sample from this. Assume we are going to do this iteratively, and let $\hat{c}_{ij}^n$ be the nth sample of the link cost from i to j.

We are going to construct approximations $\overline{V}_t^x(j)$ of the value of being at node j, where

$$\overline{V}_t^{x,n}(j) \approx \mathbb{E}\{V_{t+1}(S_{t+1})|S_t\}.$$

Let $\overline{V}_t^{x,n}(j)$ be our approximation of $\mathbb{E}\{V_{t+1}(S_{t+1})|S_t\}$ after n samples. One way to build this approximation is to use samples of the value of being at node j. Imagine that we are going to pass forward through the network, making decisions using approximations $\overline{V}_t^{x,n-1}(S_t)$ obtained from previous iterations, along with sampled costs $\hat{c}_{ij}^n$. We can obtain a sampled estimate of the value of being in state S_t using

$$\hat{v}_t^n(i) = \min_{j \in \mathcal{N}_i^+} (\hat{c}_{ij}^n + \overline{V}_t^{x,n-1}(j)). \tag{5.17}$$

We are then going to update our estimate of the value of being at node i using

$$\overline{V}_t^{x,n}(i) = (1 - \alpha_n)\overline{V}_t^{x,n-1}(i) + \alpha_n \hat{v}_t^n(i). \tag{5.18}$$

Here, α_n is known as a smoothing factor or learning rate, but for technical reasons it is also known as a "stepsize." We might just fix this equal to a constant such as $\alpha_n = .1$ or .05, but a common strategy is to use a declining formula such as

$$\alpha_n = \frac{\theta^\alpha}{\theta^\alpha + n - 1}, \tag{5.19}$$

where θ^α is a tunable parameter. For example, if we set $\theta^\alpha = 1$, we get $\alpha_n = 1/n$. In this case, it is possible to verify the equation (5.18) is averaging over the values $\hat{v}_t^n(i)$. In practice, this formula is unlikely to work well for this problem because the stepsize goes to zero too quickly.

One challenge with equation (5.17) is that we are going to need initial values for $\overline{V}_t^{x,0}(i)$. A natural choice would be to solve the deterministic version of this problem where the costs $\hat{c}_{ij}$ are set equal to estimates of their means, and then obtain initial estimates of the cost incurred to get from each node to the destination.

Equation (5.17), however, is simply not a good sampling method for shortest path problems. A better method is to use the estimates $\overline{V}^{x,n-1}(i)$ to make decisions using

$$x_t^n(i) = \arg\min_{j \in \mathcal{N}_i^+} (\hat{c}_{ij}^n + \overline{V}_t^{x,n-1}(j)). \tag{5.20}$$

The decision $i_t^n = x_t^n(i)$ gives us the next node after node i based on the sampled costs $\hat{c}_{ij}^n$ and the estimates of the cost $\overline{V}_t^{x,n-1}(j)$ to get from node j to the destination node r. When we arrive at r, we have an entire path consisting of the nodes

$$(q, i_1^n, i_2^n, \ldots, r).$$

We also have the sampled costs $\hat{c}^n_{i_t^n, i_{t+1}^n}$ over the entire path. Assume there are T links in the path. We can then traverse backward over the path starting with $\hat{v}_T^n(r) = 0$, and computing

$$\hat{v}_t^n(i_t^n) = \hat{c}^n_{i_t^n, i_{t+1}^n} + \hat{v}_{t+1}^n(i_{t+1}^n). \tag{5.21}$$

We can then use these estimates in our smoothing process in equation (5.18).

This procedure is a form of *approximate dynamic programming* (alternatively known as *reinforcement learning*). More specifically, it is a

form of *forward approximate dynamic programming* since it progresses by stepping forward through time. We have illustrated a pure forward pass procedure using equation (5.17), which requires single passes through the network, and a double pass procedure using equation (5.21), which consists of first stepping forward through the graph simulating decisions, and then backward for updating the value of being at each state.

This method is very robust with respect to complex pre-decision states. For example, we do not care how many links might be going out of each node), but we are leveraging the fact that our post-decision state variable is fairly simple (in this case, it is just the node where we are sitting.

6

Stochastic shortest path problems - Dynamic

6.1 Narrative

We are going to tackle stochastic shortest paths again, but now we are going to do it just as it is done in Google maps (or any commercial navigation system). We all recognize that transportation networks often have predictable patterns of congestion, along with random variations that happen in the natural course of events. For example, an accident might create a backlog where we might estimate how the travel delays might evolve as a result of the accident.

The point of departure from the static shortest path problem is that our estimates of costs in the future are evolving over time. We are going to return to the problem where costs are stochastic, but when we arrive at a node i, we do not see the actual realizations of the costs out of node i. However, we are going to assume that we are given updated estimates of costs over the entire network. These estimates can be viewed as a forecast; we are going to assume that the actual cost that we incur when we traverse an arc will, on average, equal the forecast (that is, the forecasts are unbiased), but these forecasts will evolve over time.

6.2 Basic model

Assume that when we have to make a decision at time t, we have an updated estimate of travel costs based on *current* congestion levels (by tracking the speed at which our smartphones are moving through traffic). We are going to represent these times using

$$\begin{aligned}\tilde{c}_{t,k\ell} &= \text{The estimated cost of traversing link } (k,\ell) \text{ at time } t, \text{ using estimates based on what we know at time } t.\end{aligned}$$

For now, we are not going to try to model the cost if we arrive at a point in time $t' > t$ given what we know at time t. So, we may estimate, at 3pm, that we are going to arrive at a link at 5pm, but we are going to use our 3pm estimate (as google does now).

6.2.1 State variables

A traveler at node $R_t = i$ at time t is assumed to be given a set of forecasts

$$\begin{aligned}\tilde{c}_t &= (\tilde{c}_{tt,k\ell})_{k,\ell\in\mathcal{N}},\\ &= \text{The vector of estimates of the cost to traverse link } (k,\ell) \text{ at time } t, \text{ given what is known at time } t.\end{aligned}$$

The traveler's state S_t at time t is then

$$S_t = (R_t, \tilde{c}_t).$$

Note that this state variable is *very* large; it consists of a vector of estimates of link costs for *every* link in the network.

6.2.2 Decision variables

The decision variables are the same as with the static stochastic shortest path problem:

$$x_{tij} = \begin{cases} 1 & \text{If we traverse link } i \text{ to } j \text{ when we are at } i \text{ at time } t,\\ 0 & \text{Otherwise.}\end{cases}$$

This decision has to obey the constraint that we do *something* when we are in state $R_t = i$ as long as i is not the destination. We write this constraint as

$$\sum_j x_{t,R_t,j} = 1 \text{ for } i \text{ other than the destination.} \tag{6.1}$$

If we are at the destination, then we do nothing, and instead write $x_{t,ij} = 0$ for i equal to the destination, and j any other node.

As above, we let $X^\pi(S_t)$ be our policy for determining the vector x_t which we assume has to satisfy the constraint (6.1).

6.2.3 Exogenous information

There are two types of exogenous information for this problem:

$$\hat{c}_{t+1,ij} = \text{This is the cost of traversing link } (i,j) \text{ after the traveler made the decision at time } t \text{ to traverse this link.}$$

The second type of new information is the updated estimates of the link costs. We are going to model the exogenous information as the change in the estimates:

$$\begin{aligned} \delta\tilde{c}_{t+1,k\ell} &= \tilde{c}_{t+1,k\ell} - \tilde{c}_{tk\ell}, \\ \delta\tilde{c}_{t+1} &= (\tilde{c}_{t+1,k\ell})_{(k,\ell)\in\mathcal{N}}. \end{aligned}$$

Our exogenous information variable, then, is given by

$$W_{t+1} = (\hat{c}_{t+1}, \delta\tilde{c}_{t+1})$$

6.2.4 Transition function

We are assuming that $\hat{c}_{t+1}$ arrives as exogenous information (we could have let the exogenous information be the change in the costs, but this is more natural).

The transition function for the forecasts evolves according to

$$\tilde{c}_{t+1,k\ell} = \tilde{c}_{tk\ell} + \delta\tilde{c}_{t+1,k\ell}. \tag{6.2}$$

Finally, we update the physical state R_t using

$$R_{t+1} = \{j | x_{t,R_t,j} = 1,\}. \tag{6.3}$$

In other words, if we are at node $i = R_t$ and we make the decision $x_{tij} = 1$ (which requires that we be at node i, since otherwise $x_{tij} = 0$), then $R_{t+1} = j$.

The updating of $\hat{c}_{t+1}$, equation (6.2) for the forecasts $\tilde{c}_{t+1}$ and equation (6.3) for our physical state R_t, make up our transition equation

$$S_{t+1} = S^M(S_t, X^\pi(S_t), W_{t+1}).$$

6.2.5 Objective function

We now write our objective function as

$$\min_\pi \mathbb{E} \sum_{t=0}^{T} \sum_{(i,j)\in\mathcal{N}} \hat{c}_{t+1,i,j} X^\pi(S_t). \tag{6.4}$$

Note that our policy $X^\pi(S_t)$ makes the choice of the next link we move to given what we know at time t, captured by S_t.

6.3 Modeling uncertainty

In practice, the dynamic updating of costs (and forecasts) come from real systems, which means they are *data driven*. When this is the case, we do not use a mathematical model of the link costs. The alternative is to have a mathematical model of the random information W^{n+1}

If we wish to run simulations, then we face the challenge of modeling the realization of the costs captured by $\hat{c}_t$, as well as the sequence of forecasts. Considerable care has to be given to this model. First, the updated forecast $\bar{c}_{t+1}$ has to be drawn from a distribution with mean $\bar{c}_t$. In addition, the realizations $\hat{c}_{t+1}$ have to be drawn from a distribution with mean $\bar{c}_t$.

With this in mind, a realistic network simulator should try to capture real traffic conditions, whether they represent the evolution of traffic delays, or the effect of weather patterns that may affect an entire network.

6.4 Designing policies

A quick hint that we are not going to be using Bellman's equation (even approximately) is the size of the state variable, which now includes

forecasts of travel costs on every link in the network.

Instead, we are going to base our policy on a special model that we call a *lookahead model*. Variables in the lookahead model are designated by using tilde's. Lookahead variables are typically indexed by two time indices: time t, which is the time at which a decision is being made, and a second index t' which is the time within the lookahead model.

Assume we are at a node i planning our trip into destination node r. We are going to solve this problem deterministically, treating the cost estimates $\tilde{c}_{tt',k\ell}$ as the correct cost. We do this by solving the deterministic shortest path problem we solved in our basic model in section 5. There are three ways we might do this:

Static lookahead model In a static lookahead model, we use our best estimate of the time for a link (k,ℓ) as $\bar{c}_{tk\ell} = \tilde{c}_{tt',k\ell}$. We then solve Bellman's equation as

$$\tilde{V}_{tt'}(j) = \min_{(\tilde{x}_{tt'jk},k\in\mathcal{N}_j^+)} \left(\sum_{k\in\mathcal{N}_j^+} \bar{c}_{tjk}\tilde{x}_{tt'jk} + \tilde{V}_{t,t'+1}(k) \right). \quad (6.5)$$

The variables $\tilde{x}_{tjk}$ represent the decisions in this static model using the information as we know it at time t. Once we have our value functions $\tilde{V}_{tt'}(j)$ for all nodes j, we can compute the optimal decision we would make when we are at node j, which we designate $\tilde{x}^*_{tt'}(j)$ using

$$\tilde{x}^*_{tt'}(j) = \arg\min_{(\tilde{x}_{tt'jk},k\in\mathcal{N}_j^+)} \left(\sum_{k\in\mathcal{N}_j^+} \bar{c}_{tjk}\tilde{x}_{t,t'+1,jk} + \tilde{V}_{t,t'+1}(k) \right) \quad (6.6)$$

We run the algorithm by starting at a time $t' = t + H$ for an appropriate horizon H, where H is large enough to ensure that we can get to our destination in H time periods. We then execute equation (6.5) stepping backward in time. Since we may arrive to our destination early, we have to include the ability to sit and wait at the destination node r if we arrive earlier than time $t + H$.

We do not need the decision $\tilde{x}^*_{tt'}(j)$ for all nodes j and time periods $t' \geq t$. If we are at node i at time t in the base model, we only

need $\tilde{x}^*_{tt}(i)$ which tells us which node to move toward next. Once we make this decision, we obtain new costs $\tilde{c}_{t+1,t',jk}$ for all the links, solve Bellman's equation, and then get the decision we need to make at node j.

Deterministic dynamic model Now assume that we are given cost estimates $\tilde{c}_{tt',k\ell}$ that are projections of the cost that we think we will incur when we traverse link (k,ℓ) at time t' in the future.

$$\tilde{V}_{tt'}(j) = \min_{(\tilde{x}_{tt',jk},k\in\mathcal{N}^+_j)} \left(\sum_{k\in\mathcal{N}^+_j} \tilde{c}_{tt',jk}\tilde{x}_{tt',jk} + \tilde{V}_{t,t'+1}(k) \right) \quad (6.7)$$

This is a deterministic model of the future. Although we are trying to solve a stochastic shortest path problem, we are approximating the lookahead model using deterministic estimates of costs (just as we did with the static model).

This works just like the static version, except that now we index the value function by time t'. It is important to recognize that t' is the true time variable in this model, since this is the time at which activities are happening within our model. The index t is just capturing the fact that we are solving the problem at time t, using information as we know it at time t. Since we are currently at node i at time t, our decision would be

$$\tilde{x}^*_{tt'}(i) = \arg\min_{(\tilde{x}_{tt'ij},j\in\mathcal{N}^+_i)} \left(\sum_{j\in\mathcal{N}^+_i} \tilde{c}_{tt'ij}\tilde{x}_{tt'ij} + \tilde{V}_{t,t'+1}(j) \right). \quad (6.8)$$

Stochastic dynamic model Finally, we might want to capture uncertainty as we look into the future in the hope that this might produce better decisions now. It is important to keep in mind that just as we are able to choose whether to use deterministic static or dynamic approximations of the future, we can also make choices about how we wish to represent uncertainty, or even how we model the sequencing of decisions and information. For example, even if our problem allows us to see the costs on the links out of node i (where we are located), we could assume that in the lookahead

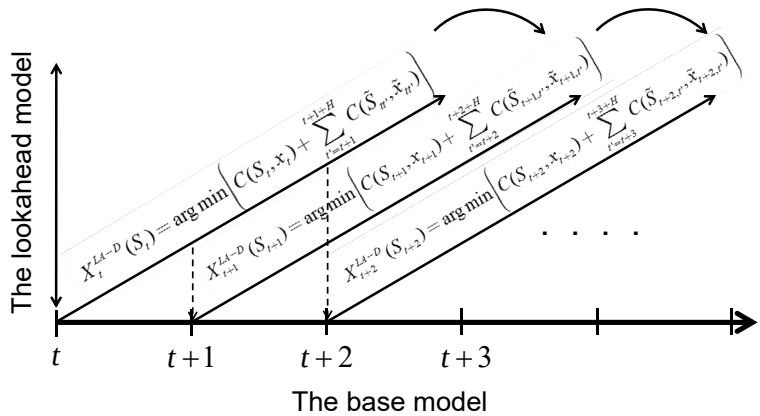


Figure 6.1: Illustration of simulating a direct lookahead policy, using a deterministic model of the future.

model that the reverse is true, which, as we have seen, means that we now have a deterministic shortest path.

But what if we want to make the lookahead model the same as the base model? Now we are back to the adaptive stochastic shortest path problem that we introduced and solved (using approximate dynamic programming) in section 5.6.1. Problem solved? The difficulty is that we have to re-optimize the lookahead model fairly frequently, possibly in under a minute.

Figure 6.1 illustrates a rolling lookahead process. At times t, $t+1$ and $t+2$, we create and solve a lookahead model using estimates of costs as we know them. We then solve our shortest path problem, which is represented in the decisions $\tilde{x}_{tt'}(j)$ for all nodes j, but then we only implement the decision $\tilde{x}_{tt}(i)$ for the node i where we are located at time t.

Our first two lookahead models are themselves deterministic, dynamic programs. These can be solved using Bellman's equation, but note that our state variable is just the node, so we might ask: why don't we have to model the forecast in the state variable as we did with our original model? The reason is that the forecasts evolve over time in the

base model, but not in the lookahead model. When we look forward deterministically, we imbed the forecast in the lookahead, but do not model its evolution.

When we hold a dynamically changing variable constant in a lookahead model, we refer to this variable as a *latent variable* in the lookahead model. This is one of a number of different types of approximations that can be made in a lookahead model. The most obvious approximation we are making is that we use a deterministic future, but we could hold the forecast constant even if we assume the future is stochastic.

The lookahead model, then, is its own model with its own characteristics, which is the reason why we use variables with tilde's - this is how we make the distinction between our base model, which uses variables such as S_t and x_t, and the lookahead model, where we use variables such as $\tilde{S}_{tt'}$ and $\tilde{x}_{tt'}$.

7

Designing policies

By now, we have introduced different ways to solve problems. These include:

1) Asset selling - Rule-based decisions.

2) Adaptive market planning - Gradient-based search.

3) Learning the best diabetes treatment - Upper confidence bounding policies.

4) Static stochastic shortest paths - Policies based on Bellman's equation.

5) Dynamic stochastic shortest paths - Deterministic lookahead with static or time-dependent costs.

6) Value of information policies for learning bids.

It turns out that each of the policies are examples from four fundamental classes of policies. Below we describe the four classes of policies, two of which require searching over parameterized classes. Depending on the properties of the problem, we perform the search over a parameterized class using either derivative-based or derivative-free methods.

7.1 The four classes of policies

We first observe that these four strategies can be divided into two categories, each of which can then be further divided into two subclasses:

Policy search The "policy search" class of policies involves searching over a set of functions for making decisions to find the function that works the best. Most of the time this will mean searching for the best value of a set of parameters that characterized a parameterized policy.

PFAs Policy function approximations - These are analytical functions that directly map a state to an action. Some examples are:

- A parameterized function such as the high-low policy given in equation (2.14), which we repeat here:

$$X^{high-low}(S_t|\theta^{high-low}) = \begin{cases} 1 & \text{If } p_t < \theta^{low} \text{ or } p_t > \theta^{high} \\ 1 & \text{If } t = T \text{ and } R_t = 1 \\ 0 & \text{Otherwise} \end{cases}$$

- A linear function, such as

$$X^{\pi}(S_t) = \theta_0 + \theta_1\phi_1(S_t) + \theta_1\phi_1(S_t) + \ldots + \theta_F\phi_F(S_t)$$

  where $(\phi_f(S_t))$, $f = 1, \ldots, F$ is a set of features. For example, we might be trying to decide how much to bid to have a movie advertised on a website, and a feature might be the genre of the movie or the name of the lead actor or actress.
- Advanced functions such as neural networks, although these typically have large numbers of parameters (the weights in a neural network) that need to be optimized over.

CFAs Cost function approximations - These are policies that require solving a parameterized optimization problem, where we may parameterize either the objective function or the constraints. Some examples are

- A simple example of a parameterized cost function approximation is the interval estimation policy we introduced in equation (4.6) and repeat here

$$X^{IE}(S^n|\theta^{IE}) = \arg\max_{x\in\mathcal{X}} \left(\bar{\mu}_x^n + \theta^{IE}\bar{\sigma}_x^n\right).$$

- We could parameterize the static shortest path problem by replacing the expected costs $\bar{c}_{ij}$ with, say, the θ-percentile of the costs, suggesting that we write the costs as $\bar{c}_{ij}(\theta)$. So, we might use the 80th percentile as a way of protecting ourselves against the potential of long travel times.
- Scheduling an airline - Airlines solve complex optimization problems to optimize the movement of their aircraft and crews, but this has to be done in the face of significant weather delays. As with the shortest path problem, the airline could use the θ-percentile of the travel times.

Both PFAs and CFAs have parameters that have to be tuned. The only difference is whether the policy involves an imbedded optimization problem or not. Both are exceptionally powerful and are widely used in different settings.

Lookahead approximations Policies based on lookahead approximations are constructed by approximating the downstream costs (or rewards) from making a decision now, which are then considered along with the initial cost (or reward) of the initial decision.

VFAs Policies based on value function approximations - These are policies that are based on Bellman's equation. The most basic form of Bellman's equation for stochastic problems was first presented in equation (5.13), and is given by

$$V_t(S_t) = \min_{x\in\mathcal{X}_s} \left(C(S_t,x) + \mathbb{E}\{V_{t+1}(S_{t+1})|S_t,x\}\right).$$

In practice, we typically have to replace the value function $V_{t+1}(S_{t+1})$ with an approximation $\overline{V}_{t+1}(S_{t+1})$. The field that studies these approximations goes under names such

as approximate dynamic programming, reinforcement learning (which originated in computer science), and adaptive dynamic programming (the term used in the engineering controls community). In this case, the policy would be given by

$$X^\pi(S_t) = \arg\min_{x \in \mathcal{X}_s} \big(C(S_t, x) + \mathbb{E}\{\overline{V}_{t+1}(S_{t+1})|S_t, x\}\big).$$

If we use the post-decision state S_t^x, we can write our policy as

$$X^\pi(S_t) = \arg\min_{x \in \mathcal{X}_s} \big(C(S_t, x) + \overline{V}_t^x(S_t^x)\big).$$

We used the deterministic shortest path problem to illustrate an application where value functions could be computed exactly. This can sometimes be done in stochastic problems, but in most applications, it has to be done approximately. The challenge is doing computations that are of sufficiently high quality to produce effective policies.

DLAs Direct lookahead approximations - The first three classes of policies require finding some form of a functional approximation: the policy (for PFAs), the function being optimized (for CFAs), or the value of being in a downstream state (for VFAs). However, there are many problems where these functional approximations are just not possible. Instead, we have to resort to direct lookaheads, where we optimize a model (typically an approximation of the base model) over some horizon. The simplest and most popular form of direct lookahead is to solve a deterministic approximation of the future, as we did with the dynamic shortest path problem.

It would be ideal if we could solve the true problem in the future, starting from the state S_{t+1} produced by starting in state S_t, taking action x_t, and then observing the random information W_{t+1}. Although it is quite clumsy, this would

be written as

$$X^{DLA}(S_t) = \arg\min_{x_t \in \mathcal{X}} \Bigg(C(S_t, x_t) + \mathbb{E}_{W_{t+1}} \Bigg\{ \min_{\hat{\pi}} \mathbb{E}_{W_{t+2},\ldots,W_T} \Bigg\{ \sum_{t'=t+1}^{T} C(S_{t'}, X^{\hat{\pi}}(S_{t'})) | S_{t+1} \Bigg\} | S_t, x_t \Bigg\} \Bigg). \tag{7.1}$$

If we could compute equation (7.1), we would have an optimal policy. It is quite rare that equation (7.1) can be solved exactly, so we are again looking for approximations that strike a balance between solution quality and computational efficiency.

These four classes of policies (PFAs, CFAs, VFAs and DLAs) are universal, which is to say, any policy chosen for a sequential decision problem (*any* sequential decision problem) will be one of these four classes, or a hybrid. In all the remaining applications, we will end up choosing from one (or a combination) of these policies.

7.2 Online vs. offline objectives

There are two perspectives for evaluating the performance of a policy:

Online learning There are many settings where we have to learn as we go in the field. For example, we might be trying to learn the best price for a product, the best path through a congested city, or the best drug for a patient to lower blood pressure. In all these cases, we want to do as well as we can given what we know, but we are still learning so we can make better decisions in the future. This means we need to maximize the *cumulative reward* over time (or iterations).

Offline learning In other cases, we have a setting where we can learn in a laboratory setting, which might be a physical lab (to test different materials or observe the performance of a drug on mice), a computer simulator, or even a field setting which is being run

as a test market (to evaluate a product) or a clinical trial (to test drugs).

A word of caution about the terms online and offline. In the machine learning community, "offline" refers to the estimation of models using a single, batch dataset. By contrast, online learning is used to refer to fully sequential settings where data arrives over time. This is typically in field situations where data is created by some exogenous process (such as observing patients arriving to a doctor's office), which is the same setting we assume when using the term "online." However, in machine learning "online" would still be used by an iterative algorithm being used in a simulation.

We illustrate each of these below, and then show that from the perspective of the design of algorithms for policy search, they both converge to the same problem.

7.2.1 Online (cumulative reward) optimization

It is typically the case when doing policy search that we have a parameterized policy that we can write as $X^\pi(S_t|\theta)$. The decision $x_t = X^\pi(S_t|\theta)$ might be the price of a product, the choice of a blood-pressure drug or the bid placed to maximize ad-clicks. In all these cases, we have to learn as we go, which means we need to maximize performance as we are learning.

Let $C(S_t, x_t)$ be our performance metric (revenue, reduction in blood-pressure, or net revenue from ad-clicks), where we want to find θ that solves

$$\max_\theta F(\theta) \quad = \quad \mathbb{E}_{W_1,\ldots,W_T} \sum_{t=0}^{T} C(S_t, X^\pi(S_t|\theta)), \tag{7.2}$$

where $S_{t+1} = S^M(S_t, X^\pi(S_t), W_{t+1})$. The expectation in (7.2) is over all possible realizations of $W_1, \ldots, W_T$.

Equation (7.2) is an example of an "online" or "cumulative reward" objective function, since we want to maximize the sum of all the rewards over some horizon. This is of particular interest in online learning problems where we have to learn the performance, such as revenue from a price or the performance of a drug for a particular patient, which

means balancing the process of learning while also trying to do as well as we can.

7.2.2 Offline (final reward) optimization

In offline settings, we typically have a budget of N experiments. A classical (if inappropriate) problem that is often used to illustrate offline, derivative-free learning is the newsvendor problem which we can write as

$$F(x) = \mathbb{E}_W(p \min\{x, W\} - cx), \tag{7.3}$$

where x is the number of "newspapers" (or any product) that is placed in the morning for sale. The unit cost of each newspaper is c, and we sell each newspaper up to an unknown demand W at a price p. We do not know the distribution of W, so we cannot compute the expectation. Instead, we are going to depend on iterative learning procedures that allow us to set $x = x^n$ and then observe W^{n+1}.

Let $x^n = X^\pi(S^n|\theta)$ be our choice of x given what we know which is captured by S^n, after which we observe W^{n+1} that allows us to compute a performance metric $F(x^n, W^{n+1})$. Finally, let $x^{\pi,N}(\theta)$ be the final choice of x produced by our policy $X^\pi(S|\theta)$ after our budget of N experiments is exhausted. To evaluate how well this design works, we have to test it over many possible outcomes of W. To distinguish the outcomes of W observed during training iterations from outcomes of W when testing the performance of the final design, we denote the observations of W during testing by $W^1, \ldots, W^N$, after which we let $\widehat{W}$ be the outcomes observed for testing.

This notation allows us to write our objective function for offline learning as

$$\max_\theta F(\theta) = \mathbb{E}_{W^1,\ldots,W^N}\mathbb{E}_{\widehat{W}}F(x^{\pi,N}(\theta), \widehat{W}). \tag{7.4}$$

Keep in mind that our final decision $x^{\pi,N}(\theta)$ depends on the sequence $W^1, \ldots, W^N$, as well as our policy $X^\pi(S^n|\theta)$.

7.2.3 Bringing them together

Equation (7.2) illustrates an online, or cumulative reward, objective function where we have to maximize total performance during the learning process. Equation (7.4) illustrates the offline, or final reward, objective function where we have to search for the best design that will work best on average after we fix the design. What is important at the moment is that both problems involve solving

$$\max_{\theta} F(\theta), \tag{7.5}$$

where $F(\theta)$ is an unknown function which we can sample in a noisy way.

At this point, our search for the best θ (in this setting, this means the best policy) reduces to a single optimization problem. The design of algorithms tends to be centered around whether we have access to derivatives $\nabla_\theta F(\theta)$ or not. We give a brief introduction below to both derivative-based and derivative-free optimization.

7.3 Derivative-based policy search

Assume that we are trying to solve the problem

$$\max_{\theta} F(\theta), \tag{7.6}$$

where $F(\theta)$ is some parametric function in θ. Further assume that θ is a vector and that we can compute the gradient

$$\nabla_\theta F(\theta) = \begin{pmatrix} \frac{\partial F(\theta)}{\partial \theta_1} \\ \frac{\partial F(\theta)}{\partial \theta_2} \\ \vdots \\ \frac{\partial F(\theta)}{\partial \theta_K} \end{pmatrix}.$$

A basic gradient algorithm would consist of

$$\theta^{n+1} = \theta^n + \alpha_n \nabla_\theta F(\theta^n), \tag{7.7}$$

where α_n is a stepsize computed using

$$\alpha_n = \arg\max_{\alpha \geq 0} F(\theta^n + \alpha \nabla_\theta F(\theta^n)). \tag{7.8}$$

The difficulty with this algorithmic strategy for stochastic problems is that we generally cannot compute the expectation in the objective function, whether it be equation (7.2) or (7.4). This creates problems with computing the gradient, but also with computing the stepsize using equation (7.8).

There is an alternative strategy that is simple and practical. The idea is to fix θ, and then to observe a sample of all random variables that we will refer to generically as W. For the online objective in equation (7.2), W would be the entire sequence $W_1, \ldots, W_T$. For the offline objective in equation (7.4), W represents both the learning sequence $W_1, \ldots, W_T$ and the testing variable $\widehat{W}$. Using this generic view of W, we can write both objectives (7.2) or (7.4) using the generic formulation

$$\max_{\theta} \mathbb{E}_W F(\theta, W). \tag{7.9}$$

Next assume that while we cannot compute the expectation, we can compute the derivative of $F(\theta, W)$ after W is known, which means that we can compute $\nabla_\theta F(\theta^n, W^{n+1})$. Finally, we replace the deterministic updating formula in (7.7) with a stochastic gradient algorithm that is written

$$\theta^{n+1} = \theta^n + \alpha_n \nabla_\theta F(\theta^n, W^{n+1}). \tag{7.10}$$

This looks easy, except that we next have to figure out how to find the stepsize α_n, because we will never be able to do the one-dimensional search. The good news is, we do not need to. In fact, if we use the basic stepsize rule

$$\alpha_n = \frac{\theta^{step}}{N^n + \theta^{step} - 1}, \tag{7.11}$$

where N^n is the number of times that the objective has improved. The parameter θ^{step} guides the rate at which the stepsize decreases, and is a parameter that has to be tuned. It is possible to show that

$$\lim_{n \to \infty} \theta^n \to \theta^*,$$

where θ^* is the optimal solution to (7.6).

The sheer elegance and simplicity of this algorithm helps to explain its popularity. However, anyone who has ever tried it struggles with

stepsizes. The formula in (7.11) is known as a harmonic stepsize rule, and this can work quite well if θ^{step} is properly tuned. There is by now a rich literature on stepsize rules (see chapter 6 of *Stochastic Optimization and Learning*).

It is useful to point out that the stochastic gradient algorithm, which we are using to find θ to find the best learning policy for a sequential decision problem, is its own sequential decision problem, as we first pointed out in chapter 3. We can model the five elements of the problem as:

- State variable - This would be $S^n = (x^n, N^n)$.
- Decision - The decision in this process is the stepsize α_n.
- Exogenous information - This would be W^{n+1}.
- Transition function - This is just the stochastic gradient algorithm

$$\begin{aligned} \theta^{n+1} &= \theta^n + \alpha_n \nabla_\theta F(\theta^n, W^{n+1}), \\ N^{n+1} &= \begin{cases} N^n + 1 & \text{If } F(\theta^n, W^{n+1}) > F(\theta^{n-1}, W^n) \\ N^n & \text{Otherwise.} \end{cases} \end{aligned}$$

- Objective function - Let $\theta^{\pi,N}(\theta^{step})$ be the final solution of our stochastic gradient algorithm. We wish to find θ^{step} that solves

$$\max_{\theta^{step}} \mathbb{E}_{W^1,\ldots,W^N} \mathbb{E}_{\widehat{W}} F(\theta^{\pi,N}(\theta^{step}), \widehat{W}). \tag{7.12}$$

Thus, we have to tune the search algorithm (by finding θ^{step}) to optimize the policy to solve our original sequential decision problem, which might be either a policy function approximation (PFA) or a parametric cost function approximation (CFA), and which might use either the online (cumulative reward) or offline (final reward) objective functions.

7.4 Derivative-free policy search

Derivative-free policy search is nothing more than a sequential decision problem that is the focus of this entire volume, with the main difference that the only state variable will be the belief about the function we are maximizing. We can form beliefs using any of:

Lookup tables Assume that we can discretize the set of possible values of θ into a set $\Theta = \{\theta_1, \ldots, \theta_K\}$. Define a random variable $\mu_\theta = F(\theta) = \mathbb{E}F(\theta, W)$ which is a random variable because we do not know $F(\theta)$ (or μ_θ). Assume that we can run experiments to sample $\hat{F}^{n+1} = F(\theta^n, W^{n+1})$. We can use these samples to create estimates $\bar{\mu}_\theta^n$ for each $\theta \in \Theta$. This would be a lookup table belief model.

Parametric model We might estimate a linear model of the form

$$F(\theta|\rho) \quad \approx \quad \rho_0 + \rho_1\phi_1(\theta) + \rho_2\phi_2(\theta) + \ldots$$

where $\phi_f(\theta)$ are features drawn from the vector θ, which might consist of terms such as θ, θ^2, or $\ln\theta$. While linear models are popular (this means linear in the parameter vector ρ - the feature vector $\phi_f(\theta)$ can create any transformation of the input variable θ.

Nonparametric models There are many ways to create nonparametric models. Perhaps the most common would be to create an estimate $\bar{\mu}_\theta$ by averaging over $\bar{\mu}_{\theta_1}, \ldots, \bar{\mu}_{\theta_K}$ for values θ_k that are close to θ. We are not going to draw on nonparametric methods in this book, primarily because they are data intensive and somewhat clumsy to use.

Chapter 3 of *Stochastic Optimization and Learning* describes a number of methods for recursively estimating functions. We already saw the recursive equations for a lookup table for our diabetes example, where the updating was given by equations (4.1)-(4.2). Later we will illustrate the recursive updating of linear and nonlinear models.

It is possible to approach the learning of θ in our policy search using any of the four classes of policies we have illustrated above. The choice of the best policy for a pure learning problem will not be the same as for problems where we have other complicating elements such as a physical state.

8

Energy storage I

8.1 Narrative

New Jersey is looking to develop 3,500 megawatts (MW) of off-shore wind generating power. A challenge is that wind (and especially offshore wind) can be highly variable, due in part to the property that wind power (over intermediate ranges) increases with the cube of wind speed. This variability is depicted in figure 8.1, developed from a study of off-shore wind.

Energy from wind power has become popular in regions where wind is high, such as the midwestern United States, coastal regions off of Europe, the northeast of Brazil, and northern regions of China (to name just a few). Sometimes communities (and companies) have invested in renewables (wind or solar) to help reduce their carbon footprint and minimize their dependence on the grid.

It is quite rare, however, that these projects allow a community to eliminate the grid from their portfolio. Common practice is to let the renewable source (wind or solar) sell directly to the grid, while a company may purchase from the grid. This can be useful as a hedge since the company will make a lot of money during price spikes (prices may jump from $20 per megawatt-hour (mwh) to $300 per mwh or

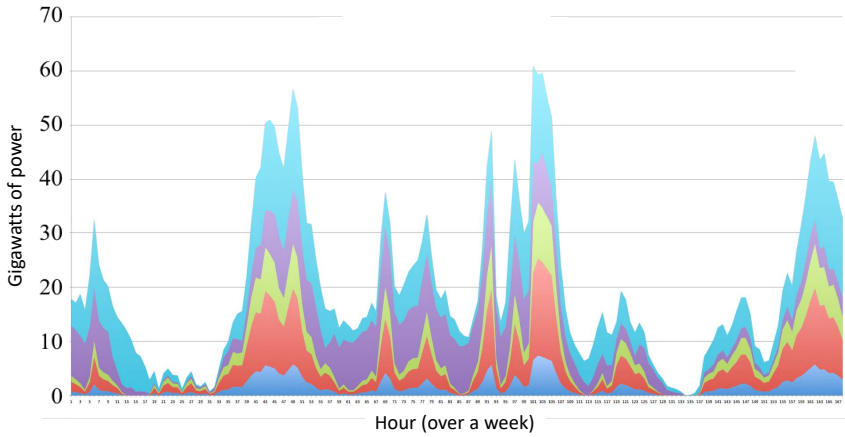


Figure 8.1: Power from five levels of wind generating capacity.

more) that offsets the cost of purchasing power during those periods.

A challenge with renewables is handling the variability. While one solution is to simply pump any energy from a renewable source into the grid and use the tremendous power of the grid to handle this variability, there has been considerable interest in using storage (in particular, battery storage) to smooth out the peaks and valleys. In addition to handling the variability in the renewable source, there has also been interest in using batteries to take advantage of price spikes, buying power when it is cheap (prices can even go negative) and selling it back when they are high. Exploiting the variability in power prices on the grid is known as battery arbitrage.

We are going to use the configuration shown in figure 8.2 to illustrate a number of modeling and algorithmic issues in energy storage. This problem will provide insights into virtually any inventory/storage problem, including

- Holding cash in mutual funds - A bank has to determine how much of its investment capital to hold in cash to meet requests for redemptions, versus investing the money in loans, stocks and bonds.

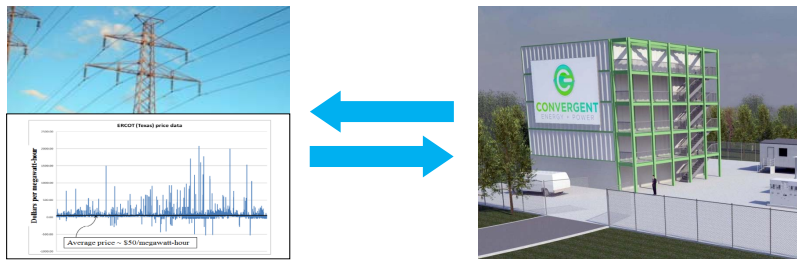


Figure 8.2: Grid to storage system for power stabilization and battery arbitrage.

- Retailers (including both online as well as bricks-and-mortar stores) have to manage inventories of the hundreds of thousands of products.
- Auto dealers have to decide how many cars to hold to meet customer demand.
- Consulting firms have to decide how many employees to keep on staff to meet the variable demand of different consulting projects.

Energy storage is a particularly rich form of inventory problem. While we are not going to consider all possible variations (which are endless), our problem will exhibit the following characteristics:

- Electricity prices on the grid can be highly volatile. As of this writing, typical energy prices run around $20-$25 per mwh, but often spike to over $300, and can exceed $1000, typically during extreme weather events.
- Energy from wind can be forecasted, although not very well. Rolling forecasts are available to update these estimates.
- Solar energy exhibits three types of variability: the highly predictable process of the diurnal cycle of the sun rising and setting, the presence of very sunny or very cloudy days (these can typically be predicted a day or more in advance), and the variability of spot clouds that are difficult to predict even an hour into the future, but which can create severe power surges on the grid.

- The demand for energy is variable, but relatively predictable since it primarily depends on temperature (and, to a lesser extent, humidity).
- Energy may be bought from or sold to the grid at current grid prices. Similarly energy from the renewables source may be used to satisfy the current load (demand for power), stored, or sold back to the grid (depending on the configuration).
- There is a roughly 5 to 10 percent loss for the conversion of power from AC (as it arrives over the grid) to DC (required to store power in the battery).

8.2 Basic model

We will use a sequence of variations of this problem to illustrate different modeling issues, beginning with a basic system of using a battery to buy from and sell to the grid to take advantage of price volatility. For this application, we will step forward in 5-minute time increments, since this is the frequency with which prices are updated on the grid (this time increment varies depending on the grid operator).

8.2.1 State variables

For our basic model, we only need to keep track of two variables:

$$\begin{aligned} R_t &= \text{The amount of energy (measured in megawatt-hours, or mwh) stored in the battery at time } t, \\ p_t &= \text{The price of energy on the grid.} \end{aligned}$$

Our state variable is then

$$S_t = (R_t, p_t).$$

The state variable quickly becomes more complex as we add different elements to the model. For example, the state variable to represent prices depends on how we model the price process, as described in the transition function (see below).

8.2.2 Decision variables

Our only decision is whether to buy from or sell to the grid:

$$x_t = \text{The amount of power bought from or sold to the grid.}$$

($x_t > 0$ for buy, $x_t < 0$ for sell)

When we transfer energy into or out of the battery, we are going to assume that we only get a fraction η in the transfer, implying a loss of $1 - \eta$. For simplicity we are going to assume that this loss is the same regardless of whether we are charging or discharging the battery.

The decision is limited by the capacity of the battery, which means we have to observe the constraints

$$\begin{aligned} x_t &\leq \frac{1}{\eta}(R^{max} - R_t), \\ x_t &\geq -R_t, \end{aligned}$$

where the first constraint applies when we are buying from the grid ($x_t > 0$) while the second constraint applies when we are selling to the grid ($x_t < 0$).

As always, we assume that decisions are made with a policy $X^\pi(S_t)$, to be determined below.

8.2.3 Exogenous information

In our basic model, the only exogenous information is the change in the prices. We may assume that the price in each time period is revealed, without any model to predict the price based on past prices. In this case, our exogenous information W_t would be

$$W_{t+1} = p_{t+1}.$$

Alternatively, we can assume that we observe the change in price $\hat{p}_t = p_t - p_{t-1}$, in which case we would write

$$W_{t+1} = \hat{p}_{t+1}.$$

8.2.4 Transition function

The evolution of the state variables are given by

$$R_{t+1} = R_t + \eta x_t, \tag{8.1}$$

$$p_{t+1} = p_t + \hat{p}_{t+1}. \tag{8.2}$$

This style of modeling the price process by "observing" the change in price helps us when writing the transition function. In practice, we would typically be observing p_{t+1} directly (rather than the change), in which case there is no need for an explicit transition equation. These two equations make up our transition function $S_{t+1} = S^M(S_t, x_t, W_{t+1})$.

Later, we are going to find it useful to model the post-decision state S_t^x, which is the state just after we make a decision x_t, but before any new information arrives. The post-decision resource state is

$$R_t^x = R_t + \eta x_t.$$

Because the storage variable evolves deterministically, the transition to the next pre-decision state is just

$$R_{t+1} = R_t^x.$$

The price p_t, on the other hand, is not affected by the decision, so the post-decision price would just be

$$p_t^x = p_t.$$

This means the post-decision state is

$$S_t^x = (R_t^x, p_t).$$

8.2.5 Objective function

In any period, the amount of money we make or lose is given by

$$C(S_t, x_t) \quad = \quad -p_t x_t.$$

Our objective function is then the canonical problem given by

$$\max_{\pi} \mathbb{E} \sum_{t=0}^{T} -p_t X^{\pi}(S_t),$$

where $S_{t+1} = S^M(S_t, X^{\pi}(S_t))$ is given by equations (8.1) and (8.2). We then also need to specify the initial state S_0 (that is, R_0 and p_0) and have a method for generating $W_1, W_2, \ldots,$ which we describe next.

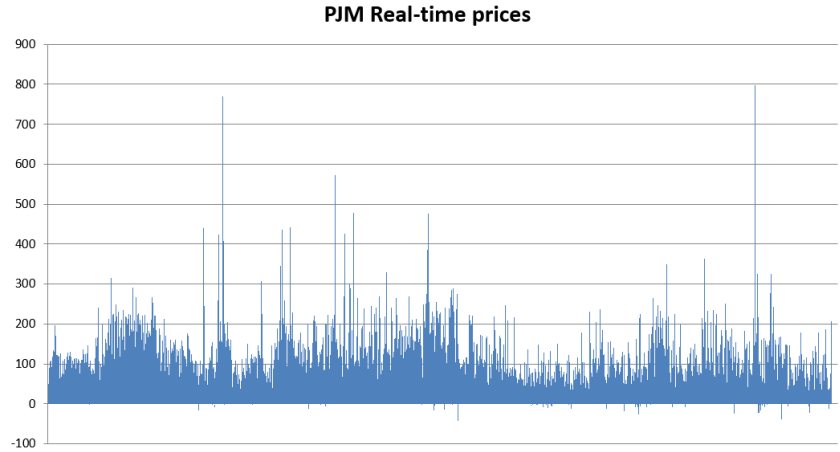


Figure 8.3: Locational marginal prices for the PJM grid for 2010.

8.3 Modeling uncertainty

In our asset selling problem, we assumed that we could model prices according to

$$p_{t+1} = p_t + \varepsilon_{t+1},$$

where we then assumed that ε_{t+1} was normally distributed with mean 0 and some known variance. Figure 8.3 shows grid prices, known as "locational marginal prices" (or LMPs in the terminology of the power community) over a year, which illustrates the tremendous volatility that prices on the grid exhibit. This volatility arises because there are short term surges in load (or losses in power) that can create short term shortages. Since demand is inelastic (the grid is supposed to meet 100 percent of the load), prices can jump by a factor of 20 to 50 for short periods (prices are updated in 5-minute increments). There are several methods for modeling electricity prices. Below we are going to describe four that have been used for this problem.

8.3.1 Time series models

The time series literature is quite rich, so we are just going to illustrate a basic model which represents the price p_{t+1} as a function of the recent history of prices. For illustration, we are going to use the last three time periods, which means that we would write our model as

$$\begin{aligned} p_{t+1} &= \bar{\theta}_{t0} p_t + \bar{\theta}_{t1} p_{t-1} + \bar{\theta}_{t2} p_{t-2} + \varepsilon_{t+1}, \\ &= \bar{\theta}_t^T \phi_t + \varepsilon_{t+1}, \end{aligned} \tag{8.3}$$

where

$$\phi_t = \begin{pmatrix} p_t \\ p_{t-1} \\ p_{t-2} \end{pmatrix}$$

is our vector of prices. We assume that the noise $\varepsilon \sim N(0, \sigma_\epsilon^2)$ for a given σ_ϵ^2.

The vector of coefficients $\bar{\theta}_t = (\bar{\theta}_{t0}, \bar{\theta}_{t1}, \bar{\theta}_{t2})^T$ can be estimated recursively using the methods presented in chapter 3 of *Stochastic Optimization and Learning*. Assume that we start with an initial estimate $\bar{\theta}_0$ of the vector of coefficients. We are also going to need a three-by-three matrix B^0 that for now we can assume is a scaled identity matrix (we provide a better idea below).

The basic updating equation for $\bar{\theta}_t$ is given by

$$\bar{\theta}_{t+1} = \bar{\theta}_t - H_t \phi_t \varepsilon_{t+1}, \tag{8.4}$$

The error $\hat{\varepsilon}_t$ is computed using

$$\varepsilon_{t+1} = \bar{\theta}_t^T \phi_t - p_{t+1}. \tag{8.5}$$

The three-by-three matrix H_t is computed using

$$H_t = \frac{1}{\gamma_t} B_t, \tag{8.6}$$

where the matrix B_n is computed recursively using

$$B_t = B_{t-1} - \frac{1}{\gamma_t}(B_{t-1}\phi_t(\phi_t)^T B_{t-1}). \tag{8.7}$$

The variable γ_t is a scalar computed using

$$\gamma_t = 1 + (\phi_t)^T B_{t-1}\phi_t. \tag{8.8}$$

These equations need initial estimates for $\bar{\theta}_0$ and B_0. One way to do this is to collect some initial data and then solve a static estimation problem. Assume you observe K prices. Let Y_0 be a K-element column factor of the observed prices $p_3, p_4, \ldots, p_{K+3-1}$ (we have to start with the third price because of our need for the trailing three prices in our model).

Then let X_0 be a matrix with K rows, where each row consists of p_k, p_{k-1}, p_{k-2}. Our best estimate of $\bar{\theta}$ is given by the normal equations

$$\bar{\theta}_0 = [(X_0)^T X_0]^{-1}(X_0)^T Y_0. \tag{8.9}$$

Finally let $B_0 = [(X_0)^T X_0]^{-1}$, which shows that the matrix B_t is the time t estimate of $[(X_t)^T X_t]^{-1}$.

There are entire families of time series models that capture the relationship of variables over time. If we were to just apply these methods directly to price data, the results would be quite poor. First, the prices are not normally distributed. Second, the prices cannot be negative, but a direct application of this model would be very likely to produce negative prices if the variance σ_ϵ^2 was calibrated to the high-noise of this type of data. Finally, the behavior of the jumps in prices over time would not be realistic.

8.3.2 Jump diffusion

A major criticism of the linear model above is that it does a poor job of capturing the large spikes that are familiar in the study of electricity prices. A simple idea for overcoming this limitation is to use what is known as a *jump diffusion model*, where we add another noise term to equation (8.3) giving us

$$p_{t+1} = \bar{\theta}_{t0}p_t + \bar{\theta}_{t1}p_{t-1} + \bar{\theta}_{t2}p_{t-2} + \varepsilon_{t+1} + \mathbb{I}_t\varepsilon_{t+1}^J. \tag{8.10}$$

Here, the indicator variable $\mathbb{I}_t = 1$ with some probability p^{jump}, and the noise ε_{t+1}^J is normally distributed with mean μ^{jump} (which is typically much larger than zero) and variance $(\sigma^{jump})^2$ which is quite large.

We have to estimate the jump probability p^{jump}, and the mean and variance
$(\mu^{jump}, (\sigma^{jump})^2)$. This is done by starting with a basic model where $p^{jump} = 0$. We use this basic model to estimate σ_ϵ^2. We then choose some tolerance such as three standard deviations (that is, $3\sigma_\epsilon$), and assume any observations outside of this range should be attributed to a different source of noise. Let p^{jump} be the fraction of time periods where these observations occur. Then, compute the mean and standard deviation of these observations to get $(\mu^{jump}, (\sigma^{jump})^2)$.

We do not stop here. After taking these extreme variations from the data, we should re-fit our linear model without these observations. Standard practice is to repeat this process several times until these estimates stop changing.

Jump diffusion models do a better job of replicating the tails, but it still depends on the tail behavior of the normal distribution. A better fit can be obtained by recognizing that the variance of the noise depends on the temperature, and particularly on extreme temperatures. We might group temperatures into three ranges: below freezing, above 90 degrees F, and in-between these two values. Introducing temperature dependence, while adding another variable to the set of state variables, does introduce an additional degree of complexity (the effect of this depends on the class of policy),

8.3.3 Empirical distribution

While it may be possible to fit other parametric distributions, a powerful strategy is to numerically compute the cumulative distribution from the data, creating what is often called an *empirical distribution*. To compute this, we simply sort the prices from smallest to largest. Denote this ordered sequence by $\tilde{p}_t$, where $\tilde{p}_{t-1} \leq \tilde{p}_t$. Let $T = 105,210$ which is the number of 5-minute time periods in a year. The percentage of time periods with a price less than $\tilde{p}_t$ is then t/T. We can create a cumulative distribution using

$$F_P(\tilde{p}_t) = \frac{t}{T},$$

which is illustrated in figure 8.4.

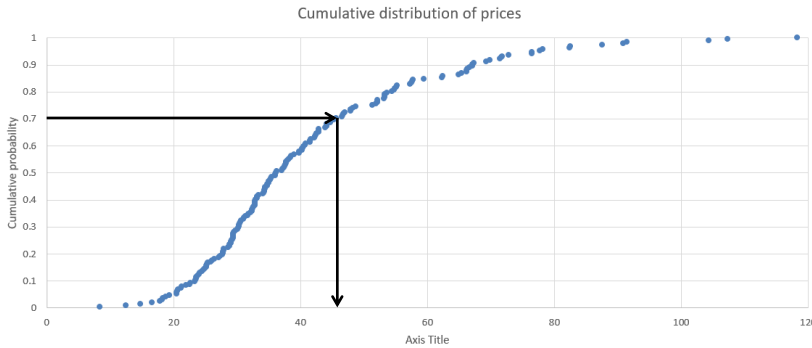


Figure 8.4: Empirical distribution of prices.

We can create a continuous distribution $F_P(p)$ for any p by finding the largest $\tilde{p}_t < p$ and setting $F_P(p)$ equal to this value, creating a step function. The function $F_P(p)$ is sometimes called a quantile distribution, and is a form of *nonparametric* distribution, since we are not fitting the distribution to any known parametric form. The good news is that it will perfectly match the data, which means that we will accurately represent the extreme tails that occur with electricity prices. The downside is that we require a good dataset to create these distributions, and we have to retain the dataset to compute the distribution, rather than just storing a small number of parameters as we would if we fit a parametric model for the distribution.

We can sample from this distribution by generating a random variable U that is uniformly distributed between 0 and 1. Let's say we generate $U = 0.70$. Then we want to find the price $p^{\cdot 70}$ that corresponds to $F_P(p^{\cdot 70}) = 0.70$, as illustrated in figure 8.4. We write this mathematically by defining the inverse function $F_P^{-1}(u)$ which returns the price p that produces $F_P(p) = u$. We can sample repeatedly from our price distribution by just sampling the uniform random variable U and then observing a price $p = F_P^{-1}(U)$.

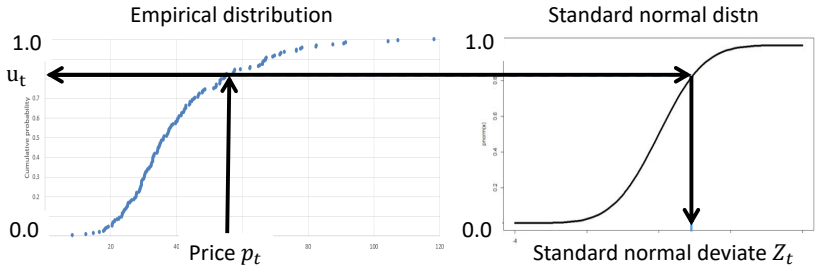


Figure 8.5: Empirical distribution of prices.

8.3.4 Hybrid time-series with transformed data

A powerful strategy is to combine the use of empirical distributions with classical time series methods. We start by fitting an empirical distribution to the price data, giving us the cumulative distribution $F_P(p)$. Now, let p_t be a price and compute $u_t = F_P(p_t)$, where $0 \leq u_t \leq 1$. Next, let $\Phi(z)$ be the cumulative distribution of a standard normal random variable $Z \sim N(0, 1)$, and let $\Phi^{-1}(u)$ be its inverse. Next let $z_t = \Phi^{-1}(u_t)$. The process of mapping $p_t \rightarrow u_t \rightarrow z_t$ is illustrated in figure 8.5.

We can use this method to transform the highly non-normal prices p_t to the sequence z_t of values that are normally distributed with mean 0 and variance 1. We can then perform any time series modeling on the sequence z_t. After this, any estimates coming from this normalized model can be transformed back to prices by tracing the path in figure 8.5 in the reverse direction: $u_t = \Phi(z_t)$, and then $p_t = F_P^{-1}(u_t)$.

This strategy is very effective when dealing with data that is not normally distributed, and performs much better than the jump diffusion model, which is popular in finance.

8.4 Designing policies

We are going to illustrate solving this problem using two classes of policies, plus a hybrid:

Policy search - We are going to use a simple buy low, sell high param-

eterized policy. This belongs to the PFA class of policies (policy function approximations).

Lookahead policy - We will use Bellman's equation to produce a value function approximation that approximates the impact of a decision now on the future. This belongs to the VFA class of policies (policies based on value function approximations).

Hybrid policy - Finally, we are going to introduce a tunable class of policy that begins with a policy based on value functions, and then switches to policy search to further tune the policy. This will start as a VFA policy, but then transition to a parametric cost function approximation (CFA) when we use policy search to tune the parameters of what started as the value function approximation.

Policies based on Bellman's equation require computing (or approximating) the value $V_{t+1}(S_{t+1})$ which results from being in a state S_t, taking a decision x_t, and then observing random exogenous information W_{t+1}. We first saw these methods in the context of shortest path problems. One significant difference now is that the state S_{t+1} is random given S_t and x_t (in the shortest path problem, only the cost $\hat{c}_t$ was random). Also, our state variable now has two continuous dimensions, rather than just the discrete node.

We first describe the buy low, sell high policy, and then introduce three methods for solving Bellman's equation:

- Classical backward dynamic programming, which produces an optimal policy.
- Backward approximate dynamic programming.
- Forward approximate dynamic programming.

We finish with a description of a hybrid policy that combines value function approximations from Bellman's equation with a form of policy search.

8.4.1 Buy low, sell high

A buy low, sell high policy works on the simple principle of charging the battery when the price falls below a lower limit, and then sells when the price goes above an upper limit. The policy can be written as

$$X^{low-high}(S_t|\theta) = \begin{cases} -1 & \text{If } p_t \leq \theta^{buy} \\ 0 & \text{If } \theta^{buy} < p_t < \theta^{sell} \\ +1 & \text{If } p_t \geq \theta^{sell}. \end{cases} \tag{8.11}$$

We now have to tune $\theta = (\theta^{buy}, \theta^{sell})$. We evaluate our policy by following a sample path of prices $p_t(\omega)$ (or we may be observing the changes in prices $\hat{p}(\omega)$). Assuming we are generating sample paths from a mathematical model, we can generate sample paths $\omega^1, \ldots, \omega^N$. We can then simulate the performance of the policy over each sample path and take an average using

$$\overline{F}^{low-high} = \frac{1}{N}\sum_{n=1}^{N} C\big(S_t(\omega^n), X^{low-high}(S_t(\omega^n)|\theta)\big).$$

Tuning θ requires solving the problem

$$\max_{\theta} \overline{F}^{\pi}(\theta). \tag{8.12}$$

Since θ has only two dimensions, one strategy is to do a full grid search by discretizing each dimension, and then searching over all possible values of the two dimensions. A common discretization is to divide a region into 5-percent increments. Including the boundaries, this means we have to represent 21 values of each parameter, creating a grid of size 441 points, which is manageable (although not trivial) for most problems.

We note that a brute-force grid search only works if we run enough simulations N so that the variance in the estimate $\overline{F}^{\pi}(\theta)$ is relatively small. However, we have methods for doing the search for θ even with noisy estimates of the performance of the policy. In fact, this is just another instance of a sequential decision problem. For example, rather than using a smoothed estimate $\overline{F}^{\pi}(\theta)$, which requires running repeated simulations, we can use a single sample $\hat{F}^{\pi}(\theta)$, and apply the same search methods that we are trying to develop to solve the decision

problems covered in this volume. For example, if we are searching over a discrete set of values of $\theta \in \Theta = \{\theta_1, \ldots, \theta_K\}$ (for example, the 441 points in the grid), we can apply the same methods we introduced for the diabetes problem in chapter 4.

8.4.2 Backward dynamic programming

Backward dynamic programming involves directly solving Bellman's equation

$$V_t(s_t) = \max_{x_t} \left(C_t(s, x_t) + \mathbb{E}\{V_{t+1}(S_{t+1})|S_t, x_t\}\right), \quad (8.13)$$

where $S_{t+1} = S^M(s_t, x_t, W_{t+1})$, and where the expectation is over the random variable W_{t+1}. Assume that W_{t+1} is discrete, taking values in $\mathcal{W} = \{w_1, w_2, \ldots, W_M\}$, and represent the probability distribution using

$$f^W(w|s_t, x_t) = Prob[W_{t+1} = w|s_t, x_t].$$

We write the distribution as depending on the state s_t and decision x_t, but this depends on the problem. For example, we might reasonably assume that the change in the price $p_{t+1} - p_t$ depends on the current price p_t (if prices are very high they are more likely to drop), which would be a reason to condition on s_t. We might even need the dependence on x_t if purchasing a large quantity of electricity from the grid drives up prices.

We can then rewrite equation (8.13) as

$$V_t(s_t) = \max_{x_t} \left(C_t(s_t, x_t) + \sum_{w \in \mathcal{W}} f^W(w|s_t, x_t) V_{t+1}(S^M(s_t, x_t, w))\right).$$

A vanilla implementation of backward dynamic programming is given in figure 8.6 which exhibits four loops:

1) The loop stepping backward in time from T to time 0.

2) The loop over all possible states $s_t \in \mathcal{S}$ (more precisely, this is the set of possible values of the state S_t at time t).

3) The loop that would be required to search over all possible decisions x_t in order to solve the maximization problem.

Step 0. Initialization:

Initialize the terminal contribution $V_{T+1}(S_{T+1}) = 0$ for all states S_{t+1}.

Step 1. Do for $t = T, T-1, \ldots, 1, 0$:

Step 2. For all $s \in \mathcal{S}$, compute

$$V_t(s_t) = \max_{x_t} \left(C_t(s_t, x_t) + \sum_{w \in \mathcal{W}} f^W(w|s_t, x_t) V_{t+1}(S^M(s_t, x_t, w)) \right)$$

Figure 8.6: A backward dynamic programming algorithm.

4) The loop over all possible values of the random variable W that is captured in the summation required to compute $V_t(s)$.

It is useful to consider the range of values that each loop might take. For an energy problem, we might optimize a storage device in hourly increments over a day, giving us 24 time steps. If we use 5-minute time steps (some grid operators update prices every 5-minutes), then a 24 hour horizon would imply 288 time periods (multiply by seven if we want to plan over a week). If we are doing frequency regulation, then we have to make decisions every 2 seconds, which translates to 43,200 time periods over a day.

Our state variable consists of $S_t = (R_t, p_t)$, which means we have to replace the loop over all states, with nested loops over all values of R_t, and then all values of p_t. Since both are continuous, each will have to be discretized. The resource variable R_t would have to be divided into increments based on how much we might charge or discharge in a single time increment. We then have to discretize the grid price p_t. Grid prices can go as low as -$100, and as high as $10,000 (in extreme cases). A reasonable strategy might be to construct an empirical distribution, and then represent prices corresponding to, say, each increment of two percent of the cumulative distribution, giving us 50 possible prices.

The number of charge-discharge decisions might be as small as three (charge, discharge or do nothing), or much larger if we can charge or discharge at different rates.

Finally, the probability distribution $f^W(w)$ would be the distri-

bution of the random changes in prices, $\hat{p}_{t+1}$. Again, we recommend constructing an empirical distribution of the changes in $\hat{p}_{t+1}$ and then discretizing the cumulative distribution into increments of, say, two percent.

If we have a two-dimensional state variable (as is the case with our basic model), then we already have five loops (time, the two state variables, the max operator over x, and then the summation over outcomes of W). This can get expensive, and we have just started. Now imagine that we are using the time series model in equation (8.3), where we now have to keep track of the prices (p_t, p_{t-1}, p_{t-2}). In this case, our state variable would be

$$S_t = (R_t, p_t, p_{t-1}, p_{t-2}).$$

In this case, we would now have seven nested loops. While the complexity depends on the discretization of the continuous variables, running the algorithm in figure 8.6 could easily require a year (or more).

8.4.3 Backward approximate dynamic programming

A powerful algorithmic strategy is known as "backward approximation dynamic programming." This approach progresses exactly as we just did in figure 8.6, with one difference. Instead of looping over all the states, we choose a random sample $\hat{S}$. We then compute the value of being in state $s \in \hat{S}$ just as we originally did, and compute the corresponding value $\hat{v}$. Assume we repeat this N times, and acquire a dataset $(\hat{s}^n, \hat{v}^n), n = 1, \ldots, N$. We then use this to fit a linear approximation of the form

$$\overline{V}(s) = \theta_0 + \theta_1\phi_1(s) + \theta_2\phi_2(s) + \ldots + \theta_F\phi_F(s), \qquad (8.14)$$

where $\phi_f(s), f = 1, \ldots, F$ is a set of appropriately chosen features. Examples of features might be

$$\begin{aligned}
\phi_1(s) &= R_t, \\
\phi_2(s) &= R_t^2, \\
\phi_3(s) &= p_t, \\
\phi_4(s) &= p_t^2, \\
\phi_5(s) &= p_{t-1}, \\
\phi_6(s) &= p_{t-2}, \\
\phi_7(s) &= R_t p_t.
\end{aligned}$$

Note that with a constant term θ_0, this model has only eight coefficients to be estimated. Sampling a few hundred states should be more than enough to get a good statistical approximation. This methodology is relatively insensitive to the number of state variables, and of course there is no problem if any of the variables are continuous.

We have found backward ADP to work exceptionally well on a small set of problems, but there are no guarantees, and its performance clearly depends on choosing an effective set of features. In one application, we reduced a run time of one month for a standard backward MDP algorithm, down to 20 minutes, with a solution that was within 5 percent of the optimal (produced by the one-month run time). However, there are no bounds or guarantees.

8.4.4 Forward approximate dynamic programming

Forward approximate dynamic programming works in an intuitive way. Imagine that we start with a value function approximation $\overline{V}_t^{x,n-1}(S_t^x)$ around the post-decision state S_t^x that we computed from the first $n-1$ iterations of our algorithm. We first introduced the idea of post-decision states back in section 1.3, but this is the state immediately after we make a decision, but before any new information has arrived.

Now, imagine that we are in a particular state S_t^n during the *nth* iteration of our algorithm, following a sample path ω^n that guides the sampling as we step forward in time. Now assume we have a function $S_t^{x,n} = S^{M,x}(S_t^n, x)$ that takes us to the post-decision state. For our

energy problem where $S_t^n = (R_t^n, p_t^n)$, the post-decision state would be

$$S^{x,n} = (R_t^n + x_t^n, p_t^n).$$

We then make a decision using

$$x_t^n = \arg\max_x \left(C(S_t^n, x) + \overline{V}_t^{x,n-1}(S^{x,n})\right).$$

Given S_t^n and our decision x_t^n, we then sample $W_{t+1}(\omega^n)$ which translates to the change in the prices $\hat{p}_{t+1}^n$. We then simulate our way to the next state

$$S_{t+1}^n = (R_t^n + x_t^n, p_t^n + \hat{p}_{t+1}^n(\omega)).$$

Thus, we are just simulating our way forward in time, which means we do not care how complex the state variable is. There are different strategies for then updating the value function approximation $\overline{V}_t^{n-1}$. One strategy is illustrated in figure 8.7, which leaves the actual update in an updating function $U^V(\cdot)$ since this depends on how we are approximating the value function.

Forward approximate dynamic programming is attractive because it scales to high dimensional problems. At no time are we looping over all possible states or outcomes. In fact, we can even handle high dimensional decisions x if we approximate the value function appropriately so that we can take advantage of powerful algorithms. However, forward ADP (as with backward ADP) possesses little in the way of performance guarantees.

8.4.5 A hybrid policy search-VFA policy

Whatever way we choose to approximate the value function, our policy is given by

$$X^{VFA}(S_t) = \arg\max_x \left(C(S_t, x) + \overline{V}_t^x(S_t^x)\right).$$

We simulate the policy forward, where we let ω^n represent a sample path of the exogenous information (that is, the set of changes in prices). It is frequently the case that we are going to test our policy on historical data, in which case there is only a single sample path. However, if we

have developed a mathematical model of the uncertain prices, we can sample a series of sample paths that we designate by $\omega^1, \ldots, \omega^N$, and compute the average value of the policy using

$$\overline{F}^{VFA} = \frac{1}{N}\sum_{n=1}^{N}\sum_{t=0}^{T} C(S_t(\omega^n), X^{VFA}(S_t(\omega^n))).$$

When we are building a value function approximation (which belongs in the lookahead class of policies), we typically no longer have a step where we tune the policy. However, this does not mean that we cannot try. Assume that our value function is given by the linear model in equation (8.14). We can now write our policy using

$$X^{VFA}(S_t|\theta) = \arg\max_x \left(C(S_t, x) + \sum_{f=1}^{F} \theta_f \phi_f(S_t) \right).$$

It makes sense to use one of our backward or forward ADP algorithms to get an initial estimate of θ, but as we noted above, there is no guarantee that the resulting policy is high quality. However, we can always make it better by using this as a starting point,

$$\overline{F}^{VFA}(\theta) = \frac{1}{N}\sum_{n=1}^{N}\sum_{t=0}^{T} C(S_t(\omega^n), X^{VFA}(S_t(\omega^n)|\theta)).$$

Now, we just have to solve the policy search problem that we might pose as

$$\max_\theta \overline{F}^{\pi}(\theta). \tag{8.17}$$

In our earlier examples for policy search, θ as a scalar, which makes this problem relatively easy. Now, θ is a vector which might have dozens of dimensions. We are going to return to this problem later.

Step 0. Initialization:

Step 0a. Initialize $V_t^{\pi,0}, \quad t \in \mathcal{T}$.

Step 0b. Set $n = 1$.

Step 0c. Initialize S_0^1.

Step 1. Do for $n = 1, 2, \ldots, N$:

Step 2. Do for $m = 1, 2, \ldots, M$:

Step 3. Choose a sample path ω^m.

Step 4: Initialize $\hat{v}^m = 0$

Step 5: Do for $t = 0, 1, \ldots, T$:

Step 5a. Solve:

$$x_t^{n,m} = \arg\max_{x_t \in \mathcal{X}_t^{n,m}} \left(C_t(S_t^{n,m}, x_t) + V_t^{\pi,n-1}(S^{M,x}(S_t^{n,m}, x_t)) \right) \tag{8.15}$$

Step 5b. Compute:

$$\begin{aligned} S_t^{x,n,m} &= S^{M,x}(S_t^{n,m}, x_t^{n,m}) \\ S_{t+1}^{n,m} &= S^M(S_t^{x,n,m}, x^{n,m}, W_{t+1}(\omega^m)). \end{aligned}$$

Step 6. Do for $t = T-1, \ldots, 0$:

Step 6a. Accumulate the path cost (with $\hat{v}_T^m = 0$)

$$\hat{v}_t^m = C_t(S_t^{n,m}, x_t^m) + \hat{v}_{t+1}^m$$

Step 6b. Update approximate value of the policy starting at time t:

$$\overline{V}_{t-1}^{n,m} \leftarrow U^V(\overline{V}_{t-1}^{n,m-1}, S_{t-1}^{x,n,m}, \hat{v}_t^m) \tag{8.16}$$

where we typically use $\alpha_{m-1} = 1/m$.

Step 7. Update the policy value function

$$V_t^{\pi,n}(S_t^x) = \overline{V}_t^{n,M}(S_t^x) \quad \forall t = 0, 1, \ldots, T$$

Step 8. Return the value functions $(V_t^{\pi,N})_{t=1}^T$.

Figure 8.7: Forward approximate dynamic programming.

9

Energy storage II

9.1 Narrative

We are now going to solve a somewhat more complex energy storage problem depicted in figure 9.1. In contrast with our previous storage system that just bought and sold energy from the grid, we now face the problem of meeting a time-dependent load for a building using energy from a wind farm and the grid, with a single energy storage device to help smooth the different processes.

This problem is also going to exhibit another distinctive property, which is that all the exogenous processes (wind, prices, loads and temperature) are going to come from a dynamic process that varies with different types of predictability:

- Loads - The demand for energy follows a fairly predictable pattern that depends on time of day (a building needs to be at a particular temperature by 8am when people start showing up) as well as temperature.

- Temperature - The temperature is a reasonably predictable process that depends on time of day and season, but also reflects local weather conditions that can be forecasted with some accuracy.

Figure 9.1: Energy system to serve a load (building) from a wind farm (with variable wind speeds) the grid (with variable prices), and a battery storage device.

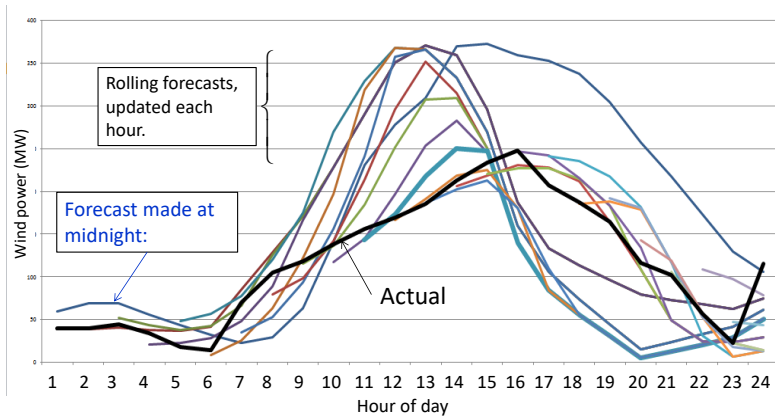


Figure 9.2: Evolution of evolving forecasts of wind power over the course of a 24-hour period, with hourly updates. The black line is the actual.

- Wind - There are vendors who perform forecasting services for wind, although the forecasts are not very accurate and evolve fairly quickly even over the course of the day (see figure 9.2).
- Prices - Prices are correlated with temperature which can be forecasted, but they require their own forecast model.

Each of these processes can be forecasted with varying degrees of accuracy. Forecasting wind power is the least accurate, and while wind can be stronger at night, the diurnal pattern is the weakest of the three. Loads are highly correlated with hour of day, largely because of human

activity but also because of temperature. Note that hot afternoons can create high lights in the middle of a summer day, while it can actually reduce the heating load (which may be served by electrical heating) during the winter. Temperature also has a strong hour-of-day component due to the rising and setting of the sun, but there can be variations as weather fronts move through.

These time-dependent patterns, combined with forecasts with varying degrees of accuracy, introduces new complexities, both in terms of the modeling the uncertainties but also in the design of policies.

9.2 Basic model

9.2.1 State variables

We start by modeling the snapshot of the system at time t which includes

$$\begin{aligned}
R_t &= \text{The amount of energy (in MWh) stored in the battery at time } t,\\
L_t &= \text{The load (demand) for energy at time } t \text{ (in MW)},\\
\tau_t &= \text{The temperature at time } t,\\
W_t &= \text{The energy from wind at time } t \text{ (in MW)},\\
p_t^{load} &= \text{The amount we are paid per MWh to satisfy the load to the building at time } t,\\
c_t^{grid} &= \text{The cost of purchasing power from the. grid (this is the price we are paid if we sell back to the grid)}
\end{aligned}$$

Since the underlying problem is highly nonstationary, we are going to need to use the availability of forecasts, both to model the dynamics of the problem as well as to make decisions which need to anticipate what might happen in the future. We assume that we are given a rolling set of forecasts (as illustrated for wind in figure 9.2. We model the forecasts for load (L), temperature (T), wind (W), market prices (p), and grid

prices (G) using

$f^L_{tt'}$ = Forecast of the load L_t (in MW) at time $t' > t$ given what we know at time t,

$f^\tau_{tt'}$ = Forecast of the temperature τ_t at time $t' > t$ given what we know at time t,

$f^W_{tt'}$ = Forecast of the wind power W_t (in MW) at time $t' > t$ given what we know at time t,

$f^p_{tt'}$ = Forecast of market prices p^{load}_t (in \$/MWh) at $t' > t$ given what we know at time t,

$f^G_{tt'}$ = Forecast of grid prices c^{grid}_t (in \$/MWh) at $t' > t$ given what we know at time t.

All forecasts are vectors over the horizon $t, t+1, \ldots, t+H$ where H is a specified horizon (e.g. 24 hours). We let f^X_t be the vector of forecasts for $X \in \mathcal{X} = \{L, T, W, P, G\}$.

Our state variable is then

$$S_t = (R_t, (L_t, \tau_t, W_t, p^{load}_t, c^{grid}_t), (f^L_t, f^T_t, f^W_t, f^P_t, f^G_t)).$$

Here we have grouped the controllable resource R_t, the snapshot of load, temperature, wind power and price, and then the forecasts.

A more compact representation is to let f^X_{tt} be the actual value of variable $X \in \{L, T, W, p, G\}$. In other words,

$$\begin{aligned} f^L_{tt} &= L_t, \\ f^T_{tt} &= \tau_t, \\ f^W_{tt} &= W_{tt}, \\ f^p_{tt} &= p^{load}_t, \\ f^G_{tt} &= c^{grid}_t. \end{aligned}$$

This notation allows us to write the state variable as

$$S_t = (R_t, (f^X_t)_{X \in \mathcal{X}}).$$

where

$$f^X_t = (f^X_{tt'})_{t' \geq t}$$

and

$$f^X_{tt} = X_t$$

9.2.2 Decision variables

The decision variables for our system are now

$$
\begin{aligned}
x_t^{wr} &= \text{The amount of power moved from the wind farm to the battery at time } t,\\
x_t^{w\ell} &= \text{The amount of power moved from the wind farm to the load (the building) at time } t,\\
x_t^{gr} &= \text{The amount of power moved from the grid to the battery at time } t,\\
x_t^{rg} &= \text{The amount of power moved from the battery to the grid at time } t,\\
x_t^{g\ell} &= \text{The amount of power moved from the grid to the load at time } t,\\
x_t^{r\ell} &= \text{The amount of power moved from the battery to the load at time } t,\\
x_t^{loss} &= \text{Uncovered load (known as “load shedding”).}
\end{aligned}
$$

subject to the constraints

$$
\begin{aligned}
x_t^{w\ell} + x_t^{g\ell} + \eta x_t^{r\ell} + x_t^{loss} &= L_t, && (9.1)\\
x_t^{r\ell} + x_t^{rg} &\leq R_t, && (9.2)\\
x_t^{wr} + x_t^{gr} &\leq \frac{1}{\eta}(R^{max} - R_t), && (9.3)\\
x_t^{w\ell} + x_t^{wr} &\leq W_t, && (9.4)\\
x_t^{wr} + x_t^{gr} &\leq u^{charge}, && (9.5)\\
x_t^{r\ell} + x_t^{rg} &\leq u^{discharge}, && (9.6)\\
x_t^{wr}, x_t^{w\ell}, x_t^{gr}, x_t^{rg}, x^{g\ell}, x_t^{r\ell} &\geq 0. && (9.7)
\end{aligned}
$$

Equation (9.1) limits the power to serve the load (the building) to the amount of the load (we cannot overload the building). The variable x^{loss} captures the amount by which we have not covered the load. The constraint captures conversion losses for energy taken from the battery (similar losses might apply to the energy from the wind farm, since this might be DC power that has to be converted to AC). Equation (9.2)

says that we cannot move more power out of our battery storage than is in the battery. Equation (9.3) then limits how much we can move into the battery to the amount of available capacity, adjusted by the conversion losses. Equation (9.4) limits the amount from the wind farm to what the wind farm is generating at that point in time. Equations (9.5) (9.6) limit the flows into and out of the battery to charge and discharge rates.

As always, we assume that decisions are made with a policy $X^\pi(S_t)$, to be determined below. For this problem, the policy has to return a five-dimensional vector.

9.2.3 Exogenous information

Following the lead of the first energy model, we are going to represent exogenous information in the form of changes. We again let f_t^X be a vector of forecasts for each information process $X \in \mathcal{X}$, where f_{tt}^X is the actual value at time t. Let

$$\begin{aligned} \hat{f}_{t+1,t'}^X &= \text{The change in the forecast for time } t' \\ &\quad \text{between } t \text{ and } t+1, \\ &= f_{t+1,t'}^X - f_{tt'}^X,\ t' = t, t+1, \ldots, t+H. \end{aligned}$$

The exogenous changes $\hat{f}_{t+1,t'}^X$ are correlated across the time periods t'. If this were not the case, then the forecasts when plotted over the horizon $t' = t, \ldots, t+H$ would no longer exhibit the smoothness that we see in the wind forecasts in figure 9.2. We return to the issue of modeling the uncertainty in forecasts in the uncertainty modeling section.

9.2.4 Transition function

The evolution of the resource state variable is given by

$$R_{t+1} = R_t + \eta x_t. \tag{9.8}$$

We write the evolution of the forecasts using

$$f_{t+1,t'}^X = f_{tt'}^X + \hat{f}_{t+1,t'}^X,\ X \in \mathcal{X},\ t' = t+1, \ldots, t+1+H. \tag{9.9}$$

9.2.5 Objective function

Our profit function at time t is given by

$$C(S_t, x_t) \quad = \quad (x_t^{w\ell} + x_t^{g\ell} + \eta x_t^{r\ell}) p_t^{load} - (x_t^{g\ell} + x_t^{gr}) c_t^{grid},$$

where the market price p_t^{load} and grid price c_t^{grid} are contained in the state variable S_t. Our objective function is still the canonical problem given by

$$\max_{\pi} \mathbb{E} \sum_{t=0}^{T} p_t X^{\pi}(S_t),$$

As before, $S_{t+1} = S^M(S_t, X^{\pi}(S_t))$ is given by equations (9.8) and (9.9).

9.3 Modeling uncertainty

We describe below two styles for modeling uncertainty over time. We first describe a method for modeling the correlations in errors in the forecasts over time using a technique sometimes called *Gaussian process regression*. This method ensures that as we move forward in time, a vector of forecasts evolves in a natural way.

Then we describe a hidden-semi Markov model that we have found provides exceptionally realistic sample paths for stochastic processes. This model closely replicates error distributions, but also does a very nice job capturing *crossing times*, which is the time that the actual process (e.g. wind speed) stays above or below a benchmark such as a forecast.

This section will illustrate several powerful methods for stochastic modeling that should be in any toolbox for modeling uncertainty.

9.3.1 GPR for forecast errors

Let $X_{t'}$ be the actual outcome of any of our exogenous processes (prices, loads, temperature, wind energy) at time t', and let $f_{tt'}^X$ be the forecast of $X_{t'}$ made at time $t < t'$. It is common to assume some error $\varepsilon_{t'-t}$ that describes the difference between $X_{t'}$ and the forecast $f_{tt'}^X$. We would then assume some model for $\varepsilon_{t'-t}$ such as

$$\varepsilon_{t'-t}^X \quad \sim \quad N(0, (t'-t)\sigma_X^2).$$

We are going to take a somewhat different approach by assuming that the distribution in the change in a forecast $\hat{f}^X_{t+1,t'}$ is described by

$$\hat{f}^X_{t+1,t'} \quad \sim \quad N(0, \sigma^2_X).$$

We then assume that the changes in the forecasts $\hat{f}^X_{t+1,t'}$ are correlated across times t' with a covariance function

$$Cov(\hat{f}^X_{t+1,t'}, \hat{f}^X_{t+1,t''}) \quad = \quad \sigma^2_X e^{-\beta|t''-t'|}, \tag{9.10}$$

We can use this covariance function to create a covariance matrix Σ^X with element $\Sigma^X_{t't''} = \sigma^2_X e^{-\beta|t''-t'|}$. There is a simple way of creating a correlated sample of changes in forecasts using a simple method called Cholesky decomposition. It begins by creating what we might call the "square root" of the covariance matrix Σ^X which we store in a lower triangular matrix L. In python, using the NumPy package, we would use the python command

$$L \quad = \quad \texttt{scipy.linalg.cholesky}(\Sigma^X, lower = True)$$

We note that

$$\Sigma^X \quad = \quad L^T L,$$

which is why we think of L as the square root of Σ^X.

Next generate a sequence of independent random variables Z_τ for $\tau = 1, \ldots, H$ that are normally distributed with mean 0 and variance 1. Now let $Z = (Z_{t+1}, Z_{t+2}, \ldots, Z_{t+H})^T$ be a column vector made up of these independently distributed standard normal random variables. We can create a correlated sample of changes in the forecasts using

$$\begin{pmatrix} \hat{f}^X_{t+1,t+1} \\ \hat{f}^X_{t+1,t+2} \\ \vdots \\ \hat{f}^X_{t+1,t+H} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} + LZ.$$

This formula gives us a sampled set of changes in forecasts $\hat{f}^X_{t+1,t+1}, \ldots, \hat{f}^X_{t+1,t+H}$ that are correlated according to our exponential decay function in equation (9.10). The result will be an evolving set of forecasts where the

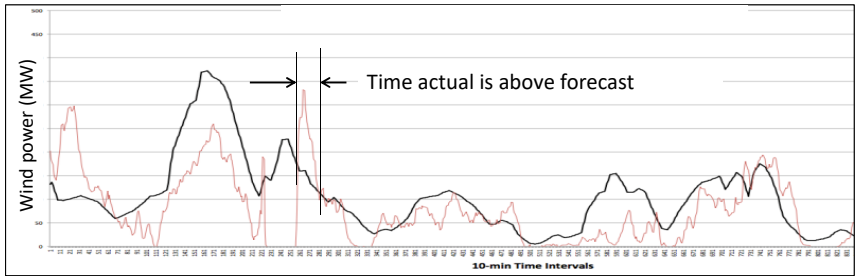


Figure 9.3: Forecasted (black) and actual wind power, illustrating a period where the actual is above the forecast. The length of this period of being above is called an up-crossing time.

variance in the errors in the forecasts grows linearly over time according to

$$Var(\varepsilon^X_{t'-t}) \quad = \quad (t'-t)\sigma^2_X.$$

The evolving forecasts $f^X_{t,t'}, f^X_{t+1,t'}, \ldots$ will exhibit the behavior that we saw in our evolving wind forecasts in figure 9.2.

9.3.2 Hidden semi-Markov model

A challenge when developing stochastic models in energy is capturing a property known as a *crossing time*. This is the time that an actual process (e.g. price or wind speed) is above or below some benchmark such as a forecast. Figure 9.3 illustrates an up-crossing time for a wind process.

Replicating crossing times using standard time series modeling proved unsuccessful. What did work was the development of Markov model with a hidden state variable S^C_t which is calibrated to capture the dynamics of the process moving above or below the benchmark. The process uses the following steps:

Step 1 Comparing the actual process to the benchmark, find the times at which the actual process moves above or below the benchmark, and output a dataset that captures whether the process was above (A) or below (B) and for how long. Aggregate these periods into three buckets (S/M/L) for short/medium/long, and label each

segment with A/B-S/M/L, creating six states. These are called "hidden states" because, while we will know at time t if the actual process is above or below the benchmark, we will not know if the length is short, medium or long until after the process crosses the benchmark.

Step 2 Using the historical sequence of S_t^C, compute a one-step transition matrix

$$P^C[S_{t+1}^C|S_t^C] = \text{The probability that the crossing process takes on value } S_{t+1}^C \text{ given that it is currently in state } S_t^C.$$

Step 3 Aggregate the actual process (e.g. wind speed) into, say, five buckets based on the empirical cumulative distribution. Let W_t^g be the aggregated wind speed (a number from 1 to 5).

Step 4 From history, compute the conditional distribution of wind speed given W_t^g and S_t^C,

$$F^W[W_{t+1}|W_t^g, S_t^C] = \text{Empirical cumulative distribution of the wind speed } W_{t+1} \text{ given } W_t^g \text{ and } S_t^C.$$

Using the one-step transition matrix $P^C[S_{t+1}^C|S_t^C]$ and the conditional cumulative distribution $F^W[W_{t+1}|W_t^g, S_t^C]$, we can now simulate our stochastic process by first simulating the hidden state variable S_{t+1}^C given S_t^C (note that there are only 30 of these). Then, from a wind speed W_t, we can find the aggregated wind speed W_t^g, and then sample the actual wind speed W_{t+1} from the conditional cumulative distribution $F^W[W_{t+1}|W_t^g, S_t^C]$.

This logic has been found to accurately reproduce both the error distribution (actual vs. benchmark), as well as both up-crossing and down-crossing distributions across a range of datasets modeling wind as well as grid prices. Figure 9.4 illustrates these distributions on a particular dataset.

9.4 Designing policies

The biggest complication of this problem is that the forecasts are part of the state variable, which allows us to explicitly model the rolling

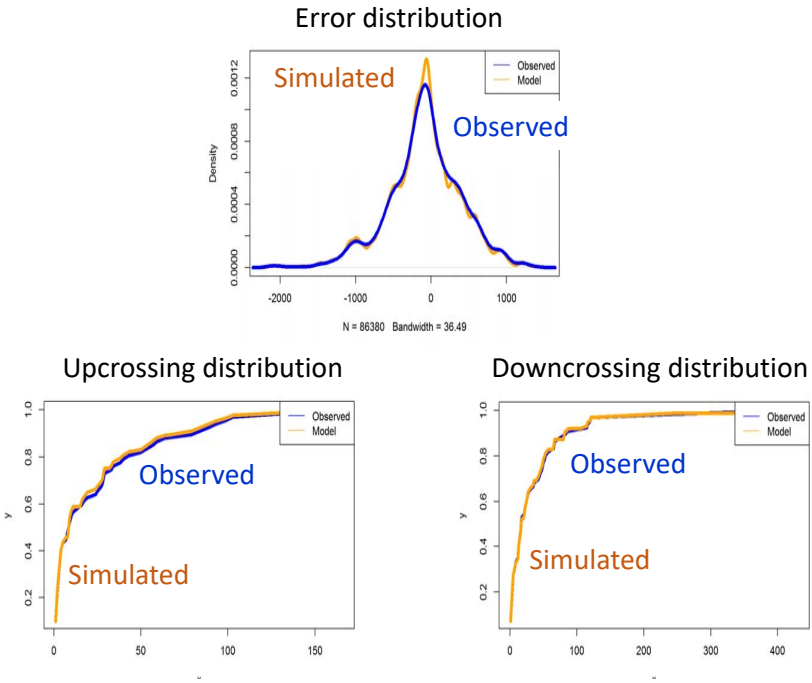


Figure 9.4: Comparison of actual vs predicted forecast error distributions (top), up-crossing time distributions (bottom left) and down-crossing time distributions (bottom right).

evolution of the forecast. The difficulty is that this makes the state variable high-dimensional.

The most common approaches for handling forecasts use a lookahead model that approximates the future by fixing the forecast. The two most popular strategies are:

- Stochastic lookahead with latent variables - We can use the forecast to develop a stochastic lookahead model which we then solve using classical dynamic programming. In the lookahead model, we fix the forecasts into the model, rather than modeling their evolution over time. When we ignore a variable in a model (including a lookahead model), it is called a *latent variable*.

- Deterministic lookahead with the forecast captured in the formu-

lation of the lookahead.

Both of these methods use the forecast as a latent variable in that they do not model the evolution of the forecast within the lookahead model. The difficulty with the stochastic lookahead model is that it is harder to solve. If we are optimizing our energy storage problem in short-time increments (this might be 5-minutes, or even less), then this can create problems for techniques such as exact or even approximate dynamic programming.

For this reason, the most popular strategy for handling time-dependent problems with a forecast is to solve a deterministic lookahead model, just as we did for our dynamic shortest path problem. We describe such a model below, and then introduce a parameterized version that allows the deterministic model to better handle uncertainty.

9.4.1 Deterministic lookahead

We are going to use the same notational style we first introduced in chapter 6, where we distinguish our *base model*, which is the problem we are trying to solve, from the *lookahead model* which we solve as a form of policy for solving the base model.

Recall that the canonical formulation of our base model is

$$\max_{\pi} \mathbb{E}\left\{\sum_{t=0}^{T} C(S_t, X^{\pi}(S_t))|S_0\right\},$$

where $S_{t+1} = S^M(S_t, X^{\pi}(S_t), W_{t+1})$, and where we have an exogenous information process $(S_0, W_1, W_2, \ldots, W_T)$ (the variables W_t may depend on the state S_t and/or decision x_t).

We are going to create a policy by formulating a deterministic lookahead model, where all the variables are labeled with tilde's, and indexed both by the time t at which we are making our decision, and time t' which is the time variable within the lookahead model. So we

would define

$$\begin{aligned}
\tilde{x}_{tt'} &= \text{The decision at time } t' \text{ in the lookahead model being generated at time } t,\\
\tilde{c}_{tt'} &= \text{The cost coefficient for } \tilde{x}_{tt'},\\
\tilde{R}_{tt'} &= \text{The energy in the battery at time } t' \text{ in the lookahead model generated at time } t.
\end{aligned}$$

Note that $x_t = \tilde{x}_{tt}$, $c_t = \tilde{c}_{tt}$ and so on.

We create our deterministic lookahead policy $X_t^{DLA}(S_t)$ as the following linear program:

$$X_t^{DLA}(S_t) = \arg\max_{x_t,(\tilde{x}_{tt'},t'=t+1,\ldots,t+H)} C(S_t,x_t) + \sum_{t'=t+1}^{t+H} C(\tilde{S}_{tt'},\tilde{x}_{tt'}) \quad (9.11)$$

where

$$C(S_t,x_t) = (x_t^{w\ell} + x_t^{g\ell} + \eta x_t^{r\ell})p_t^{load} - (x_t^{g\ell} + x_t^{gr})c_t^{grid}, \quad (9.12)$$

$$C(\tilde{S}_{tt'},\tilde{x}_{tt'}) = (\tilde{x}_{tt'}^{w\ell} + \tilde{x}_{tt'}^{g\ell} + \eta\tilde{x}_{tt'}^{r\ell})\tilde{p}_{tt'}^{load} - (\tilde{x}_{tt'}^{g\ell} + \tilde{x}_{tt'}^{gr})\tilde{c}_{tt'}^{grid}. \quad (9.13)$$

This problem has to be solved subject to the constraints (9.1)-(9.7) for x_t, and the following constraints for $\tilde{x}_{tt'}$ for all $t' = t+1,\ldots,t+H$:

$$\tilde{R}_{t,t'+1} - (R_{tt'} - x_t^{r\ell} + \eta x_{tt'}^{wr} + \eta x_{tt'}^{gr} - x_{tt'}^{rg}) = 0, \quad (9.14)$$

$$\tilde{x}_{tt'}^{w\ell} + \tilde{x}_{tt'}^{g\ell} + \eta\tilde{x}_{tt'}^{r\ell} + \tilde{x}_{tt'}^{loss} \leq f_{tt'}^{L}, \quad (9.15)$$

$$\tilde{x}_{tt'}^{r\ell} + \tilde{x}_{tt'}^{rg} \leq \tilde{R}_{tt'}, \quad (9.16)$$

$$\tilde{x}_{tt'}^{wr} + \tilde{x}_{tt'}^{gr} \leq \frac{1}{\eta}(R^{max} - \tilde{R}_{tt'}), \quad (9.17)$$

$$\tilde{x}_{tt'}^{w\ell} + \tilde{x}_{tt'}^{w\ell} \leq f_{tt'}^{W}, \quad (9.18)$$

$$\tilde{x}_{tt'}^{wr} + \tilde{x}_{tt'}^{gr} \leq u^{charge}, \quad (9.19)$$

$$\tilde{x}_{tt'}^{r\ell} + \tilde{x}_{tt'}^{rg} \leq u^{discharge}, \quad (9.20)$$

$$\tilde{x}_{tt'} \geq 0. \quad (9.21)$$

These equations mirror the ones in the base constraints (9.1)-(9.7), with the only change that we use the lookahead variables such as $\tilde{x}_{tt'}$, $\tilde{R}_{tt'}$, and forecasts such as $f_{tt'}^{W}$ instead of the actual wind W_t.

The model described by equations (9.11) - (9.21) is a relatively simple linear program, for which packages are now available in languages such as Matlab or python.

Lookahead policies such as $X_t^{DLA}(S_t)$ are widely used in dynamic, time-varying problems such as this. They have to be solved on a rolling basis as we first illustrated in figure 6.1 for our deterministic shortest path problem. For this reason, these are sometimes called "rolling horizon procedures" or "receding horizon procedures." There is an entire field known as "model predictive control" which is based on these lookahead policies.

For applications such as this energy storage problem, the use of a deterministic lookahead model raises the concern that we are not accounting for uncertainties. For example, we might want to store extra energy in the battery to protect ourselves from a sudden drop in wind or a surge in prices on the grid.

9.4.2 Parameterized lookahead

There is a very simple way to address the problem that our deterministic lookahead does not handle uncertainty. What we need to do is to think about how we might modify the model (or the solution) because of uncertainty. For example, we might wish to play for extra storage *in the future* to handle unexpected variations. Of course, we cannot force the model to keep energy in storage right now when we might need it. We might also want to factor-down forecasts that might not be very accurate.

We can introduce these changes by replacing the constraints (9.14)-

(9.21) with the following

$$\tilde{R}_{t,t'+1} - (\tilde{R}_{tt'} - \tilde{x}^{r\ell}_t + \eta\tilde{x}^{wr}_{tt'} + \eta\tilde{x}^{gr}_{tt'} - \tilde{x}^{rg}_{tt'}) = 0, \quad (9.22)$$

$$\tilde{x}^{w\ell}_{tt'} + \tilde{x}^{g\ell}_{tt'} + \eta\tilde{x}^{r\ell}_{tt'} + \tilde{x}^{loss}_{tt'} = \theta^L_{t'-t} f^L_{tt'}, \quad (9.23)$$

$$\tilde{x}^{r\ell}_{tt'} + \tilde{x}^{rg}_{tt'} \leq \theta^R_{t'-t}\tilde{R}_{tt'}, \quad (9.24)$$

$$\tilde{x}^{wr}_{tt'} + \tilde{x}^{gr}_{tt'} \leq \frac{1}{\eta}(R^{max} - \tilde{R}_{tt'}) \quad (9.25)$$

$$\tilde{x}^{w\ell}_{tt'} + \tilde{x}^{w\ell}_{tt'} \leq \theta^W_{t'-t} f^W_{tt'}, \quad (9.26)$$

$$\tilde{x}^{wr}_{tt'} + \tilde{x}^{gr}_{tt'} \leq u^{charge}, \quad (9.27)$$

$$\tilde{x}^{r\ell}_{tt'} + \tilde{x}^{rg}_{tt'} \leq u^{discharge}, \quad (9.28)$$

$$\tilde{x}_{tt'} \geq 0. \quad (9.29)$$

Note that we have introduced parameters to modify the right hand side of constraints (9.22) and (9.25) where we have introduced coefficients $\theta^L_{t'-t}$ and $\theta^W_{t'-t}$ to modify the forecasts of load and wind, where the coefficients are indexed by how many time periods we are forecasting into the future. Then, we modified the constraint (9.23) with the idea that we may want to restrict our ability to use all the energy in storage in order to maintain a reserve. The coefficient $\theta^R_{t'-t}$, is also indexed by how many time periods we are looking into the future.

Let $X^{DLA-P}(S_t|\theta)$ represent the lookahead policy that is solved subject to the parameterized constraints (9.22) - (9.29). Once we have decided how to introduce these parameterizations (this is the art behind any parametric model), there is the problem of finding the best value for θ. This is the problem of parameter search that we addressed in chapter 7.

10

Supply chain management I: The two-agent newsvendor

10.1 Narrative

Imagine that a logistics manager for Amazon has to provide trailers to move freight out of Chicago on a weekly basis. The logistics manager, working in the field, has access to information that allows him to estimate how many trailers he will need that week, but the actual might be higher or lower. The field manager then makes a request to a central manager for trailers, who then makes a judgment on her own how many trailers to provide, and makes the final decision on the number of trailers that will be provided.

The two managers both work for the same company, but the field manager is far more worried about running out, since he has to do short-term rentals if he runs out of trailers. The central manager, on the other hand, also does not want the field manager to run out, but she also does not want him to have excess trailers, since she has to pay for those trailers.

We assume the process unfolds as follows:

Step 1: The field manager observes the initial estimate of how many trailers are needed. This information is private to the field man-

ager.

Step 2: The field manager then requests trailers from the central manager.

Step 3: The central manager then decides how many trailers to give to agent q.

Step 4: The field manager receives the number of trailers granted by the central manager, and then observes the actual need for trailers.

Step 5: The field and central managers compute their own costs.

The tension in this problem arises first because the initial estimate of trailers needed is just an estimate, which we may assumed is unbiased (that is, it is true on average). The problem is that the field manager has a large cost if he runs out, so his strategy is to overestimate his needs (recall the newsvendor problem in chapter 3). The central manager, on the other hand, probably has balanced costs for being over and under, and does not want to order too many or too few.

What complicates the problem is the initial estimate given to the field manager. While not perfect, it has value information since it will indicate if a day is going to have high or low demand. This means that the central manager has to pay attention to the request made by the field manager, while recognizing that the field manager will make request that are biased upward. Knowing this, the central manager would tend to use the field manager's request as a starting point, but then reduce this for the final allocation. Not surprisingly, the field manager knows that the central manager will do this, and compensates accordingly.

10.2 Basic model

We will model the problem for both agents, since the information is not the same for both players. Throughout the model, we will be referring to the field manager as q and to the central manager as q'.

10.2.1 State variables

The initial information available to the field manager is the estimate of the number of trailers that will be needed, which we represent using

$$R_{tq}^{est} = \text{The initial estimate of how many trailers are needed.}$$

This initial estimate may be biased, so we are going to let our initial estimate of the bias be

$$\delta_{tq}^{est} = \text{Initial estimate of the difference between } R_{tq}^{est} \text{ and the true demand.}$$

We will also have to estimate how much the central manager reduces the request of the field manager, which we represent using

$$\delta_{tq}^{central} = \text{Estimate of how much the central manager will reduce the request of the field manager, which we define (below) as } x_{tqq'}.$$

Similarly, the central manager will learn the difference between the request made by the field manager, and what the field manager eventually needs, which we represent by

$$\delta_{tq'}^{field} = \text{Estimate of the difference between what the field manager requests } (x_{tqq'}), \text{ and what the field eventually needs.}$$

The state variable for each agent is the information they have before they make a decision. For the field manager, the state variable is

$$S_{tq} = (R_{tq}^{est}, \delta_{tq}^{est}, \delta_{tq}^{central}).$$

The state variable for the central manager is

$$S_{tq'} = (x_{tqq'}, \delta_{tq'}^{field}).$$

10.2.2 Decision variables

The decisions for each agent are given by

$$\begin{aligned} x_{tqq'} &= \text{The number of trailers that agent } q \\ &\quad \text{asks for from agent } q', \\ x_{tq'q} &= \text{The number of trailers that agent } q' \\ &\quad \text{gives to agent } q, \text{ which is what is im-} \\ &\quad \text{plemented in the field.} \end{aligned}$$

10.2.3 Exogenous information

The exogenous information for the field manager can be thought of as the initial estimate of the trailers needed (although we put that in the state variable):

$$\begin{aligned} R_{tq}^{est} &= \text{The initial estimate of how many trail-} \\ &\quad \text{ers are needed.} \end{aligned}$$

After making the decision $x_{tqq'}$, we then receive two types of information: what the central manager grants us, and then the actual required demand:

$$\begin{aligned} x_{tq'q} &= \text{The decision made by the central man-} \\ &\quad \text{ager in response to the request of the} \\ &\quad \text{field manager,} \\ \hat{R}_{t+1} &= \text{The actual number of trailers that field} \\ &\quad \text{manager } q \text{ ends up needing (this in-} \\ &\quad \text{formation is available to the central} \\ &\quad \text{manager as well).} \end{aligned}$$

The exogenous information for agent q is then

$$W_{t+1,q} = (x_{tq'q}, \hat{R}_{t+1}).$$

We note in passing that while this information is indexed at time $t+1$, the request from the field, $x_{tqq'}$, is indexed by t (since it depends on information available up through time t).

The central manager receives the initial request $x_{tqq'}$ which arrives as exogenous information, but because this is received before she makes

her decision, then enters through the state variable for the central manager. The only exogenous information for the central manager is the final demand, so

$$W_{t+1,q'} = (\hat{R}_{t+1}).$$

10.2.4 Transition function

For the field manager, there are three state variables: R^{est}_{tq}, the bias δ^{est}_{tq} between the estimate R^{est}_{tq} and the actual $\hat{R}_t$, and the bias $\delta^{central}_{tq}$ introduced by the central manager when the field makes a request. The first state variable, R^{est}_{tq}, arrives directly as exogenous information. The biases δ^{est}_{tq} and $\delta^{central}_{t,q}$ are updated using

$$\begin{aligned}\delta^{est}_{t+1,q} &= (1-\alpha)\delta^{est}_{tq} + \alpha(\hat{R}_t - R^{est}_t)\\ \delta^{central}_{t+1,q} &= (1-\alpha)\delta^{central}_{tq} + \alpha(x_{tqq'} - x_{tq'q}).\end{aligned}$$

The transition function for the central manager is similar. Again, the decision of the field manager, $x_{tqq'}$ arrives to the state variable exogenously. Then, we update the bias that the central manager estimates in the request of the field manager using

$$\delta^{field}_{t+1,q'} = (1-\alpha)\delta^{field}_{t,q'} + \alpha(x_{tqq'} - \hat{R}_t).$$

10.2.5 Objective function

We begin by defining:

c^o_q = The unit cost incurred by the field manager for each excess trailer (what the field pays per day for each trailer), also known as the overage cost,

c^u_q = The unit cost incurred by the field manager for each trailer that has to be rented to make up for lack of capacity, also known as the underage cost,

$c^o_{q'}, c^u_{q'}$ = The cost of overage and underage for the central manager.

The costs for each agent are given by

$$\begin{aligned} C_{tq}(S_{tq}, x_{tq'q}) &= \text{Overage/underage costs for agent } q, \\ &= c^o_q \max\{x_{tq'q} - \hat{R}_t, 0\} + c^o_{q'} \max\{\hat{R}_t - x_{tq'q}, 0\}, \\ C_{tq'}(S_{tq'}, x_{tq'q}) &= \text{Overage/underage costs for agent } q', \\ &= c^o_{q'} \max\{x_{tq'q} - \hat{R}_t, 0\} + c^o_q \max\{\hat{R}_t - x_{tq'q}, 0\}. \end{aligned}$$

The performance of both the field and central managers depend on the number of trailers $x_{tq'q}$ that the central manager gives to the field. This decision, however, depends on the decision made by the field manager.

The decisions of the field manager are made with the policy $X^{field}_{tqq'}(S_t|\theta^{field}_q)$, where θ^{field}_q is one or more tunable parameters that are used to solve

$$\min_{\theta^{field}_q} \mathbb{E}\{\sum_{t=0}^{T} C_{tq}(S_{tq}, X^{field}_{tqq'}(S_t|\theta^{field}_q))|S_0\}. \tag{10.1}$$

Similarly, the decisions of the central manager are made with the policy $X^{central}_{tq'q}(S_t|\theta^{central}_{q'})$ where $\theta^{central}_{q'}$ is one or more tunable parameters that solve

$$\min_{\theta^{central}_{q'}} \mathbb{E}\{\sum_{t=0}^{T} C_{tq'}(S_{tq'}, X^{central}_{tqq'}(S_t|\theta^{central}_{q'}))|S_0\}. \tag{10.2}$$

The optimization problems in (10.1) and (10.2) have to be solved simultaneously, since both policies have to be simulated at the same time. Of course we could hold $\theta^{central}_{q'}$ for the central manager constant while tuning θ^{field}_q for the field manager, but ultimately we are looking for a stable local minimum.

10.3 Modeling uncertainty

This problem is data-driven, which means that we react to data as it arrives. There are three types of information, depending on which agent is involved:

- The initial estimate R^{est}_t of the resources required.

- The request $x_{tqq'}$, made by the field manager, that arrives to the central manager. This decision involves logic introduced by the field manager, which may include randomization. This comes as information to the central manager.
- The decision $x_{tq'q}$ made by the central manager that determines the number of trailers given to the field manager. This comes as information to the field manager.
- The final realization $\hat{R}_t$ of the number of trailers actually required, which is revealed (in this basic model) to both agents.

If we wish to simulate the process, we only need to model the generation of R_t^{est} and $\hat{R}_t$. More precisely, we would have to generate R_t^{est} from one distribution, and the error $\hat{R}_t - R_t^{est}$ from another distribution.

10.4 Designing policies

For our two-agent newsvendor problem, we have to develop policies for each agent. We begin with the policy for the field manager.

10.4.1 Field manager

The field manager starts with an estimate R_t^{est}, but has to account for three factors:

1) The estimate R_t^{est} may have a bias β^{est} (we cannot be sure about the source of the estimate R_t^{est}). The bias is given by

$$\beta^{est} = \mathbb{E}\hat{R}_t - R_t^{est}.$$

So, if $\beta^{est} > 0$ then this means that R_t^{est} is upwardly biased.

2) The true number of trailers needed, $\hat{R}_t$ is random even once you have factored in the bias. The field manager has a higher cost of having too few trailers than too many, so he will want to introduce an upward bias to reflect the higher cost of being caught short.

3) The central manager has a balanced attitude toward having too many or too few, and knows about the bias of the field manager.

As a result, the central manager will typically use the request of the field manager, $x_{tqq'}$, just as the field manager may be adjusting for a possible bias between the estimate R^{est} and the actual $\hat{R}_t$. The field manager knows that the central manager will be making this adjustment, and as a result has to try to estimate it, and counteract it. Since the field manager knows both his request $x_{tqq'}$ and then sees what the central manager provides, the time t observation of the bias is given by

$$\beta_{tq}^{central} = x_{tq'q} - x_{tqq'}.$$

We need to use our estimates of the differences between R_t^{est} and $\hat{R}_t$, the difference between $x_{tqq'}$ and $x_{tq'q}$, and the difference between $x_{tqq'}$ and $\hat{R}_t$. We propose a policy for the field manager given by

$$X_{tqq'}^{field}(S_t|\theta_q^{field}) = R_t^{est} - \delta_{t-1,q}^{est} - \delta_{t-1,q}^{central} + \theta_q^{field}. \quad (10.3)$$

This policy starts with the initial estimate R_t^{est}, corrects for the bias in this initial estimate using $\delta_{t-1,q}^{est}$, then corrects for the bias from the central manager $\delta_{t-1,q}^{central}$, and then finally introduces a shift that can capture the different costs of over and under for the field manager. The parameter θ^{field} has to be tuned.

10.4.2 Central manager

Our policy for the central manager is given by

$$X_{tq'q}^{central}(S_t|\theta_{q'}^{central}) = x_{tqq'} - \delta_{t-1,q'}^{field} + \delta_{q'}^{central}. \quad (10.4)$$

Here, we start with the request made by the field manager, subtract our best estimate of the difference between the field manager's request and what was eventually needed, $\delta_{tq'}^{field}$, and then add in $\theta_{q'}^{central}$ which is a tunable parameter for the central manager.

11

Supply chain management II: The beer game

11.1 Narrative

This chapter covers a famous game from the 1950's known as the "beer game." This was originally designed by Jay Forrester, a professor at MIT who created the game to illustrate the instabilities of supply chains. The problem involves a linear supply chain where various suppliers move beer from where it is made (the manufacturer) toward the market (the retailer). The beer has to move through several middlemen on its way from point of manufacture to the market.

There are two types of flows:

- The flow of beer - Each case of beer is represented by a penny that moves from manufacturer to retailer.
- The flow of information - Each point in the supply chain replenishes its inventory by making requests for more beer to the next level down.

Each level of the supply chain is known as an *echelon*. Typically there are four to six echelons for each team. The demands at the retailer are fixed in advance, but hidden, in a deck of cards. As the retailer reveals

that week's demand, she tries to fill the demand from inventory. The retailer, and every other supplier in the supply chain (other than the manufacturer) then fills out a sheet of paper requesting more inventory.

It is possible that the retailer, or any of the middle suppliers, may not be able to fill the request for more inventory (or the market demand at the retail level). In this case, the unsatisfied demand sits in an order backlog waiting to be filled as new inventory arrives.

After filling orders, everyone (for that supply chain) has to stop and record either their inventory (the number of cases of beer sitting in inventory) or their order backlog. Order backlog carries a penalty of $2 per case. Excess inventory carries a holding cost of $0.50 per case.

The steps of the process are illustrated in figure 11.1. There are five steps:

Step 1: Each week, each player will have an inventory (in pennies, each representing a case of beer), and an order, which might be the retail demand (for the retail echelon) or an order placed by the player to their left.

Step 2: Each player tries to take as many cases from their inventory and move it to their left to a point *between* them and the player to the left (do not add the pennies to the inventory of the player to the left). If there is not enough inventory to satisfy the order, cross out the order and replace it with the number of cases that remain to be satisfied (this is the order backlog).

Step 3: Now write out an order of how many cases you want to replenish your inventory, and place it in the area *between* you and the player to your right (the manufacturer places an order on the pile of pennies from which all beer originates).

Step 4: Stop and record on your inventory sheet how much inventory you have. If you have not been able to satisfy an order, you will have no inventory, and you will have orders in your backlog (the unsatisfied orders). If this is the case, record this as your order backlog.

Step 5: This is the key step: reach out with your left hand and pull the next order slip into your order pile (the sheets of paper), and at the same time reach out with your right hand to pull the pennies coming to you into your inventory. Now you are back to where you were for step 1.

It is very important that everyone in the same chain be coordinated. The retailer has to play the role of keeping everyone synchronized.

11.2 Basic model

We are going to model a supplier other than one of the endpoints (retailer or beer manufacturer). This model will closely follow the style of the two-agent newsvendor problem in the previous chapter, although there are some adjustments.

In our basic model, we are going to ignore our ability to anticipate what the upstream echelon might do.

We are going to label the different agents in the supply chain by $\mathcal{Q} = \{1, 2, \ldots, Q\}$ where $q = 1$ represents the retailer and $q = Q$ represents the manufacturer.

11.2.1 State variables

$$\begin{aligned} R_{tq} &= \text{The inventory left over after iteration } t \text{ after delivering product to the upstream vendor for agent } q, \\ D_{tq} &= \text{Unsatisfied demand after satisfying the previous request(s).} \end{aligned}$$

The manufacturer also maintains inventory, although his requests for new product are always satisfied in full.

11.2.2 Decision variables

Our only decision is how much to order from the downstream vendor. Using the notation of the two-agent newsvendor problem, it is natural

to call this variable $x_{tq,q+1}$ since q will always make the request to agent $q+1$. We simply this notation by defining

$$x_{tq} = \text{Order placed by supplier } q \text{ to be passed to supplier } q+1, \text{ made at the order instant in iteration } t, \text{ which will be received by } q+1 \text{ to be filled in iteration } t+1.$$

There are no constraints on x_{tq} other than that it cannot be negative.

11.2.3 Exogenous information

There are two types of exogenous information for supplier q:

$$\begin{aligned} y_{t+1,q} &= \text{The amount of product received from supplier } q+1 \text{ in response to the request made at time } t, \\ x_{t,q-1} &= \text{The order made by supplier } q-1 \text{ at time } t \text{ of supplier } q, \text{ which would arrive at time } t+1. \end{aligned}$$

We assume that each supplier always fills as much of a request as possible, so

$$y_{t+1,q} = \min\{x_{tq}, R_{t,q+1}\}.$$

We can represent the exogenous information for agent q arriving by time $t+1$ using

$$W_{t+1} = (y_{t+1,q}, x_{t,q-1}).$$

11.2.4 Transition function

$$\begin{aligned} R_{t+1,q} &= \max\{0, R_{tq} - x_{t,q-1}\} + y_{t+1,q}, \\ D_{t+1,q} &= D_{tq} \end{aligned}$$

11.2.5 Objective function

... to be written

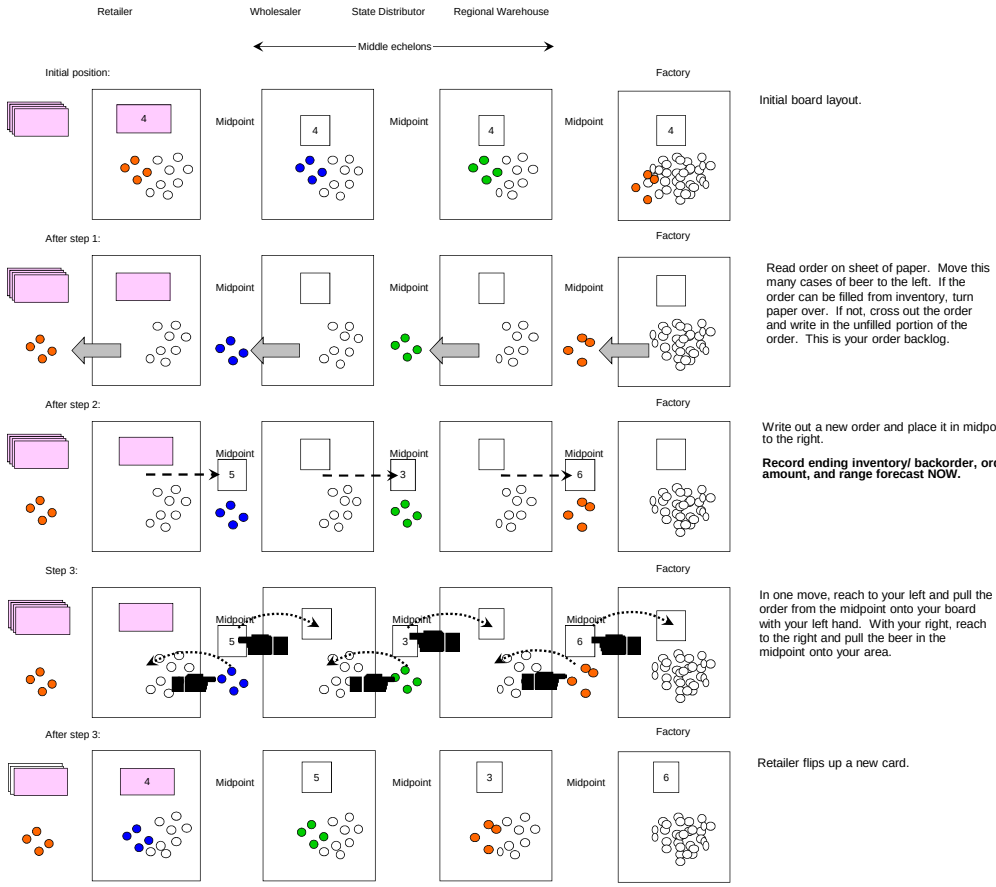


Figure 11.1: Layout of the Princeton version of the beer game.

12

Ad-click optimization

12.1 Narrative

Companies advertising on internet sites such as Google have to bid to get their ads in a visible position (that is, at the top of the list of sponsored ads). When a customer enters a search term, Google identifies all the bidders who have listed the same (or similar) search terms in their list of ad-words. Google then takes all the matches, and sorts them in terms of how much each participant has bid, and runs an auction. The higher the bid, the more likely your ad will be placed near the top of the list of sponsored ads, which increases the probability of a click. Figure 12.1 is an example of what is produced after entering the search terms "hotels in baltimore md."

If a customer clicks on the ad, there is an expected return that reflects the average amount a customer spends when they visit the website of the company. The problem is that we do not know the bid response curve. Figure 12.2 reflects a family of possible response curves. Our challenge is to try out different bids to learn which curve is correct.

We will begin by assuming that we can adjust the bid after every auction, which means we only learn a single response (the customer did

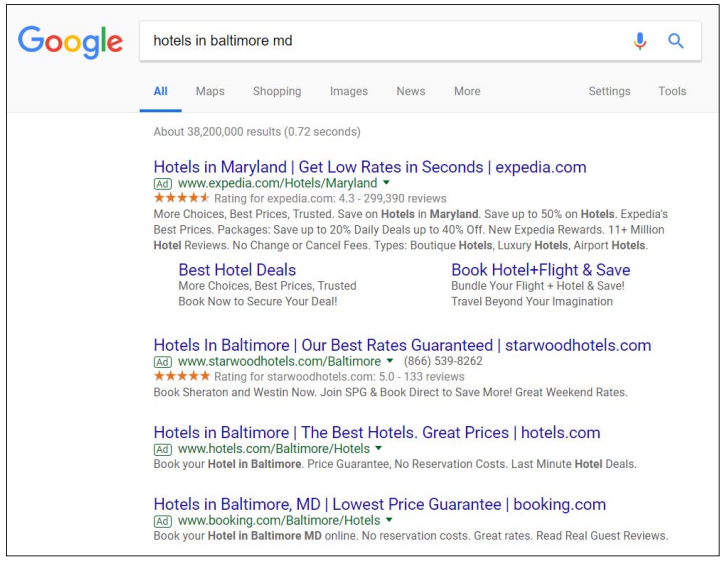


Figure 12.1: Possible instances of the probability of an ad getting a click given the bid.

or did not click on the link). It is possible that the customer looked at a displayed link and decided not to click on it, or our bid may have been so low that we were not even in the list of displayed ads.

Our challenge is to design a policy for setting the bids. The goal is to maximize the net revenue, including what we make from selling our products or services, minus what we spend on ad-clicks.

12.2 Basic model

We are going to assume that we use some sort of parameterized model to capture the probability that a customer clicks on an ad. At a minimum this probability will depend on how much we bid for an ad - the more we bid, the higher the ad will appear in the sponsored ad list, which increases the likelihood that a customer will click on it. Let $K^n = 1$ if the nth customer clicks on the ad. Let

$$P^{click}(x|\theta_k) \quad = \quad Prob[K^{n+1} = 1|\theta = \theta_k]$$

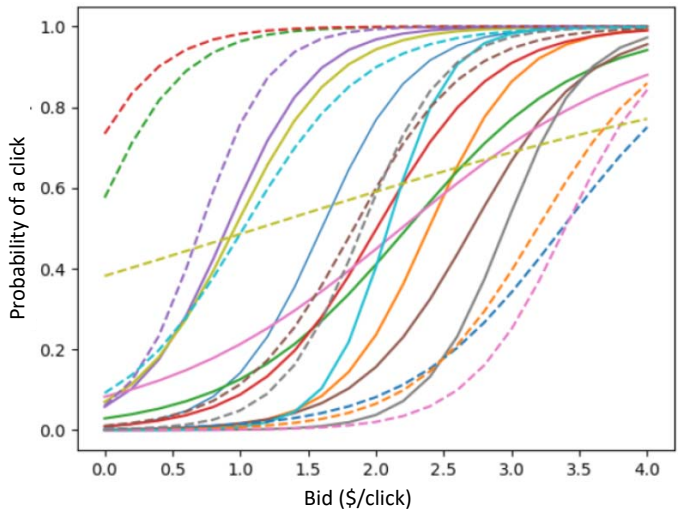


Figure 12.2: Possible instances of the probability of an ad getting a click given the bid.

where $P^{click}(x|\theta)$ will be described by a logistic function given by

$$Prob[K^{n+1} = 1|\theta = \theta_k, H^n] = \frac{e^{\theta_k^{const} + \theta_k^{bid} x^n}}{1 + e^{\theta_k^{const} + \theta_k^{bid} x^n}}. \quad (12.1)$$

This function is parameterized by $\theta = (\theta^{const}, \theta^{bid})$. We do not know what θ is, but we are going to assume that it is one of a sampled set $\Theta = \{\theta_1, \ldots, \theta_K\}$.

12.2.1 State variables

We use the following variables: The initial state S_0 includes

$$\begin{aligned} \Theta &= \{\theta_1, \ldots, \theta_K\} \\ &= \text{The set of possible values that } \theta \text{ may take,} \\ \bar{R}^0 &= \text{Initial estimate of revenue earned when a customer clicks on a link.} \end{aligned}$$

The dynamic state variables S^n includes:

$$\begin{aligned} p_k^n &= \text{Probability that the true } \theta = \theta_k, \\ p^n &= (p_k^n)_{k=1}^K, \\ \bar{R}^n &= \text{Estimate of revenue earned from an ad-click after } n \text{ auctions.} \end{aligned}$$

Our state variable, then, is

$$S^n = (\bar{R}^n, p^n).$$

Note that we can create a point estimate of θ after n observations using

$$\bar{\theta}^n = \sum_{k=1}^K p_k^n \theta_k,$$

but this is a statistic that we can compute from the information in S^n, so we do not put $\bar{\theta}^n$ in the state variable.

12.2.2 Decision variables

Our only decision variable is the bid which we define as

$$x^n = \text{The bid (in \$ per click) for the } n+1st \text{ auction.}$$

As before, we let $X^\pi(S^n)$ our generic policy that gives us the bid x^n as a function of the information available to us, represented by S^n, which means that we would write

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

We assume that the policy enforces any constraints, such as making sure that the bid is not negative or something too large.

12.2.3 Exogenous information

In our initial model, we only observe the results of a single auction, which we model using

$$\begin{aligned} K^{n+1} &= \begin{cases} 1 & \text{If the customer clicks on our ad,} \\ 0 & \text{Otherwise.} \end{cases} \\ \hat{R}^{n+1} &= \text{Revenue earned from the } n+1st \text{ auction.} \end{aligned}$$

This means our complete exogenous information vector is

$$W^{n+1} = (\hat{R}^{n+1}, K^{n+1}).$$

12.2.4 Transition function

We are going to update our estimated revenue when a customer clicks on the ad using

$$\bar{R}^{n+1} = \begin{cases} (1-\alpha^{lrn})\bar{R}^n + \alpha^{lrn}\hat{R}^{n+1} & \text{If } K^{n+1}=1, \\ \bar{R}^n & \text{Otherwise.} \end{cases} \quad (12.2)$$

Thus, we only update our estimated revenue when we get a click. The parameter α^{lrn} is a smoothing parameter (sometimes called a "learning rate") between 0 and 1 that we fix in advance.

We next address the updating of the probabilities p_k^n. We let H^n be the history of states, decisions and exogenous information

$$H^n = (S^0, x^0, W^1, S^1, x^1, \ldots, W^n, S^n, x^n).$$

We use this to write

$$p_k^n = Prob[\theta = \theta_k | H^n].$$

We use Bayes theorem to write

$$\begin{aligned} p_k^{n+1} &= Prob[\theta = \theta_k | W^{n+1}, H^n] \\ &= \frac{Prob[K^{n+1}|\theta = \theta_k, H^n] Prob[\theta = \theta_k | H^n]}{Prob[K^{n+1}|H^n]}. \end{aligned} \quad (12.3)$$

Remember that the history H^n includes the decision x^n which (given a policy for making these decisions) is directly a function of the state S^n. We now use our logistic curve in equation (12.10) to write

$$Prob[K^{n+1}=1|\theta=\theta_k, H^n] = Prob[K^{n+1}=1|\theta=\theta_k, x^n] \quad (12.4)$$

$$= \frac{e^{\theta_k^{const}+\theta_k^{bid}x^n}}{1+e^{\theta_k^{const}+\theta_k^{bid}x^n}}. \quad (12.5)$$

We then note that

$$Prob[\theta = \theta_k | H^n] = p_k^n. \quad (12.6)$$

Finally, we note that the denominator can be computed using

$$Prob[K^{n+1}|H^n] = \sum_{k=1}^{K} Prob[K^{n+1}|\theta = \theta_k, H^n]p_k^n. \quad (12.7)$$

Our use of a sampled representation of the possible outcomes of θ is saving us here. Even if θ has only two dimensions (as it is here, but only for now), performing a two-dimensional integral over a multivariate distribution for θ would be problematic.

Equations (12.5) - (12.7) allow us to calculate our Bayesian updating equation for the probabilities in (12.3). Equations (12.2)-(12.3) make up our transition function

$$S^{n+1} = S^M(S^n, x^n, W^{n+1}).$$

12.2.5 Objective function

We begin by writing the single period profit function as

$$C(S^n, x^n, W^{n+1}) = (\hat{R}^{n+1} - x^n)K^{n+1},$$

which means we make nothing if the customer does not click on the ad ($K^{n+1} = 0$). If the customer does click on the ad ($K^{n+1} = 1$), we receive revenue given by $\hat{R}^{n+1}$, but we also have to pay what we bid for the ad-click, given by our bid x^n.

We will end up taking the expected contribution, which we write as

$$\mathbb{E}\{C(S^n, x, W^{n+1})|S^n\} = \mathbb{E}\{(\hat{R}^{n+1} - x^n)K^{n+1}|S^n\} \quad (12.8)$$

There are three random variables hidden in the expectation: θ, with distribution p^n (contained in S^n), K^{n+1}, where $P^{click}(x|\theta) = Prob[K^{n+1} = 1|\theta]$, and $\hat{R}^{n+1}$, which we observe from some unknown distribution if $K^{n+1} = 1$, and where $\hat{R}^{n+1} = 0$ if $K^{n+1} = 0$ (we do not get any revenue if the customer does not click on the ad).

We can then break the expectation into three expectations:

$$\mathbb{E}\{(\hat{R}^{n+1} - x^n)K^{n+1}|S^n\} = \mathbb{E}_\theta\mathbb{E}_{K|\theta}\mathbb{E}_{\hat{R}}\{(\hat{R}^{n+1} - x^n)K^{n+1}|S^n\}.$$

We start by taking the expectation over $\hat{R}$ where we just use $\mathbb{E}\{\hat{R}^{n+1}|S^n\} = \bar{R}^n$ (remember that $\bar{R}^n$ is in S^n), which allows us to write

$$\mathbb{E}\{(\hat{R}^{n+1} - x^n)K^{n+1}|S^n\} = \mathbb{E}_\theta\mathbb{E}_{K|\theta}\{(\bar{R}^n - x^n)K^{n+1}|S^n\}.$$

Next we are going to take the expectation over K^{n+1} for a given θ using

$$\mathbb{E}_{K|\theta}\{(\bar{R}^n - x^n)K^{n+1}|S^n\} = (\bar{R}^n - x^n)P^{click}(x|\theta).$$

where we have used the fact that $(\bar{R}^n - x^n)K^{n+1} = 0$ if $K^{n+1} = 0$.

Finally we take the expectation over θ using

$$\mathbb{E}_{\theta}\{(\bar{R}^n - x^n)P^{click}(x|\theta)|S^n\} = \sum_{k=1}^{K}(\bar{R}^n - x^n)P^{click}(x|\theta = \theta_k)p_k^n.$$

We are going to let $\bar{C}(S^n, x)$ be the expected contribution, which is to say

$$\bar{C}(S^n, x) = \mathbb{E}_{\theta}\mathbb{E}_{K|\theta}\{(\bar{R}^n - x^n)K^{n+1}|S^n\}.$$

Our objective function can now be written in the canonical form

$$\max_{\pi} \mathbb{E}_{S^0}\{\mathbb{E}_{W^1,\ldots,W^n|S^0} \sum_{n=0}^{N} C(S^n, X^{\pi}(S^n), W^{n+1})|S_0\} = \mathbb{E}_{S^0}\{\sum_{n=0}^{N} \bar{C}(S^n, X^{\pi}(S^n))|S_0$$

Note that the conditioning on S_0 is how we communicate our prior $p_k^0 = Prob[\theta = \theta_k]$.

12.3 Modeling uncertainty

We have three forms of uncertainty: the ad-click K^{n+1}, the revenue we receive $\hat{R}^{n+1}$ if $K^{n+1} = 1$, and then true value of θ. We are going to assume that we simply observe $\hat{R}^{n+1}$ from a real datastream, which means we do not need a formal probability model for these random variables. We assume that K^{n+1} is described by our logistic function

$$P^{click}(x|theta) = P[K^{n+1} = 1|x = x^n, \theta] \quad (12.9)$$

$$= \frac{e^{\theta^{const}+\theta^{bid}x}}{1 + e^{\theta^{const}+\theta^{bid}x}}, \quad (12.10)$$

but it is important to recognize that this is just a fitted curve. The values of K^{n+1} are observed from data, which means we have no guarantee that the distribution precisely matches our logistic regression.

Finally, we assume that $\theta \in \Theta = \{\theta_1, \ldots, \theta_K\}$ which is also an approximation. There are ways to relax the requirement of a sampled set, but the logic becomes somewhat more complicated without adding much educational value.

12.4 Designing policies

12.4.1 Pure exploitation

The starting point of any online policy should be pure exploitation, which means doing the best that we can. To compute this we start by using

$$\begin{aligned}\mathbb{E}\hat{R}^{n+1}K^{n+1} &= \mathbb{E}\{\hat{R}^{n+1}|K^{n+1}=1\}Prob[K^{n+1}=1|\theta=\theta_k] \\ &= \bar{R}^n P^{click}(x|\theta).\end{aligned}$$

To find the best bid, we find (after a bit of algebra)

$$\begin{aligned}\frac{d\bar{C}(x)}{dx} &= (\bar{R}^n - x)\frac{dP^{click}(x|\theta)}{d\theta} - P^{click}(x|\theta) \\ &= 0,\end{aligned}$$

where

$$\frac{dP^{click}(x|\theta)}{dx} = \frac{\theta_1 e^{-\theta_0-\theta_1 x}}{(1+e^{-\theta_0-\theta_1 x})^2}$$

Figure 12.3 shows $\frac{d\bar{C}(x|\theta)}{dx}$ versus the bid x, showing the behavior that it starts positive and transitions to negative. The point where it is equal to zero would be the optimal bid, a point which can be found numerically quite easily. Let

$$X^{explt}(S^n) = \text{A pure exploitation policy which chooses the optimal bid to maximize the single-period profit using the belief distribution } p^n \text{ for } \theta.$$

12.4.2 An excitation policy

A potential limitation of our pure exploitation policy is that it ignores the value of trying a wider range of bids to help with the process of learning the correct values of θ. A popular strategy is to add a noise term, known in engineering as "excitation," giving us the policy

$$X^{excite}(S^n|\rho) = X^{explt}(S^n) + \varepsilon(\rho)$$

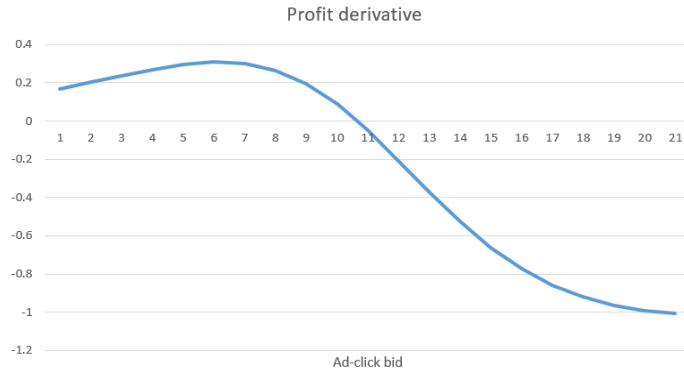


Figure 12.3: Derivative of ad-click profit function versus the bid.

where $\varepsilon(\rho) \sim N(0, \rho^2)$. In this policy, ρ is our tunable parameter which controls the amount of exploration in the policy. If it is too small, then there may not be enough exploration. If it is too large, then we will be choosing bids that are far from optimal, possibly without any benefit from learning.

12.4.3 A value of information policy

The pure exploitation and the excitation policies we just introduced are both relatively simple. Now we are going to consider a policy that maximizes the value of information in the future. This seems like a reasonable idea, but it requires that we think about how information now affects what decision we *might* make in the future, and this will be a bit more difficult.

Our exploitation policy assumes that the estimated parameters θ^n after n experiments is the correct value, and chooses a bid based on this estimate. Now imagine that we bid $x^n = x$ and observe K^{n+1} and $\hat{R}^{n+1}$, and use this information to get an updated estimate of θ^{n+1} as well as $\bar{R}^{n+1}$. We can then use these updated estimates to make a better decision. We want to choose the bid x that gives us the greatest improvement in the quality of a decision, recognizing that we do not know the outcome of $W^{n+1} = (\hat{R}^{n+1}, K^{n+1})$ until we actually place the bid.

Let $\theta^{n+1}(x|W^{n+1}$ be the updated estimate of θ assuming we bid $x^n = x$ and observe $W^{n+1} = (\hat{R}^{n+1}, K^{n+1})$. This is a random variable, because we are thinking about placing a bid $x^n = x$ for the $n+1st$ auction, but we have not yet placed the bid, which means we have not yet observed W^{n+1}.

To simplify our analysis, we are going to assume that the random variable $K^{n+1} = 1$ with probability $P^{click}(x|\theta)$ and $K^{n+1} = 0$ with probability $1-P^{click}(x|\theta)$. We are then going to assume that $\bar{R}^{n+1} = \bar{R}^n$, which is probably a fairly accurate approximation if we have observed enough auctions to get a good estimate of the revenue we will receive if a customer clicks on our ad.

We can think of this as an approximate lookahead model, where $\bar{R}^n$ does not change. We would then write our exogenous information in our lookahead model as

$$\tilde{W}^{n,n+1} = \tilde{K}^{n,n+1},$$

where the double-superscript $(n, n+1)$ means that this is the information in a lookahead model created at time n, looking at what might happen at time $n+1$. The random variable $\tilde{K}^{n,n+1}$ is the ad-click that we are simulating *might* happen in our lookahead model, rather than the actual observation of whether someone clicked on the ad.

We next use our updating equation (12.3) for the probabilities $p_k^n = Prob[\theta = \theta_k|H^n]$. We can write these updated probabilities as $p_k^{n+1}(K^{n+1})$ to capture the dependence of the updating on K^{n+1} (equation (12.3) is written for $K^{n+1} = 1$). Since K^{n+1} can take on two outcomes (0 or 1) we will have two possible values for $p_k^{n+1}(K^{n+1})$.

Now imagine that we perform our pure exploitation policy $X^{explt}(S^n|\theta^n)$ that we described above, but we are going to do it in our approximate lookahead model (this is where we ignore changes in $\bar{R}^n$). Let $\tilde{S}^{n,n+1}$ represent our state in the lookahead model given by

$$\tilde{S}^{n,n+1}(K^{n+1}) = (\bar{R}^n, p^{n+1}(K^{n+1})).$$

Remember - since $K^{n,n+1}$ is a random variable (we are still at time n), $\tilde{S}^{n,n+1}(K^{n+1})$ is also a random variable, which is why we write its explicit dependence on the outcome K^{n+1}.

The way to think about this lookahead model is as if you are playing a game (such as chess) where you think about a move (for us, that would be the bid x^n) and then, before you make the move, think about what might happen in the future. In this problem, our future only has two outcomes (whether or not a customer clicks on the ad), which means two possible values of $\tilde{S}^{n,n+1}$, which produces two sets of updated probabilities $p^{n+1}(K^{n+1})$.

Finally, this means that there will be two values of the optimal myopic bid (using our pure exploitation policy) $X^{explt}(\tilde{S}^{n,n+1})$. The expected contribution we would make in the future is then given by $\bar{C}(\tilde{S}^{n,n+1}, \tilde{x}^{n,n+1})$ where $\tilde{x}^{n,n+1}$ (this is the decision we are thinking of making in the future) is given by

$$\tilde{x}^{n,n+1} = X^{explt}(\tilde{S}^{n,n+1})).$$

This means there are two possible optimal decisions, which means two different values of the expected contribution $\bar{C}(\tilde{S}^{n,n+1}, X^{explt}(\tilde{S}^{n,n+1}))$. For compactness, let's call these $\tilde{C}^{n,n+1}(1)$ (if $K^{n+1} = 1$) and $\tilde{C}^{n,n+1}(0)$ (if $K^{n+1} = 0$). Think of these as the expected contributions that *might* happen in the future given what we know now. Finally we can take the expectation over K^{n+1} to obtain the expected contribution of placing a bid $x^n = x$ right now, which we can compute using

$$\tilde{C}^n(x) = \sum_{k=1}^{K} \left(P^{click}(x|\theta = \theta_k)\tilde{C}^{n,n+1}(1) + (1 - P^{click}(x|\theta = \theta_k))\tilde{C}^{n,n+1}(0)\right)p_k^n.$$

Our policy, then, is to pick the bid x that maximizes $\tilde{C}^n(x)$. Assume that we discretize our bids into a set $\mathcal{X} = \{x_1, \ldots, x_M\}$. Our value of information policy would be written

$$X^{VoI}(S^n) = \arg\max_{x \in \mathcal{X}} \tilde{C}^n(x).$$

Value of information policies are quite powerful, but they are clearly harder to compute. Imagine, for example, doing this computation when there is more than two outcomes. For example, if we had not made our simplification of holding $\bar{R}^n$ constant, we would have to recognize that this state variable is also changing.

We note only in passing that we have run many comparisons of different learning policies, and the one-step lookahead value of information

often works quite well. Care has to be used, however, for these problems where the outcome is 0 or 1. We used this setting because it made the derivations much simpler.

However, learning problems where the outcome is 0 or 1 are problems where a single experiment provides very little information. Instead, it is better to assume that we our going to make our decision (that is, set the bid) and then observe it for, say, M auctions. This means that K^{n+1} might now be a number between 0 and M. The number M becomes a tunable parameter, and the calculations just became (we have to sum over $M+1$ realizations rather than just two), but this approach can work quite well.

12.5 Policy evaluation

12.6 Extensions

12.6.1 Customers with simple attributes

Assume that we know the location of a customer down to a region or the nearest major city, which we designate by L. If we think that the behavior of each region is different, we could index θ by θ_ℓ if the customer is from location $L = \ell$. This means that if there are 1,000 locations, then we have to estimate 1,000 models, which means 1,000 values of $\theta = (\theta^{const}, \theta^{bid})$.

An alternative approach would be to specify a model of the form

$$Prob^n[K^{n+1} = 1|\theta] = \frac{e^{U(x,L|\theta}}{1 + e^{U(x,L|\theta}}. \tag{12.11}$$

where we are now going to use as our utility

$$U(x, L|\theta) = \theta^{const} + \theta^{bid}x + \sum_{\ell=1}^{L} \theta_\ell^{loc} I_{\ell=L}.$$

This is a more compact model because we now assume that the constant term θ^{const} and bid coefficient θ^{bid} do not depend on the location. Instead, we are just adding a shift θ_ℓ^{loc}. So, we still have 1,000 parameters to estimate (the location coefficients), but before we had 2,000 parameters to estimate - θ_ℓ^{const} and θ_ℓ^{bid} for each location $\ell \in \{1, \ldots, L\}$.

13

Blood management problem

13.1 Narrative

The problem of managing blood inventories serves as a particularly elegant illustration of a resource allocation problem. We are going to start by assuming that we are managing inventories at a single hospital, where each week we have to decide which of our blood inventories should be used for the demands that need to be served in the upcoming week.

We have to start with a bit of background about blood. For the purposes of managing blood inventories, we care primarily about blood type and age. Although there is a vast range of differences in the blood of two individuals, for most purposes doctors focus on the eight major blood types: $A+$ (“ A positive”), $A-$ (“A negative”), $B+$, $B-$, $AB+$, $AB-$, $O+$, and $O-$. While the ability to substitute different blood types can depend on the nature of the operation, for most purposes blood can be substituted according to table 13.1.

A second important characteristic of blood is its age. The storage of blood is limited to six weeks, after which it has to be discarded. Hospitals need to anticipate if they think they can use blood before it hits this limit, as it can be transferred to blood centers which monitor inventories at different hospitals within a region. It helps if a hospital

	Recipient							
Donor	$AB+$	$AB-$	$A+$	$A-$	$B+$	$B-$	$O+$	$O-$
$AB+$	X							
$AB-$	X	X						
$A+$	X		X					
$A-$	X	X	X	X				
$B+$	X				X			
$B-$	X	X			X	X		
$O+$	X		X		X		X	
$O-$	X	X	X	X	X	X	X	X

Table 13.1: Allowable blood substitutions for most operations, 'X' means a substitution is allowed.

can identify blood it will not need as soon as possible so that the blood can be transferred to locations that are running short.

One mechanism for extending the shelf-life of blood is to freeze it. Frozen blood can be stored up to 10 years, but it takes at least an hour to thaw, limiting its use in emergency situations or operations where the amount of blood needed is highly uncertain. In addition, once frozen blood is thawed it must be used within 24 hours.

13.2 Basic model

13.2.1 State variables

We can model the blood problem as a heterogeneous resource allocation problem. We are going to start with a fairly basic model which can be easily extended with almost no notational changes. We begin by describing the attributes of a unit of stored blood using

$$b = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} = \begin{pmatrix} \text{Blood type } (A+,\, A-,\, \dots) \\ \text{Age (in weeks)} \end{pmatrix},$$

$$\mathcal{B} = \text{Set of all attribute types.}$$

We will limit the age to the range $0 \leq a_2 \leq 6$. Blood with $a_2 = 6$ (which means blood that is already six weeks old) is no longer usable. We assume that decision epochs are made in one-week increments.

Blood inventories, and blood donations, are represented using

$$\begin{aligned} R_{tb} &= \text{Units of blood of type } b \text{ available to be assigned or held at time } t, \\ R_t &= (R_{tb})_{b \in \mathcal{B}}, \end{aligned}$$

The pre- and post-decision state variables are given by

$$\begin{aligned} S_t &= (R_t, \hat{D}_t), \\ S_t^x &= (R_t^x). \end{aligned}$$

13.2.2 Decision variables

We act on blood resources with decisions given by:

$$\begin{aligned} x_{tbd} &= \text{Number of units of blood with attribute } b \text{ that we assign to a demand of type } d, \\ x_t &= (x_{tad})_{b \in \mathcal{B}, d \in \mathcal{D}}. \end{aligned}$$

The feasible region $\mathcal{X}_t$ is defined by the following constraints:

$$\sum_{d \in \mathcal{D}} x_{tbd} = R_{tb}, \tag{13.1}$$

$$\sum_{b \in \mathcal{B}} x_{tbd} \leq \hat{D}_{td}, \qquad d \in \mathcal{D}, \tag{13.2}$$

$$x_{tbd} \geq 0. \tag{13.3}$$

13.2.3 Exogenous information

$$\begin{aligned} \hat{R}_{tb} &= \text{Number of new units of blood of type } b \text{ donated between } t-1 \text{ and } t, \\ \hat{R}_t &= (\hat{R}_{tb})_{b \in \mathcal{B}}. \end{aligned}$$

The attributes of demand for blood are given by

$$d = \begin{pmatrix} d_1 \\ d_2 \\ d_3 \end{pmatrix} = \begin{pmatrix} \text{Blood type of patient} \\ \text{Surgery type: urgent or elective} \\ \text{Is substitution allowed?} \end{pmatrix},$$

$$\begin{aligned} d^\phi &= \text{Decision to hold blood in inventory (“do nothing”),} \\ \mathcal{D} &= \text{Set of all demand types } d \text{ plus } d^\phi. \end{aligned}$$

The attribute d_3 captures the fact that there are some operations where a doctor will not allow any substitution. One example is childbirth, since infants may not be able to handle a different blood type, even if it is an allowable substitute. For our basic model, we do not allow unserved demand in one week to be held to a later week.

The new demands for blood are modeled using

$$\begin{aligned} \hat{D}_{td} &= \text{Units of demand with attribute } d \text{ that arose between } t-1 \text{ and } t, \\ \hat{D}_t &= (\hat{D}_{td})_{d \in \mathcal{D}}. \end{aligned}$$

13.2.4 Transition function

Blood that is held simply ages one week, but we limit the age to six weeks. Blood that is assigned to satisfy a demand can be modeled as being moved to a blood-type sink, denoted, perhaps, using $b_{t,1} = \phi$ (the null blood type). The blood attribute transition function $r^M(b_t, d_t)$ is given by

$$b_{t+1} = \begin{pmatrix} b_{t+1,1} \\ b_{t+1,2} \end{pmatrix} = \begin{cases} \begin{pmatrix} b_{t,1} \\ \min\{6, b_{t,2}+1\} \end{pmatrix}, & d_t = d^\phi, \\ \begin{pmatrix} \phi \\ \text{-} \end{pmatrix}, & d_t \in \mathcal{D}. \end{cases}$$

To represent the transition function, it is useful to define

$$\begin{aligned} \delta_{b'}(b,d) &= \begin{cases} 1 & b_t^x = b' = r^M(b_t, d_t), \\ 0 & \text{otherwise}, \end{cases} \\ \Delta &= \text{Matrix with } \delta_{b'}(b,d) \text{ in row } b' \text{ and column } (b,d). \end{aligned}$$

We note that the attribute transition function is deterministic. A random element would arise, for example, if inspections of the blood resulted in blood that was less than six weeks old being judged to have expired. The resource transition function can now be written

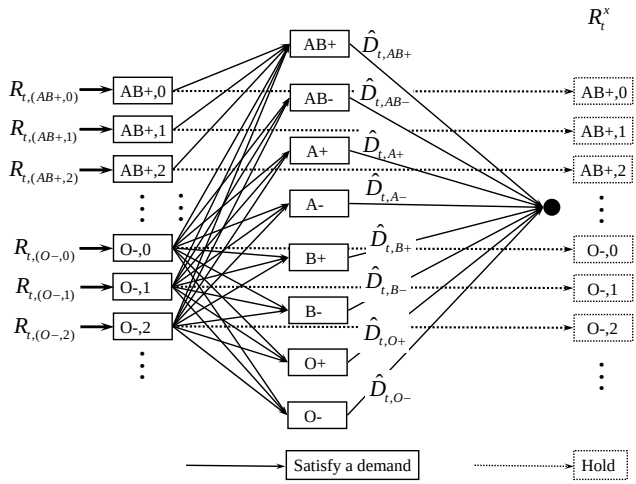
$$\begin{aligned} R^x_{tb'} &= \sum_{b\in\mathcal{B}}\sum_{d\in\mathcal{D}} \delta_{b'}(b,d)x_{tbd}, \\ R_{t+1,b'} &= R^x_{tb'} + \hat{R}_{t+1,b'}. \end{aligned}$$

In matrix form, these would be written

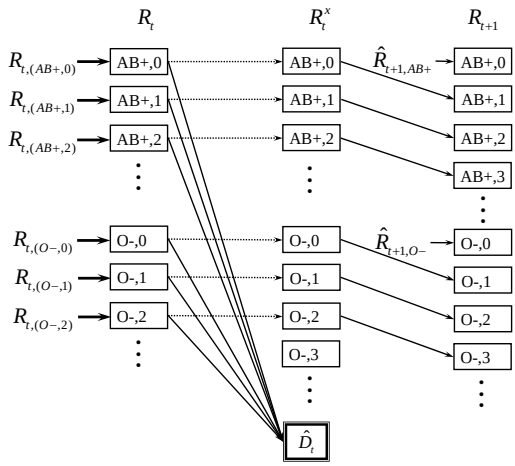
$$R^x_t = \Delta x_t, \tag{13.4}$$

$$R_{t+1} = R^x_t + \hat{R}_{t+1}. \tag{13.5}$$

Figure 13.1 illustrates the transitions that are occurring in week t. We either have to decide which type of blood to use to satisfy a demand (figure 13.1a), or to hold the blood until the following week. If we use blood to satisfy a demand, it is assumed lost from the system. If we hold the blood until the following week, it is transformed into blood that is one week older. Blood that is six weeks old may not be used to satisfy any demands, so we can view the bucket of blood that is six weeks old as a sink for unusable blood (the value of this blood would be zero). Note that blood donations are assumed to arrive with an age of 0.



13.1a - Assigning blood supplies to demands in week t. Solid lines represent assigning blood to a demand, dotted lines represent holding blood.



13.1b - Holding blood supplies until week $t+1$.

Figure 13.1: The assignment of different blood types (and ages) to known demands in week t (13.1a), and holding blood until the following week (13.1b).

Condition	Description	Value
if $d = d^{\phi}$	Holding	0
if $b_1 = b_1$ when $d \in \mathcal{D}$	No substitution	0
if $b_1 \neq b_1$ when $d \in \mathcal{D}$	Substitution	-10
if $b_1 =$ O- when $d \in \mathcal{D}$	O- substitution	5
if $d_2 =$ Urgent	Filling urgent demand	40
if $d_2 =$ Elective	Filling elective demand	20

Table 13.2: Contributions for different types of blood and decisions

13.2.5 Objective function

There is no real "cost" to assigning blood of one type to demand of another type (we are not considering steps such as spending money to encourage additional donations, or transporting inventories from one hospital to another). Instead, we use the contribution function to capture the preferences of the doctor. We would like to capture the natural preference that it is generally better not to substitute, and that satisfying an urgent demand is more important than an elective demand. For example, we might use the contributions described in table 13.2. Thus, if we use $O-$ blood to satisfy the needs for an elective patient with $A+$ blood, we would pick up a -\$10 contribution (penalty since it is negative) for substituting blood, a +\$5 for using $O-$ blood (something the hospitals like to encourage), and a +\$20 contribution for serving an elective demand, for a total contribution of +\$15.

The total contribution (at time t) is finally given by

$$C_t(S_t, x_t) = \sum_{b \in \mathcal{B}} \sum_{d \in \mathcal{D}} c_{tbd} x_{tbd}.$$

As before, let $X_t^{\pi}(S_t)$ be a policy (some sort of decision rule) that determines $x_t \in \mathcal{X}_t$ given S_t. We wish to find the best policy by solving

$$\max_{\pi \in \Pi} \mathbb{E} \sum_{t=0}^{T} \gamma^t C_t(S_t, x_t). \tag{13.6}$$

13.3 Modeling uncertainty

13.4 Designing policies

The most obvious way to solve this problem is with a simple myopic policy, where we maximize the contribution at each point in time without regard to the effect of our decisions on the future. We can obtain a family of myopic policies by adjusting the one-period contributions. For example, our bonus of \$5 for using $O-$ blood (in table 13.2), is actually a type of myopic policy. We encourage using $O-$ blood since it is generally more available than other blood types. By changing this bonus, we obtain different types of myopic policies that we can represent by the set Π^M, where for $\pi \in \Pi^M$ our decision function would be given by

$$X_t^\pi(S_t) = \arg\max_{x_t \in \mathcal{X}_t} \sum_{b \in \mathcal{B}} \sum_{d \in \mathcal{D}} c_{tbd} x_{tbd}. \tag{13.7}$$

The optimization problem in (13.7) is a simple linear program (known as a "transportation problem"). Solving the optimization problem given by equation (13.6) for the set $\pi \in \Pi^M$ means searching over different values of the bonus for using $O-$ blood.

13.4.1 An ADP algorithm

As a traditional dynamic program, the optimization problem posed in equation (13.6) is quite daunting. The state variable S_t has $|\mathcal{A}| + |\mathcal{B}| = 8 \times 6 + 8 \times 2 \times 2 = 80$ dimensions. The random variables $\hat{R}$ and $\hat{D}$ also have a combined 80 dimensions. The decision vector x_t has $27 + 8 = 35$ dimensions.

It is natural to use approximate value iteration to determine the allocation vector x_t using

$$x_t^n = \arg\max_{x_t \in \mathcal{X}_t^n} \left(C_t(S_t^n, x_t) + \overline{V}_t^{n-1}(R_t^x)\right), \tag{13.8}$$

where $R_t^x = R^M(R_t, x_t)$ is given by equation (13.4). The first (and most important) challenge we face is identifying an appropriate approximation strategy for $\overline{V}_t^{n-1}(R_t^x)$. A simple and effective approximation is to use

separable, piecewise linear approximations, which is to say

$$\overline{V}_t(R_t^x) = \sum_{b \in \mathcal{B}} \overline{V}_{tb}(R_{tb}^x),$$

where $\overline{V}_{tb}(R_{tb}^x)$ is a scalar, piecewise, linear function. It is easy to show that the value function is concave (as well as piecewise linear), so each $\overline{V}_{tb}(R_{tb}^x)$ should also be concave. Without loss of generality, we can assume that $\overline{V}_{tb}(R_{tb}^x) = 0$ for $R_{tb}^x = 0$, which means the function is completely characterized by its set of slopes. We can write the function using

$$\overline{V}_{tb}^{n-1}(R_{tb}^x) = \left(\sum_{r=1}^{\lfloor R_{tb}^x \rfloor} \bar{v}_{tb}^{n-1}(r-1) + (R_{tb}^x - \lfloor R_{tb}^x \rfloor)\bar{v}_{tb}^{n-1}(\lfloor R_{tb}^x \rfloor) \right) \quad (13.9)$$

where $\lfloor R \rfloor$ is the largest integer less than or equal to R. As we can see, this function is determined by the set of slopes $(\bar{v}_{tb}^{n-1}(r))$ for $r = 0, 1, \ldots, R^{max}$, where R^{max} is an upper bound on the number of resources of a particular type.

Assuming we can estimate this function, the optimization problem that we have to solve (equation (13.8)) is the fairly modest linear program shown in figure 13.2. As with figure 13.1, we have to consider both the assignment of different types of blood to different types of demand, and the decision to hold blood. To simplify the figure, we have collapsed the network of different demand types into a single aggregate box with demand $\hat{D}_t$. This network would actually look just like the network in figure 13.1a. The decision to hold blood has to consider the value of a type of blood (including its age) in the future, which we are approximating using separable, piecewise linear value functions. Here, we use a standard modeling trick that converts the separable, piecewise linear value function approximations into a series of parallel links from each node representing an element of R_t^x into a supersink. Piecewise linear functions are not only easy to solve (we just need access to a linear programming solver), they are easy to estimate. In addition, for many problem classes (but not all) they have been found to produce very fast convergence with high quality solutions.

With this decision function, we would use approximate value iteration, following the trajectory determined by the policy. Aside from the

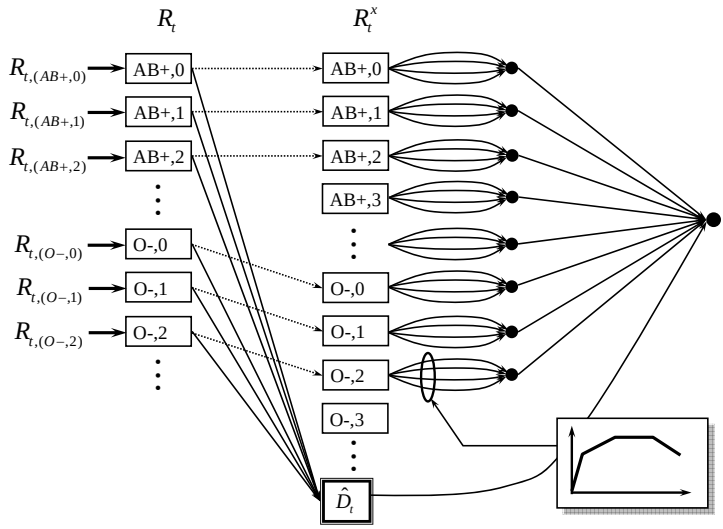


Figure 13.2: Network model for time t with separable, piecewise linear value function approximations.

customization of the value function approximation, the biggest difference is in how the value functions are updated, a problem that is handled in the next section. Of course we could use other nonlinear approximation strategies, but this would mean that we would have to solve a nonlinear programming problem instead of a linear program. While this can, of course, be done, it complicates the strategy for updating the value function approximation.

For most operational applications, this problem would be solved over a finite horizon (say, 10 weeks) giving us a recommendation of what to do right now. Alternatively, we can solve an infinite horizon version of the model by simply dropping the t index on the value function approximation.

13.4.2 Updating the value function approximation

Since our value function depends on slopes, we want to use derivatives to update the value function approximation. Let

$$\tilde{V}_t(S_t) = \max_{x_t \in \mathcal{X}_t^n} \left(C_t(S_t^n, x_t) + \overline{V}_t^{n-1}(R_t^x)\right)$$

be the value of our decision function at time t. But what we want is the derivative with respect to the previous post-decision state R_{t-1}^x. Recall that $S_t = (R_t, D_t)$ depends on $R_t = R_{t-1}^x + \hat{R}_t$. We want to find the gradient $\nabla \tilde{V}_t(S_t)$ with respect to R_{t-1}^x. We apply the chain rule to find

$$\frac{\partial \tilde{V}_t(S_t)}{\partial R_{t-1,a}^x} = \sum_{a' \in \mathcal{A}} \frac{\partial \tilde{V}_t(S_t)}{\partial R_{ta'}} \frac{\partial R_{ta'}}{\partial R_{t-1,a}^x}.$$

If we decide to hold an additional unit of blood of type a_{t-1}, then this produces an additional unit of blood of type $a_t = a^M(a_{t-1}, d)$ where $a^M(a, d)$ is our attribute transition function and d is our decision to hold the blood until the next week. For this application, the only difference between a_{t-1} and a_t is that the blood has aged by one week. We also have to model the fact that there may be exogenous changes to our blood supply (donations, deliveries from other hospitals, as well as blood that has gone bad). This means that

$$R_{ta_t} = R_{t-1,a_{t-1}}^x + \hat{R}_{ta_t}.$$

Note that we start with $R_{t-1,a_{t-1}}^x$ units of blood with attribute a, and then add the exogenous changes $\hat{R}_{ta_t}$ to blood of type a_t. This allows us to compute the derivative of R_t with respect to R_{t-1}^x using

$$\frac{\partial R_{ta_t}}{\partial R_{t-1,a_{t-1}}^x} = \begin{cases} 1 & \text{if } a_t = a^M(a_{t-1}, d) \\ 0 & \text{otherwise.} \end{cases}$$

This allows us to write the derivative as

$$\frac{\partial \tilde{V}_t(S_t)}{\partial R_{t-1,a_{t-1}}^x} = \frac{\partial \tilde{V}_t(S_t)}{\partial R_{ta_t}}.$$

$\frac{\partial \tilde{V}_t(S_t)}{\partial R_{ta}}$ can be found quite easily. A natural byproduct from solving the linear program in equation (13.8) is that we obtain a dual variable

$\hat{v}^n_{ta}$ for each flow conservation constraint (13.1). This is a significant advantage over strategies where we visit state R_t (or S_t) and then update just the value of being in that single state. The dual variable is an estimate of the slope of the decision problem with respect to R_{ta} at the point R^n_{ta}, which gives us

$$\left.\frac{\partial \tilde{V}_t(S_t)}{\partial R^x_{t-1,a_{t-1}}}\right|_{R^{x,n}_{t-1,a_{t-1}}} = \hat{v}^n_{ta_t}.$$

Dual variables are incredibly convenient for estimating value function approximations in the context of resource allocation problems, both because they approximate the derivatives with respect to R_{ta} (which is all we are interested in), but also because we obtain an entire vector of slopes, rather than a single estimate of the value of being in a state. However, it is important to understand exactly which slope the dual is (or is not) giving us.

Figure 13.3 illustrates a nondifferentiable function (any problem that can be modeled as a sequence of linear programs has this piecewise linear shape). If we are estimating the slope of a piecewise linear function $V_{ta}(R_{ta})$ at a point $R_{ta} = R^n_{ta}$ that corresponds to one of these kinks (which is quite common), then we have a left slope, $\hat{v}^-_{ta} = V_{ta}(R_{ta}) - V_{ta}(R_{ta} - 1)$, and a right slope, $\hat{v}^+_{ta} = V_{ta}(R_{ta} + 1) - V_{ta}(R_{ta})$. If $\hat{v}^n_{ta}$ is our dual variable, we have no way of specifying whether we want the left or right slope. In fact, the dual variable can be anywhere in the range $\hat{v}^+_{ta} \leq \hat{v}^n_{ta} \leq \hat{v}^-_{ta}$.

If we are comfortable with the approximation implicit in the dual variables, then we get the vector $(\hat{v}^n_{ta})_{a\in\mathcal{A}}$ for free from our simplex algorithm. If we specifically want the left or right derivative, then we can perform a numerical derivative by perturbing each element R^n_{ta} by plus or minus 1 and reoptimizing. This sounds clumsy when we get dual variables for free, but it is surprisingly fast.

Once we have computed $\hat{v}^n_{ta}$ for each $a \in \mathcal{A}$, we now have to update our value function approximation. Remember that we have to update the value function at time $t - 1$ at the previous post-decision state. For this problem, attributes evolve deterministically (for example, AB+ blood that is 3 weeks old becomes AB+ blood that is 4 weeks old).

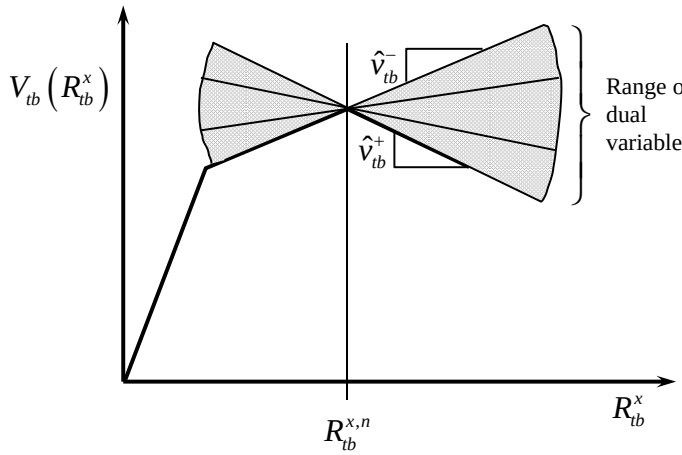


Figure 13.3: Illustration of range of dual variables for a nondifferentiable function.

Let $a_t = a^M(a_{t-1}, d)$ describe this evolution (here, d is the decision to hold the blood). Let $\hat{v}^n_{ta_t}$ be the attribute corresponding to blood with attribute a_t at time t, and assume that $R^{x,n}_{t-1,a_{t-1}}$ was the amount of blood of type a_{t-1} at time $t-1$. Let $\bar{v}^{n-1}_{t-1,a_{t-1}}(r)$ be the slope corresponding to the interval $r \leq R^{x,n}_{t-1,a_{t-1}} \leq r+1$ (as we did in (13.9)). A simple update is to use

$$\bar{v}^n_{t-1,a_{t-1}}(r) = \begin{cases} (1-\alpha_{n-1})\bar{v}^{n-1}_{t-1,a_{t-1}}(r) + \alpha_{n-1}\hat{v}^n_{ta_t} & \text{if } r = R^{x,n}_{t-1,a_{t-1}} \\ \bar{v}^{n-1}_{t-1,a_{t-1}}(r) & \text{otherwise.} \end{cases} \tag{13.10}$$

Here, we are only updating the element $\bar{v}^{n-1}_{t-1,a_{t-1}}(r)$ corresponding to $r = R^{x,n}_{t-1,a_{t-1}}$. We have found that it is much better to update a range of slopes, say in the interval $(r-\delta, r+\delta)$ where δ has been chosen to reflect the range of possible values of r. For example, it is best if δ is chosen initially so that we are updating the entire range, after which we periodically cut it in half until it shrinks to 1 (or an interval small enough that we are getting the precision we want in the decisions).

There is one problem with equation (13.10) that we have already seen before. We know that the value functions are concave, which means that we should have $\bar{v}^n_{ta}(r) \geq \bar{v}^n_{ta}(r+1)$ for all r. This might be true for $\bar{v}^{n-1}_{ta}(r)$, but it is entirely possible that after our update, it is no longer

true for $\bar{v}_{ta}^{n}(r)$. To handle this, we have to use one of our methods for maintaining concavity, which we describe in section 13.4.3.

The use of separable, piecewise linear approximations has proven effective in a number of applications, but there are open theoretical questions (how good is the approximation?) as well as unresolved computational issues (what is the best way to discretize functions?). What about the use of low-dimensional basis functions? If we use a continuously differentiable approximation (which requires the use of a nonlinear programming algorithm), we can use our regression techniques to fit the parameters of a statistical model that is continuous in the resource variable.

The best value function approximation does not just depend on the structure of the problem. It depends on the nature of the data itself.

13.4.3 Estimating concave, piecewise linear functions

13.5 Policy evaluation

13.6 Extensions

We assume that any demands that are not satisfied at time t are lost. Imagine that we have emergency surgeries that either have to be satisfied, and elective surgeries that can be delayed to a later time period.

1) Write out the state variable for the new problem.

2) Can we still use the value function approximation when demands were not held?

3) Consider using a value function approximation that is piecewise linear and separable in the blood inventories (this is the VFA suggested above), along with piecewise linear and separable VFAs for the amount of held demand (by blood type). We use the dual variables for the blood inventory to update the VFA for blood supplies. How might we update the VFA for the held demand?

4) What are the errors introduced when using a separable VFA for blood supplies and demands?

5) Instead of modeling the problem in single week increments, model daily decisions.

6) Include the presence of blood that has been frozen, and the decision to freeze blood.

7) A hospital might require weekly deliveries of blood from a community blood bank to make up for systematic shortages. Imagine that a fixed quantity (e.g., 100 units) of blood arrive each week, but where the amount of blood of each type and age (the blood may have already been held in inventory for several weeks) might be random.

8) We presented a model that focused only on blood inventories at a single hospital. We can handle multiple hospitals and distribution centers by simply adding a location attribute, and providing for a decision to move blood (at a cost) from one location to another.

This model can also be applied to any multiproduct inventory problem where there are different types of product and different types of demands, as long as we have the ability to choose which type of product is assigned to each type of demand. We also assume that products are not reusable; once the product is assigned to a demand, it is lost from the system.

14

Optimizing clinical trials

14.1 Narrative

Pharmaceutical companies run thousands of clinical trials each year testing new medications. Drug testing occurs in three phases:

Phase I These are tests run on 10 to 30 volunteers over a few months to determine the dose, identify side effects and perform an initial assessment of drug response and side effects.

Phase II These are larger trials that might involve over 100 patients spanning two years. The trial will typically involve comparing a group of patients with the standard treatment against one or two groups with new treatments.

Phase III These are trials involving hundreds of patients spanning multiple years to assess effectiveness and safety.

Both Phase II and Phase III trials require identifying patients with the proper characteristics to enter the trial, at which point they are randomly assigned to one of the groups for comparison.

In our exercise, we assume that we enroll a set of hospitals and clinics each week to achieve a *potential* population, from which patients

will be identified as candidates for the trial based on paper records. There is an up-front administrative cost to enroll a hospital or clinic in the trial. The administrative cost reflects the pool of patients that the facility might have for the study. For example, it might cost $250,000 to sign up a group of hospitals and clinics with a total potential population of 500 patients. In our model, we will simply set a $500 per patient signup cost, keeping in mind that this is for a total potential population, from which we draw actual signups.

Once we have signed up a facility, we then advertise the clinical trial from which patients (or their physicians) step forward. At that point, a patient is then subjected to a more detailed evaluation, which determines who is accepted into the trial. Ineligible patients are then dropped.

For the purpose of this exercise, we are going to assume that each patient is given a medication at the beginning of the week. By the end of the week, we know if the patient is responding or not. Patients that respond are designated as successes, the rest as failures. Each week, then, requires an entirely new set of patients, but these are drawn from the base population that we have signed up. We can only increase this population by entering more capacity into the trial, and paying the up-front administrative cost.

14.2 Basic model

We are going to assume that we make decisions at the end of each week t, to be implemented during week $t + 1$. We let time t denote the end of week t.

14.2.1 State variables

We have the following state variables:

$$\begin{aligned} R_t &= \text{The potential population of patients that are in the hospitals and clinics that have been signed up,} \\ \alpha_t &= \text{The number of successes for the treatment by week } t \text{ over the course of the clinical trial,} \\ \beta_t &= \text{The number of failures for the treatment by week } t. \\ \bar{\lambda}_t^{response} &= \text{Estimated fraction of potential patients who elect to join the trial given what we know at time } t. \end{aligned}$$

Using this information, we can estimate the probability that our treatment is successful using

$$\begin{aligned} \rho_t &= \text{The probability that the treatment is successful given what we know by the end of week } t, \\ &= \frac{\alpha_t}{\alpha_t + \beta_t}. \end{aligned}$$

This means that our state variable would be

$$S^n = (R_t, (\alpha_t, \beta_t), \bar{\lambda}_t^{response}).$$

For the initial state, it is reasonable to assume that $R_0 = 0$, but we may wish to start with an initial estimate of the probability of success (based on prior clinical trials) which we can represent by assuming nonzero values for α_0 and β_0.

14.2.2 Decision variables

We model the number of potential patients that are signed up using

$$x_t^{enroll} = \text{The number of new potential patients that are signed up at time } t \text{ (we can think of this as at the end of week } t, \text{ to be implemented for week } t+1.$$

We also have the decision of when to stop the trial represented by

$$x_t^{trial} = \begin{cases} 1 & \text{Continue the trial} \\ 0 & \text{Stop the trial} \end{cases}$$

If $x_t^{trial} = 0$, then we are going to set $R_{t+1} = 0$, which shuts down the trial. We assume that once we have stopped the trial, we cannot restart it, which means we will require that

$$x_t^{trial} = 0 \text{ If } R_t = 0.$$

If we stop the trial, we have to declare whether the drug is a success or a failure,

$$x_t^{drug} = \begin{cases} 1 & \text{If the drug is declared a success} \\ 0 & \text{If the drug is declared a failure} \end{cases}$$

We will create policies $X^{\pi^{enroll}}(S_t)$, $X^{\pi^{trial}}(S_t)$ and $X^{\pi^{drug}}(S_t)$ which determine x_t^{enroll}, x_t^{trial} and x_t^{drug}. We can then write

$$X^{\pi}(S_t) = (X^{\pi^{enroll}}(S_t), X^{\pi^{trial}}(S_t), X^{\pi^{drug}}(S_t)).$$

14.2.3 Exogenous information

We first identify new patients and patient withdrawals to and from the trial using

$$\hat{R}_{t+1}(x_t^{enroll}) = \text{The number of new patients joining the trial during week } t+1, \text{ which depends on the number of potential patients } x_t^{enroll}.$$

We next track our successes with

$$\begin{aligned} \hat{X}_t &= \text{The number of successes during week } t, \\ \hat{Y}_t &= \text{The number of failures during week } t. \end{aligned}$$

The number of failures during week t can be calculated as

$$\hat{Y}_t = \hat{R}_t - \hat{X}_t.$$

These variables depend on the number of patients R_t in the system at the end of week t. As always, we defer to the section on uncertainty modeling the development of the underlying probability models for these random variables.

Our exogenous information process is then

$$W_t = (\hat{R}_t, \hat{X}_t),$$

where we exclude $\hat{Y}_t$ because it can be computed from the other variables.

14.2.4 Transition function

The transition equation for the number of enrolled patients is given by

$$R_{t+1} = x_t^{trial}(R_t + x_t^{enroll}). \tag{14.1}$$

We update the probability the drug is a success by counting the number of successes and failures using

$$\alpha_{t+1} = \alpha_t + \hat{X}_{t+1}, \tag{14.2}$$
$$\beta_{t+1} = \beta_t + (\hat{R}_{t+1} - \hat{X}_{t+1}). \tag{14.3}$$

Finally, we update our estimate of the number of patients who enroll in the trial by smoothing the current estimate $\bar{\lambda}_t^{response}$ with the latest ratio of the number who enrolled, $\hat{R}_t$, and the number who are currently enrolled in the trial.

$$\bar{\lambda}_{t+1}^{response} = (1-\eta)\bar{\lambda}_t^{response} + \eta \frac{\hat{R}_t}{R_t + x_t^{enroll}}. \tag{14.4}$$

Equations (14.1) - (14.4) make up the transition function that we represent generically using

$$S_{t+1} = S^M(S_t, x_t, W_{t+1}).$$

14.2.5 Objective function

We have to consider the following costs:

$$\begin{aligned} c^{enroll} &= \text{The cost of maintaining a patient in the trial per time period,} \\ c^{trial} &= \text{The ongoing administrative overhead, costs of keeping the trial going (this stops when we stop testing)} \\ p^{success} &= \text{The (large) revenue gained if we stop and declare success, which typically means selling the patient to a manufacturer.} \end{aligned}$$

The profit (contribution) in a time period would then be given by

$$C(S_t, x_t) = (1 - x_t^{trial}) x_t^{drug} p^{success} - x_t^{trial}(c^{trial} + c^{enroll} x_t^{enroll}). \tag{14.5}$$

Our objective function, then, would be our canonical objective which we state as

$$\max_{\pi} \mathbb{E}\left\{\sum_{t=0}^{T} C(S_t, X^{\pi}(S_t))|S_0\right\}, \tag{14.6}$$

where we recognize that our policy is a composition of the patient enrollment policy $X^{\pi^{enroll}}(S_t)$, the trial continuation policy $X^{\pi^{trial}}(S_t)$, and the drug success/failure policy $X^{\pi^{drug}}(S_t)$.

14.3 Modeling uncertainty

There are two potential reasons to develop a formal uncertainty model. The first is for the base model, which we can use both to design policies as well as to run studies. The second might be if we want to use a stochastic lookahead model, where we have to remember that if we are creating a stochastic lookahead model, we are allowed to introduce simplifications.

We are going to begin by working to develop a probabilistic base model, which means we are going to make a best effort to model the real

problem, recognizing that all mathematical models are approximations of the real world.

We need to model three random variables:

- The number of customers $\hat{R}_{t+1}(x_t^{enroll})$ that sign up for the trial.
- The (unobservable) success rate ρ^{true}.
- The number of successes $\hat{X}_{t+1}$, which we do observe.

We address each of these below.

14.3.1 The patient enrollment process $\hat{R}_t$

We are going to use the simple model that we make choices (e.g. by signing up hospitals and clinics) that allow us to expect to sign up x_t^{enroll} patients for week $t+1$. The reality will be different. We propose to model the actual number of arrivals by assuming that they are Poisson with a mean of $\bar{\lambda}^{response}(R_t + x_t^{enroll})$ where $0 < \lambda^{response} < 1$ is the fraction of potential patients who elect to join the trial (which is unknown). This means that we can write

$$Prob[\hat{R}_{t+1}(R_t) = r] \quad = \quad \frac{(\bar{\lambda}^{response}(R_t + x_t^{enroll}))^r e^{-\lambda^{response}(R_t + x_t^{enroll})}}{r!}. \tag{14.7}$$

We can use a truncated Poisson distribution for $\hat{R}_{t+1}$, where we have to recognize that the number of patients that join the trial is limited by the number of potential patients given by $R_{t+1} = R_t + x_t^{enroll}$. Let

$$\bar{R}_t = \bar{\lambda}_t^{response}(R_t + x_t^{enroll})$$

be the expected number of patients that will volunteer for the trial (given R_t) and

$$P_{\hat{R}_{t+1}}(r|x_t^{enroll}, \bar{R}_t) \quad = \quad Prob[\hat{R}_{t+1}(x_t^{enroll}) = r|\bar{R}_t].$$

We write $P_t^{\hat{R}}(r|x_t^{enroll}, \bar{R}_t)$ as a function of x_t^{enroll} and $\bar{R}_t$ to reflect its dependence on the decision and on the number $R_{t+1} = R_t + x_t^{enroll}$.

The truncated Poisson distribution is then given by

$$P_{\hat{R}_{t+1}}(r|x_t^{enroll}, \bar{R}_t) = \begin{cases} \frac{(\bar{R}_t)^r e^{-\bar{R}_t}}{r!} & r = 0, 1, \ldots, x_t^{enroll} \\ 1 - \sum_{r=0}^{x_t^{enroll}-1} P_{\hat{R}_{t+1}}(r|x_t^{enroll}, \bar{R}_t) & \text{If } r = x_t^{enroll} \end{cases}$$

For a population process such as this, a Poisson process is a good starting point. It enjoys the property that the mean equals the variance which equals x_t^{enroll}. It is not uncommon to find that real problems of this type exhibit a higher variance than the Poisson would predict. One way to increase the variance is to add a random δx to the mean x_t^{enroll} that might be uniformly distributed in some range that needs to be tuned.

14.3.2 The success probability ρ^{true}

The successes are driven by the underlying, but unobservable, probability that the treatment will create a success in a patient during a week. We use the Bayesian style of assigning a probability distribution to ρ^{true}. There are three ways to represent the distribution of our belief about ρ^{true}:

- A uniform prior, where we would assume that ρ^{true} is uniformly distributed between 0 and 1.
- A beta distribution with parameters (α_0, β_0).
- A sampled distribution, where we assume that ρ^{true} takes on one of the set of values $(\rho_1, \ldots, \rho_K)$, where we let our initial distribution be

  $$p_{0k}^{\rho} = Prob[\rho^{true} = \rho_k].$$

  We might let $p_{0k} = 1/K$ (this would be comparable to using the uniform prior). Alternatively, we could estimate these from the beta distribution.

For now, we are going to use our sampled distribution since it is easiest to work with.

14.3.3 The success process $\hat{X}_t$

The random number of successes $\hat{X}_{t+1}$, given what we know at time t, depends first on the random variable $\hat{R}_{t+1}$ giving the number of patients who entered the trial, and the unknown probability ρ^{true} of success in the trial. The way to create the distribution of $\hat{X}_{t+1}$ is to use the power of conditioning. We assume that $\hat{R}_{t+1} = r$ and that $\rho^{true} = \rho_k$.

Given that r patients enter the trial and assuming that the probability of success is ρ_k, the number of successes $\hat{X}_{t+1}$ is the sum of r Bernoulli (that is, 0/1) random variables. The sum of r Bernoulli random variables is given by a binomial distribution, which means

$$Prob[\hat{X}_{t+1} = s|\hat{R}_{t+1} = r, \rho^{true} = \rho_k] = \binom{r}{s}\rho_k^s(1-\rho_k)^{r-s}.$$

We can find the unconditional distribution of $\hat{X}_{t+1}$ by just summing over r and k and multiplying by the appropriate probabilities, giving us

$$Prob[\hat{X}_{t+1} = s|\bar{R}_t] = \sum_{k=1}^{K}\left(\sum_{r=0}^{R_t} Prob[\hat{X}_{t+1} = s|\hat{R}_{t+1} = r, \rho^{true} = \rho_k]P_{\hat{R}_{t+1}}(r|x_t^{enroll}, \bar{R}_t)\right.$$

Using explicit probability distributions such as the one for $\hat{X}_{t+1}$ in equation (14.9) is nice when we can find (and compute) them, but there are many complex problems where this is not possible. For example, even equation (14.9) required that we use the trick of using a sampled representation of the continuous random variable ρ^{true}. Without this, we would have had to introduce an integral over the density for ρ^{true}.

Another approach, which is much easier and extends to even more complicated situations, uses Monte Carlo sampling to generate $\hat{R}_{t+1}$ and $\hat{X}_{t+1}$. This process is outlined in figure 14.1, which produces a sample $\hat{X}_{t+1}^1, \ldots, \hat{X}_{t+1}^N$ (and corresponding $\hat{R}_{t+1}^1, \ldots, \hat{R}_{t+1}^N$). We can now approximate the random variable $\hat{X}_{t+1}$ with the set of outcomes $\hat{X}_{t+1}^1, \ldots, \hat{X}_{t+1}^N$, each of which may occur with equal probability.

Step 1. Loop over iterations $n = 1, \ldots, N$:

Step 2a. Generate a Monte Carlo sample $r^n \sim \hat{R}_{t+1}(x^{enroll})$ from the Poisson distribution given by equation (14.8).

Step 2b. Generate a Monte Carlo sample of the true success probability $\rho^n \sim \rho^{true}$.

Step 2c. Given r^n and ρ^n, loop over our r^n patients and generate a 0/1 random variable which is 1 (that is, the drug was a success) with probability ρ^n.

Step 2d. Sum the successes and let this be a sample realization of $\hat{X}^n_{t+1}$.

Step 4. Output the sample $\hat{X}^1_{t+1}, \ldots, \hat{X}^N_{t+1}$.

Figure 14.1: A Monte Carlo-based model of the clinical trial process.

14.4 Designing policies

We are going to use this problem to really understand our full stochastic lookahead policy which we first introduced in chapter 8, given by

$$X^*(S_t) = \arg\max_{x_t \in \mathcal{X}} \Bigg(C(S_t, x_t) + \mathbb{E}_{W_{t+1}} \left\{ \max_{\pi} \mathbb{E}_{W_{t+2},\ldots,W_T} \left\{ \sum_{t'=t+1}^{T} C(S_{t'}, X^{\pi}_{t'}(S_{t'})) | S_{t+1} \right\} | S_t, x_t \right\} \Bigg). \tag{14.10}$$

In particular, we are going to focus on what is meant by that maximization over policies π embedded within the policy.

For our clinical trials application, we have to design policies for the three different decisions: the number of patients to enroll, whether or not to continue the trial, and whether or not the drug is declared a success when the trial is stopped. We are going to begin by designing simple policy function approximations for the decisions of whether to

stop or continue, and if we stop, whether we declare the drug a success or a failure. We then address the more difficult decision of how many patients to enroll in the trial.

14.4.1 Stopping the trial and declaring success or failure

We begin by using our belief about ρ^{true} given by the beta distribution with parameters (α_t, β_t), which gives us an estimate of

$$\bar{\rho}_t = \frac{\alpha_t}{\alpha_t + \beta_t}$$

Now introduce the parameters $\theta^{stop-low}$ and $\theta^{stop-high}$, where we are going to stop the trial and declare success if $\bar{\rho}_t > \theta^{stop-high}$, while we will stop the trial and declare failure if $\bar{\rho}_t < \theta^{stop-low}$. Let $\theta^{stop} = (\theta^{stop-low}, \theta^{stop-high})$. We use these rules to define the policy for stopping the trial as

$$X_t^{trial}(S_t|\theta^{stop}) = \begin{cases} 1 & \text{If } \theta^{stop-low} \leq \bar{\rho}_t \leq \theta^{stop-high} \\ 0 & \text{Otherwise.} \end{cases}$$

If we stop the trial, then the policy for declaring success (1) or failure (0) is given by

$$X_t^{drug}(S_t|\theta^{stop}) = \begin{cases} 1 & \text{If } \bar{\rho}_t > \theta^{stop-high} \\ 0 & \text{If } \bar{\rho}_t < \theta^{stop-low} \end{cases}$$

14.4.2 The patient enrollment policy

It is often the case that problems with a physical state (such as R_t) need a lookahead policy, just as we used with our stochastic shortest path problem. But, as we saw with the stochastic shortest path problem, we get to choose what to put in our stochastic lookahead model. One choice we have to make is the stopping policy $X^{trial}(S_t|\theta^{stop})$ and the success/failure policy $X^{drug}(S_t|\theta^{stop})$, where we propose to use the same parameter vector θ^{stop} in our lookahead model as we do in our base model. We can refer to these as $\tilde{X}^{trial}(\tilde{S}_t|\theta^{stop})$ and $\tilde{X}^{drug}(\tilde{S}_t|\theta^{stop})$, since these now apply only to the lookahead model.

The problem of determining how many new potential patients to enroll is somewhat more difficult, since it is necessary to pay an upfront

cost to acquire more potential patients, and we have to do this under uncertainty about the willingness of patients to join the trial (given by the unknown parameter λ). It is expensive to sign up a lot of potential patients initially if the response rate is high. If the response rate is high, then we can sign up fewer potential patients and then enjoy the high rate of patients joining the trial. However, if the response rate is low, then this may take a long time, incurring the administration cost of running the trial.

To create a full lookahead model as we described in section 14.3, we would create variables such as $\tilde{\lambda}_{tt'}$ for the lookahead version of $\bar{\lambda}_t^{response}$, $\tilde{\rho}_{tt'}$ for $\bar{\rho}_t$, and $(\tilde{\alpha}_{tt'}, \tilde{\beta}_{tt'})$ for (α_t, β_t). Otherwise, all the logic would be the same as the original uncertainty model.

While we can use the full uncertainty model, we can choose to simplify the model in different ways. These choices include:

- The enrollment rate λ_t - We have two options:
  - We can continue to estimate λ_t, where we would introduce the notation $\tilde{\lambda}_{tt'}$ as the estimate at time t' in the lookahead model of the enrollment rate λ.
  - We could fix $\tilde{\lambda}_{tt'} = \bar{\lambda}_t^{response}$, which is our estimate at time t in the base model.
- The drug success rate ρ^{true} - We again have two options:
  - We can continue to estimate the success rate. For this, we would define the variables $(\tilde{\alpha}_{tt'}, \tilde{\beta}_{tt'})$ for accumulating successes and failures in the lookahead model.
  - This means we would fix $(\tilde{\alpha}_{tt'}, \tilde{\beta}_{tt'}) = (\alpha_t, \beta_t)$ within the lookahead model.

Using our choices for modeling uncertainty, we can suggest three different strategies for designing a lookahead model:

Model A Deterministic lookahead model - Here, we are going to assume that the enrollment rate $\tilde{\lambda}_{tt'} = \bar{\lambda}_t^{response}$, which means that the enrollment rate is fixed at the estimate at time t when we

create the lookahead model. We then assume that the true drug success probability is fixed at

$$\tilde{\rho}_{tt'} = \bar{\rho}_t = \frac{\alpha_t}{\alpha_t + \beta_t},$$

which is our estimate at time t in the base model.

Model B We fix our estimate of the enrollment rate at $\tilde{\lambda}_{tt'} = \bar{\lambda}_t^{response}$, but assume that we continue learning about the effectiveness of the drug.

Model C We model the process of learning the enrollment rate $\tilde{\lambda}_{tt'}$ and the drug effectiveness $\tilde{\rho}_{tt'}$.

Note that we did not include the potential fourth model where we fix the drug effectiveness but continue learning the patient enrollment rate (we will see in a minute how silly this model would be).

We are going to use these three models to illustrate the process of designing a lookahead model.

14.4.3 Model A

Model A is a deterministic problem, since we are fixing both the estimated enrollment rate $\tilde{\lambda}_{tt'} = \bar{\lambda}_t^{response}$, and $\tilde{\rho}_{tt'} = \bar{\rho}_t$. The good news is that this is basically a deterministic shortest path problem, where the number of patients we have signed up (in the lookahead model), given by $\tilde{R}_{tt'}$, is like a node in a network, and the decision $\tilde{x}_{tt'}^{enroll}$ is a link that takes us to node $\tilde{R}_{t,t'+1} = \tilde{R}_{tt'} + \tilde{x}_{tt'}^{enroll}$.

To see this, recall equation (5.1) for our deterministic shortest path problem, which we repeat here

$$v_i = \min_{j \in \mathcal{N}_i^+} (c_{ij} + v_j).$$

Now we just replace v_i for the value at node i, with $\tilde{V}_{tt'}(\tilde{R}_{tt'})$ which is the value of having $\tilde{R}_{tt'}$ patients signed up (remember we are in our lookahead model). The decision to go to node j is replaced with the decision to sign up $\tilde{x}_{tt'}^{enroll}$ patients. Instead of this taking us to node j, it takes us to node $\tilde{R}_{tt'} + \tilde{x}_{tt'}^{enroll}$. So Bellman's equation becomes

$$\tilde{V}_{tt'}(\tilde{R}_{tt'}) = \min_{\tilde{x}_{tt'}^{enroll}} (\tilde{C}(\tilde{R}_{tt'}, \tilde{x}_{tt'}^{enroll}) + \tilde{V}_{t,t'+1}(\tilde{R}_{tt'} + \tilde{x}_{tt'}^{enroll})). \quad (14.11)$$

The one-period profit function $\tilde{C}(\tilde{R}_{tt'}, \tilde{x}^{enroll}_{tt'})$ is adapted from the same function for our base model (see equation (14.5)).

There is only one problem with our deterministic lookahead model: we would never stop, because our policy for stopping requires that our estimate of $\tilde{\rho}_{tt'}$ move into the "success" or "fail" regions (it would have to start in the "continue" region, since otherwise we would have stopped the base model). However, this does not mean that we cannot use the deterministic lookahead model: we just have to fix a horizon H and stop when $t' = t + H$.

Using this strategy, we solve our deterministic shortest path problem over the horizon $t' = t, \ldots, t + H$, and then from this find $\tilde{x}^*_{tt}$. Our enrollment policy is then

$$X^{\pi^{enroll}}(S_t) \quad = \quad \tilde{x}^*_{tt}.$$

We are not claiming that this will be an effective policy. We are primarily illustrating the types of modeling approximations that can be made in a lookahead model.

14.4.4 Model B

Now we are going to fix our estimate of the the response rate $\tilde{\lambda}_{tt'}$ at our estimate $\bar{\lambda}^{response}_t$ at time t in the base model. To simplify our model, we are going to assume that the number of enrollments $\tilde{\hat{R}}_{t,t'+1}$ equals the expected number of patients that will volunteer $\tilde{\bar{R}}_{tt'}$. The enrollments $\tilde{\hat{R}}_{t,t'+1}$ are generated deterministically from

$$\tilde{\hat{R}}_{t,t'+1} \quad = \quad \lfloor \bar{\lambda}^{response}_t (\tilde{R}_{tt'} + \tilde{x}^{enroll}_{tt'}) \rfloor,$$

where $\lfloor x \rfloor$ means to round x down to the nearest integer. We then compute the distribution of $\tilde{\hat{X}}_{t,t'+1}$ using $Prob[\hat{X}_{t+1} = s|\bar{R}_t]$ but where we replace $\bar{R}_t$ with $\tilde{\bar{R}}_{tt'}$.

We still have to generate the successes $\tilde{\hat{X}}_{t,t'+1}$ from a simulated truth $\tilde{\rho}_{tt'}$, from which we will update $(\tilde{\alpha}_{tt'}, \tilde{\beta}_{tt'})$.

$$\begin{aligned}
\tilde{\alpha}_{t,t'+1} &= \tilde{\alpha}_{tt'} + \tilde{\hat{X}}_{t,t'+1}, \\
\tilde{\beta}_{t,t'+1} &= \tilde{\beta}_{tt'} + \tilde{\bar{R}}_{tt'} - \tilde{\hat{X}}_{t,t'+1}.
\end{aligned}$$

We model the distribution of $\tilde{\hat{X}}_{t,t'+1}$ using $Prob[\tilde{\hat{X}}_{t,t'+1} = s|\tilde{\bar{R}}_{tt'}]$ in equation (14.9) but conditioning on $\tilde{\bar{R}}_{tt'}$ instead of $\bar{R}_t$ (remember that we can also use the sampled distribution using the method in figure 14.1 instead of the Poisson distribution).

We can solve the lookahead model by adapting Bellman's equation for Model A in equation (14.11) for $t' = t + H, \ldots, t$:

$$\tilde{V}_{tt'}(\tilde{S}_{tt'}) = \min_{\tilde{x}^{enroll}_{tt'}} \left(\tilde{C}(\tilde{S}_{tt'}, \tilde{x}^{enroll}_{tt'}) + \sum_{s=0}^{\tilde{\bar{R}}_{tt'}} Prob[\tilde{\hat{X}}_{t,t'+1} = s|\tilde{\bar{R}}_{tt'}]\tilde{V}_{t,t'+1}(\tilde{S}_{t,t'+1}|\tilde{\hat{X}}_{t,t'+1} = s) \right).$$

where $\tilde{S}_{t,t'+1} = (\tilde{R}_{t,t'+1}, \tilde{\alpha}_{t,t'+1})$ is conditioned on the number of successes $\tilde{\hat{X}}_{t,t'+1} = s$, and where $Prob[\tilde{\hat{X}}_{t,t'+1} = s|\tilde{\bar{R}}_{tt'}]$ comes from equation (14.9). We have to keep in mind that the evolution of $\tilde{R}_{tt'}$ has to reflect if we have decided to stop or continue the trial within the lookahead model.

Our physical state variable (total potential patients) $\tilde{R}_{t,t'+1}$ is given by

$$\tilde{R}_{t,t'+1} = \begin{cases} \tilde{R}_{tt'} + \tilde{x}^{enroll}_{tt'} & \text{If } \tilde{X}^{trial}(\tilde{S}_{tt'}|\theta^{stop}) = 1 \\ 0 & \text{Otherwise} \end{cases}.$$

Note that as with our base model, the number of potential patients drops to zero if we stop the trial in the lookahead model.

Our current estimate of the success of the drug (in the lookahead model) is computed using

$$\tilde{\bar{\rho}}_{tt'} = \frac{\tilde{\alpha}_{tt'}}{\tilde{\alpha}_{tt'} + \tilde{\beta}_{tt'}}.$$

In the expectation, if we condition on the number of successes being $\tilde{\hat{X}}_{t,t'+1} = s$, then the updated belief state $(\tilde{\alpha}_{tt'}, \tilde{\beta}_{tt'})$ is

$$\tilde{\alpha}_{t,t'+1} = \tilde{\alpha}_{tt'} + s, \tag{14.13}$$

$$\tilde{\beta}_{t,t'+1} = \tilde{\beta}_{tt'} + (\tilde{\bar{R}}_{tt'} - s). \tag{14.14}$$

We now have to solve the lookahead model using Bellman's equation in equation (14.12). For this problem, it makes sense to use a large-enough horizon H so that we can confidently assume that we would have stopped the trial by then (that is $\tilde{X}^{trial}(\tilde{S}_{tt'}|\theta^{stop}) = 0$). This means we can assume that $\tilde{V}_{t,t+H}(\tilde{S}_{t,t+H}) = 0$, and work backward from there to time t. Once we have solved the dynamic program, we can pull out our enrollment decision using

$$X_t^{enroll}(S_t) = \arg\min_{\tilde{x}_{tt}^{enroll}} \left(\tilde{C}(\tilde{S}_{tt}, \tilde{x}_{tt}^{enroll}) + \sum_{s=0}^{\tilde{\hat{R}}_{t,t+1}} Prob[\tilde{\hat{X}}_{t,t'+1} = s|\tilde{\hat{R}}_{tt}]\tilde{V}_{t,t+1}(\tilde{S}_{t,t+1}|\tilde{\hat{X}}_{t,t'+1} = s) \right).$$

14.4.5 Model C

Model C is almost the same as the base model, since we are modeling all the different forms of uncertainty. The only way that it is a lookahead model would be our introduction of the simplified policy (our "policy function approximation") for stopping the trial, and for determining if the drug is a success. However, we could ignore these policies and formulate the entire problem as a dynamic program, using the full state variable.