REINFORCEMENT LEARNING AND STOCHASTIC OPTIMIZATION

A unified framework for sequential decisions

Warren B. Powell

June 30, 2021

WILEY-
INTERSCIENCE

A JOHN WILEY & SONS, INC., PUBLICATION

Library of Congress Cataloging-in-Publication Data:

Optimization Under Uncertainty: A unified framework
Printed in the United States of America.

10 9 8 7 6 5 4 3 2 1

CONTENTS

PART I - INTRODUCTION

We have divided the book into 20 chapters organized into six parts. Part I includes three chapters that set the foundation for the rest of the book:

- Chapter 1 provides an introduction to the broad field that we are calling "sequential decision analytics." It introduces our universal modeling framework which reduces sequential decision problems to one of finding methods (rules) for making decisions, which we call *policies.*

- Chapter 2 introduces the major canonical modeling frameworks that have been used by different communities. Our modeling framework will span all of these communities.

- Chapter 3 is an introduction to online learning, where the focus is on sequential vs. batch learning. This can be viewed as an introduction to machine learning, but focusing exclusively on adaptive learning, which is something we are going to be doing throughout the book.

- Chapter 4 sets the stage for the rest of the book by organizing stochastic optimization problems into three categories: 1) problems that can be solved using deterministic mathematics, 2) problems where randomness can be reasonably approximated using a sample (and then solved using deterministic mathematics), and 3) problems that can only be solved with adaptive learning algorithms.

Chapter 1 provides an overview of the universal modeling framework that covers any sequential decision problem. It provides a big picture of our entire framework for modeling and solving sequential decision problems, which should be of value to any reader regardless of their background in decisions under uncertainty. It describes the scope of problems, a brief introduction to modeling sequential decision problems, and sketches the four classes of policies (methods for making decisions) that we use to solve these problems.

Chapter 2 summarizes the canonical modeling frameworks for each of the communities that address some form of sequential decision problem, using the notation of that field. Readers who are entirely new to the field might skim this chapter to get a sense of the variety of approaches that have been taken. Readers with more depth will have some level of expertise in one or more of these canonical problems, and it will help provide a bridge between that problem and our framework.

Chapter 3 covers online learning in some depth. This chapter should be skimmed, and then used as a reference source as needed. A good starting point is to read section 3.1, and then skim the headers of the remaining sections. The book will repeatedly refer back to methods in this chapter.

Finally, chapter 4 organizes stochastic optimization problems into three categories:

1) Stochastic optimization problems that can be solved exactly using deterministic mathematics.

2) Stochastic optimization problems where uncertainty can be represented using a fixed sample. These can still be solved using deterministic mathematics.

3) Stochastic optimization problems that can only be solved using sequential, adaptive learning algorithms. This will be the focus of the rest of the book.

This chapter reminds us that there are special cases of problems that can be solved exactly, possibly by replacing the original expectation with a sampled approximation. The chapter closes by setting up some basic concepts for learning problems, including making the important distinction between online and offline problems, and by identifying different strategies for designing policies for adaptive learning.

CHAPTER 1

SEQUENTIAL DECISION PROBLEMS

A sequential decision problem, simply stated, consists of the sequence

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

As we make decisions, we incur costs or earn rewards. Our challenge is how to represent the information that will arrive in the future, and how to make decisions both now, and in the future. Modeling these problems, and making effective decisions in the presence of the uncertainty of new information, is the goal of this book.

The first step in sequential decision problems is to understand what decisions are being made. It is surprising how often it is that people faced with complex problems, which spans scientists in a lab to people trying to solve major health problems, are not able to identify the decisions they face. We then want to find a method for making decisions. There are at least 45 words in the English language that are equivalent to “method for making a decision,” but the one we have settled on is *policy*. Designing effective policies will be the focus of most of this book.

Even more subtle can be identifying the different sources of uncertainty. It can be hard enough trying to identify potential decisions, but thinking about all the random events that can affect whatever it is that you are managing, whether it is reducing disease, managing inventories, or making investments, can seem like a hopeless challenge. Not only are there a wide range of sources of uncertainty, but there is also tremendous variety in how they behave.

Making decisions under uncertainty spans an exceptionally rich set of problems in analytics, arising in fields such as engineering, business, economics, finance, health, transportation, and energy. It encompasses active learning problems, where the decision is to collect information, that arise in the experimental sciences, medical decision making, e-commerce, and sports. It also includes iterative algorithms for stochastic search, which arises in machine learning (finding the model that best fits the data) or finding the best layout for an assembly line using a simulator. It also includes two-agent games and multiagent systems. In fact, we might claim that virtually any human enterprise will include instances of sequential decision problems.

Decision making under uncertainty is a universal experience, something every human has had to manage since our first experiments trying new foods when we were two years old. Some samples of everyday problems where we have to manage uncertainty in our own lives include:

- Personal decisions - These might include how much to withdraw from an ATM machine, finding the best path to a new job, and deciding what time to leave to make an appointment.
- Food shopping - We all have to eat, and we cannot run to the store every day, so we have to make decisions of when to go shopping, and how much to stock up on different items when we do go.
- Health decisions - Examples include joining a health club, getting annual checkups, having that mole checked, using dental floss, and scheduling a colonoscopy.
- Investment decisions - What mutual fund should you use? How should you allocate your investments? How much should you put away for retirement? Should you rent or purchase a house?

Sequential decision problems are ubiquitous, and as a result come in many different styles and flavors. Decisions under uncertainty span virtually every major field. Table 1.1 provides a list of problem domains and a sample of questions that can arise in each of these fields. Not surprisingly, a number of different analytical fields have emerged to solve these problems, often using different notational systems, and presenting solution approaches that are suited to the characteristics of the problems in each setting.

This book will provide the analytical foundation for sequential decision problems using a "model first, then solve" philosophy. While this is standard in fields such as deterministic optimization and machine learning, it is not at all standard in the arena of making decisions under uncertainty. The communities that work on sequential decision problems tend to come up with a method for solving a problem, and then look for applications. This can come across as if we have a hammer looking for a nail.

The limitation of this approach is that the different methods that have been developed can only serve a subset of problems. Consider one of the simplest and most classical sequential decision problems: managing an inventory of product to serve demand. Let R_t be our inventory at time t, x_t is how much we order (that initially arrives instantly), to serve a demand $\hat{D}_{t+1}$ that is not known at time t. The evolution of the inventory R_t is given by

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

For this problem, we might use the following policy: when the inventory falls below a value θ^{min}, order enough to bring it up to θ^{max}. All we have to do is to determine the parameter vector $\theta = (\theta^{min}, \theta^{max})$. The policy is quite simple, but finding the best value of θ can be quite challenging.

Field	Questions
Business	What products should we sell, with what features? Which supplies should you use? What price should you charge? How should we manage our fleet of delivery vehicles? Which menu attracts the most customers?
Economics	What interest rate should the Federal Reserve charge given the state of the economy? What levels of market liquidity should be provided? What guidelines should be imposed on investment banks?
Finance	What stocks should a portfolio invest in? How should a trader hedge a contract for potential downside? When should we buy or sell an asset?
Internet	What ads should we display to maximize ad-clicks? 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?
Materials science	What combination of temperatures, pressures and concentrations should we use to create a material with the highest strength?
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? What mode of transportation should be used? 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? How should a submarine behave given the presence of adversarial submarines?
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 list of application domains and decisions that need to be made.

Now consider a series of inventory problems with increasing complexity, illustrated by the setting in figure 1.1 of a warehouse in the south-eastern United States ordering inventory from China:

1) The inventory we order comes from China, and might take 90 days to arrive.

2) We have to serve demand that varies seasonally (and dramatically changes around the Christmas holiday season).

3) We are given the choice to use air freight for a particular order that reduces the time by 30 days.

4) We are selling expensive gowns, and we have to pay special attention to the risk of a stockout if there is a delay in either the production (which we can handle by using air freight) or a delay in offloading at the port.

Figure 1.1 Illustration of shipments coming from China to the U.S. with a threat of a storm.

5) The gowns come in different styles and colors. If we run short of one color, the customer might be willing to accept a different color.

6) We are allowed to adjust the price of the item, but we do not know precisely how the market will respond. As we adjust the price and observe the market response, we learn from this observation and use what we learn to guide future pricing decisions.

The simple inventory problem in equation (1.1) has just a single decision, x_t, specifying how much inventory to order now. In a real problem, the decisions that have to be considered might include:

- How much to order, and the choice of delivery commitment that determines how quickly the order arrives: rush orders, normal, relaxed.
- Pricing of current inventory while waiting for the new inventory to arrive.
- Reservations for space on cargo ships in the future.
- The speed of the cargo ship.
- Whether to rush additional inventory via air freight to fill a gap due to a delay.
- Whether to use truck or rail to move the cargo in from the port.
- Contracts for hedging against currency fluctuations.
- Product design decisions that might affect the speed of manufacturing.
- Product quality guidelines.

Then, we have to think about the different forms of uncertainty for a product that might take at least 90 days to arrive:

- The time to complete manufacturing.
- Weather delays affecting ship speeds.
- Land transportation delays.

- Product quality on arrival.
- Currency changes.
- Demand for inventory on hand between now and the arrival of new inventory.

If you set up a toy problem such as equation (1.1), you would never think about all of these different decisions and sources of uncertainty. Our presentation will feature a rich modeling framework that emphasizes our philosophy:

Model first, then solve.

We will introduce, for the first time in a textbook, a universal modeling framework for *any* sequential decision problem. We will introduce four broad classes of methods, known as policies, for making decisions that span *any* method that might be used, including anything in the academic literature or used in practice. Our goal is not to always choose the policy that performs the best, since there are multiple dimensions to evaluating a policy (computational complexity, transparency, flexibility, data requirements). However, we will always choose our policy with one eye to performance.

1.1 THE AUDIENCE

This book is aimed at readers who want to develop models that are practical, flexible, scalable, and implementable for sequential decision problems in the presence of different forms of uncertainty. The ultimate goal is to create software tools that can solve real problems. We use careful mathematical modeling as a necessary step for translating real problems into software. The readers who appreciate both of these goals will enjoy our presentation the most.

Given this, we have found that this material is accessible to professionals from a wide range of fields, spanning application domains (engineering, economics, and the sciences) to those with more of a methodological focus (such as machine learning, computer science, optimal control, and operations research) with a comfort level in probability and statistics, linear algebra, and, of course, computer programming.

Our presentation emphasizes modeling and computation, with minimal deviations into theory. The vast majority of the book can be read with a good course in probability and statistics, and a basic knowledge of linear algebra. Occasionally we will veer into higher dimensional applications such as resource allocation problems (e.g. managing inventories of different blood types, or trading portfolios of assets) where some familiarity with linear, integer and/or nonlinear programming will be useful. However, these problems can all be solved using powerful solvers with limited knowledge of how these algorithms actually work.

This said, there is no shortage of algorithmic challenges and theoretical questions for the advanced Ph.D. student with a strong background in mathematics.

1.2 THE COMMUNITIES OF SEQUENTIAL DECISION PROBLEMS

Figure 1.2 shows some prominent books from various methodological communities in the sequential decision-making field. These communities, which are discussed in greater depth in chapter 2, are listed in table 1.2 in the approximate order in which the field emerged. We

Introduction to Stochastic Programming
Handbook of Simulation Optimization
Robust Optimization
Introduction to Decision Analysis
Approximate Dynamic Programming
Optimal Learning
Model Predictive Control
Dynamic Programming and Optimal Control
OPTIMAL CONTROL
Introduction to Stochastic Search and Optimization
Multi-Armed Bandit Allocation Indices
Reinforcement Learning
Markov Decision Processes
Lectures on Stochastic Programming
Online Computation and Competitive Analysis
Stochastic Controls

Figure 1.2 A sampling of major books representing different fields in stochastic optimization.

note that there are two distinct fields that are known as derivative-based stochastic search, and derivative-free stochastic search, that both trace their roots to separate papers published in 1951.

Each of these communities deals with some flavor of sequential decision problems, using roughly eight notational systems, and an overlapping set of algorithmic strategies. Each field is characterized by at least one book (often several), and thousands of papers (in some cases, thousands of papers each year). Each community tends to have problems that best fit the tools developed by that community, but the problem classes (and tools) are continually evolving.

The fragmentation of the communities (and their differing notational systems) disguises common approaches developed in different areas of practice, and challenges cross-fertilization of ideas. A problem that starts off simple (like the inventory problem in (1.1)) lends itself to a particular solution strategy, such as dynamic programming. As the problem grows in realism (and complexity), the original technique will no longer work, and we need to look to another community to find a suitable method.

1) Derivative-based stochastic search	9) Stochastic programming
2) Derivative-free stochastic search	10) Multiarmed bandit problem
3) Decision trees	11) Simulation optimization
4) Markov decision processes	12) Active learning
5) Optimal control	13) Chance constrained programming
6) Approximate dynamic programming	14) Model predictive control
7) Reinforcement learning	15) Robust optimization
8) Optimal stopping	

Table 1.2 Fields that deal with sequential decisions under uncertainty.

We organize all of these fields under the title of "reinforcement learning and stochastic optimization." "Stochastic optimization" refers generally to the analytical fields that address decisions under uncertainty. The inclusion of "reinforcement learning" in the title reflects the growing popularity of this community, and the use of the term to apply to a steadily expanding set of methods for solving sequential decision problems. The goal of this book is to provide a unified framework that covers *all* of the communities that work on these problems, rather than to favor any particular method. We refer to this broader field as *sequential decision analytics*.

Sequential decision analytics requires integrating tools and concepts from three core communities from the mathematical sciences:

Statistical machine learning - Here we bring together the fields of statistics, machine learning and data sciences. Most (although not all) of our applications of these tools will involve recursive learning. We will also draw on the fields of both frequentist and Bayesian statistics, but all of this material is provided here.

Mathematical programming - This field covers the core methodologies in derivative-based and derivative-free search algorithms, which we use for purposes ranging from computing policies to optimizing the parameters of a policy. Occasionally we will encounter vector-valued decision problems that require drawing on tools from linear, integer and possibly nonlinear programming. Again, all of these methods are introduced and presented without assuming any background in stochastic optimization.

Stochastic modeling and simulation - Optimizing a problem in the presence of uncertainty often requires a careful model of the uncertain quantities that affect the performance of a process. We include a basic introduction to Monte Carlo simulation methods, but expect a background in probability and statistics, including the use of Bayes theorem.

While our presentation does not require advanced mathematics or deep preparation in any methodological area, we will be blending concepts and methods from all three of these fields. Dealing with uncertainty is inherently more subtle than deterministic problems, and requires more sophisticated modeling than arises in machine learning.

1.3 OUR UNIVERSAL MODELING FRAMEWORK

Central to the entire book is the use of a single modeling framework, as is done in deterministic optimization and machine learning. Ours is based heavily on the one widely used in optimal control. This has proven to be the most practical and flexible, and offers a clear relationship between the mathematical model and its implementation in software. While much of our presentation will focus on modeling sequential decision problems and developing practical methods for making decisions, we also recognize the importance of developing models of the different sources of uncertainty (a topic that needs a book of its own).

Although we revisit this in more detail in chapter 9, it helps to sketch our universal modeling framework. The core elements are

- State variables S_t - The state variable contains everything we know, and only what we need to know, to make a decision and model our problem. State variables include physical state variables R_t (the location of a drone, inventories, investments in stocks), other information I_t about parameters and quantities we know perfectly (such as current prices and weather), and beliefs B_t, in the form of probability distributions, describing parameters and quantities that we do not know perfectly (this could be an estimate of how much a drug will lower the blood sugar in a new patient, or how the market will respond to price).

- Decision variables x_t - A decision variable can be binary (hold or sell), a discrete set (drugs, products, paths), a continuous variable (such as a price or dosage), and vectors of discrete and continuous variables. Decisions are subject to constraints $x_t \in \mathcal{X}_t$, and we make decisions using a method we call a policy $X^\pi(S_t)$. We introduce the notation for a policy, but we defer the design of the policy until after we complete the model. This is the basis of what we call *model first, then solve*.

- Exogenous information W_{t+1} - This is the information that we learn after we make a decision (market response to a price, patient response to a drug, the time to traverse a path), that we did not know when we make a decision. Exogenous information comes from outside whatever system we are modeling. (Decisions, on the other hand, can be thought of as an *endogenous information process* since we make decisions, a form of information, internally to the process.)

- The transition function $S^M(S_t, x_t, W_{t+1})$ which consists of the equations required to update *each* element of the state variable. This covers all the dynamics of our system, including the updating of estimates and beliefs for sequential learning problems. Transition functions are widely used in control theory using the notation $f(x, u, w)$ (for state x, control u and information w); our notation, which stands for the “state transition model” or “system model” helps us avoid using the popular letter $f(\cdot)$.

- The objective function - This first consists of the contribution (or reward, cost, ...) we make each time period, given by $C(S_t, x_t)$, where $x_t = X^\pi(S_t)$ is determined by our policy, and S_t is our current state, which is computed by the transition function. As we are going to demonstrate later in the book, there are different ways to write the objective function, but our most common will be to maximize the cumulative contributions, which we write as

$$\max_\pi \mathbb{E}\left\{\sum_{t=0}^{T} C(S_t, X^\pi(S_t))|S_0\right\}. \tag{1.2}$$

  where the expectation $\mathbb{E}$ means “take an average over all types of uncertainty” which might be uncertainty about how a drug will perform, or how the market will respond to price (captured in the initial state S_0), as well as the uncertainty in the information $W_1, \ldots, W_t, \ldots$ that arrives over time. The maximization over policies simply means that we want to find the best method for making decisions. Most of this book is dedicated to the challenge of searching over policies.

Once we have identified these five elements, we still have two remaining steps to complete before we are done.

- Stochastic modeling (also known as uncertainty quantification) - There can be uncertainty about parameters and quantities in the state variable (including the initial state S_0), as well as our exogenous information process $W_1, W_2, \ldots, W_t, \ldots$. In some instances, we may avoid modeling the W_t process by observing a physical system.

Otherwise, we need a mathematical model of the possible realizations of W_{t+1} given S_t and our decision x_t (either of which can influence W_{t+1}).

- Designing policies - Only after we are done with modeling do we turn to the problem of designing the policy $X^\pi(S_t)$. This is the point of departure between this book and all the books in our jungle of stochastic optimization. We do not pick policies before we develop our model, but once the modeling is done, we will provide a roadmap to every possible policy, with guidelines of how to choose among them.

  The policy π consists of some type of function $f \in \mathcal{F}$, possibly with tunable parameters $\theta \in \Theta^f$ that are associated with the function f, where the policy maps the state to a decision. This means that we could write (1.2) as

$$\max_{\pi=(f\in\mathcal{F},\theta\in\Theta^f)} \mathbb{E}\left\{\sum_{t=0}^{T} C(S_t, X^\pi(S_t))|S_0\right\}. \tag{1.3}$$

  This leaves the question: How do we search over functions? Most of this book is dedicated to describing precisely how to do this.

Using this notation, we can revise our original characterization of a sequence decision problem, which we earlier described as *decision, information, decision, information, ...* as 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 we now write the triplet "state, decision, new information" to capture what we know (the state variable S_t), which we use to make a decision x_t, followed by what we learn after we make a decision, the exogenous information W_{t+1}. We earn a contribution $C(S_t, x_t)$ from our decision x_t (we could say we incur a cost, or any other metric), where the decision comes from a policy $X^\pi(S_t)$.

There are many problems where it is more natural to use a counter n (the n^{th} experiment, the n^{th} customer arrival), in which case we would write our sequential decision problem as

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

There are even 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.

We note in passing that there are problems that consist of "decision, information, stop," "decision, information, decision, stop," "information, decision, information, decision, . . .," and problems where the sequencing proceeds over an infinite horizon. We use a finite sequence as our default model.

We can illustrate our modeling framework using our simple inventory problem that we started with above.

- State variables S_t - For the simplest problem this is the inventory R_t.

- Decision variables x_t - This is how much we order at time t, and for now, we assume it arrives right away. We also introduce our policy $X^\pi(S_t)$, where $x_t = X^\pi(S_t)$, which we will design after we create our model.

- Exogenous information W_{t+1} - This would be the demand $\hat{D}_{t+1}$ that arises between t and $t+1$.

- The transition function $S^M(S_t, x_t, W_{t+1})$ - This would be the evolution of our inventory R_t, given by

$$R_{t+1} = \max\{0, R_t + x_t - \hat{D}_{t+1}\}. \tag{1.4}$$

- The objective function - This is an example of a problem where it is more natural to write the single-period contribution function *after* we observe the information W_{t+1} since this contains the demand $\hat{D}_{t+1}$ that we will serve with the inventory x_t we order in period t. For this reason, we write our contribution function as

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

  where p is the price at which we sell our product, and c is the cost per unit of product. Our objective function would be given by

$$\max_\pi \mathbb{E}\left\{\sum_{t=0}^{T} C(S_t, X^\pi(S_t), W_{t+1})|S_0\right\},$$

  where $x_t = X^\pi(S_t)$, and we have to be given a model of the exogenous information process $W_1, \ldots, W_T$. Since the exogenous information is random, we have to take the expectation $\mathbb{E}$ of the sum of contributions to average over all the possible outcomes of the information process.

Our next step would be to develop a mathematical model of the distribution of demand $\hat{D}_1, \hat{D}_2, \ldots, \hat{D}_t, \ldots$ which draws on tools that we introduce in chapter 10.

To design our policy $X^\pi(S_t)$, we might turn to the academic literature that shows, for this simple problem, that the policy has an order-up-to structure given by

$$X^{Inv}(S_t|\theta) = \begin{cases} \theta^{max} - R_t & \text{if } R_t < \theta^{min}, \\ 0 & \text{otherwise.} \end{cases} \tag{1.5}$$

This is a parameterized policy, which leaves us the challenge of finding $\theta = (\theta^{min}, \theta^{max})$ by solving

$$\max_\theta \mathbb{E}\left\{\sum_{t=0}^{T} C(S_t, X^{Inv}(S_t|\theta), W_{t+1})|S_0\right\}. \tag{1.6}$$

Here we chose a particular class of policy, and then optimized within the class.

We pause to note using our modeling approach creates a direct relationship between our mathematical model and computer software. Each of the variables above can be translated directly to a variable name in a computer program, with the only exception that the expectation operator has to be replaced with an estimate based on simulation (we show how to do this very carefully). This relationship between mathematical model and computer software does not exist with most of the current modeling frameworks used for decisions under uncertainty, with one major exception (optimal control, as we note in chapter 2 and chapter 9).

Later in the book we are going to introduce a number of variations to this inventory problem that make the model much richer (and more realistic), but more complex. For the moment, we are going to change our problem just by changing the characteristics of the data. Let's assume that our demands $\hat{D}_t$ have a strong seasonal pattern (for example, serving beer in the presence of major sports events), and that we have a constraint $x_t \leq u$ on how much we can order at any point in time. We might have random prices and costs, and we may have the ability to order product in advance.

Now we have to think ahead to make sure that we order enough to meet future demand while living with our ordering constraint. Our simple order-up-to policy (1.5) will no longer be effective. We are going to need a richer model, and a more sophisticated policy. As we progress through the book, we will show that our five-step universal modeling framework holds up for modeling much more complex problems. In addition, the next section will introduce four classes of policies that will span any method that we might want to consider to solve more complex versions of our problem.

At this point the natural question should be: How do we solve this? In other words, what happens to our policy now? Clearly we cannot use our simple order-up-to policy that was so convenient before.

We do not know if we are going to be able to find an optimal policy, but we have a roadmap that includes every possible policy that we, or anyone else, might think of. The next section provides a sketch of this roadmap.

1.4 DESIGNING POLICIES FOR SEQUENTIAL DECISION PROBLEMS

What often separates one field of stochastic optimization from another is the type of policy that is used to solve a problem. Possibly the most important aspect of our unified framework in this volume is how we have identified and organized different classes of policies. These are first introduced in chapter 7 in the context of derivative-free stochastic optimization (a form of pure learning problem), and then in greater depth in chapter 11 on designing policies, which sets the stage for the entire remainder of the book. In this section we are going to provide a peek at our approach for designing policies.

The entire literature on making decisions under uncertainty can be organized along two broad strategies for creating policies:

Policy search - This includes all policies where we need to search over:

- Different classes of functions $f \in \mathcal{F}$ for making decisions. For example, the order-up-to policy in equation (1.5) is a PFA in the form of nonlinear parametric function.
- Any tunable parameters $\theta \in \Theta^f$ that are introduced by function f. $\theta = (\theta^{min}, \theta^{max})$ in equation (1.5)) is an example.

If we select a policy that contains parameters, then we have to find the set of parameters θ to maximize (or minimize) an objective function such as (1.6).

Lookahead approximations - These are policies formed so we make the best decision now given an approximation of the downstream impact of the decision. These are the policy classes that have attracted the most attention from the research community.

Our order-up-to policy $X^{Inv}(S_t|\theta)$ is a nice example of a policy that has to be optimized (we might say tuned). The optimization can be done using a simulator, as is implied in equation (1.6), or in the field.

Each of these two strategies produce policies that can be divided into two classes, creating four classes of policies. We describe these below.

1.4.1 Policy search

Policies in the policy search class can be divided into two subclasses:

1) Policy function approximations (PFAs) - These are analytical functions that map a state (which includes all the information available to us) to a decision (the order-up-to policy in equation (1.5) is a PFA). These are discussed in greater depth in chapter 12.

2) Cost function approximations (CFAs) - CFA policies have an imbedded minimization or maximization problem *within the policy*, where we might make parametric modifications to the objective function or the constraints. The concept of CFAs are presented in this book as a major class of policies for the first time. CFAs are introduced and illustrated in chapter 13.

PFAs are any analytical function that maps what we know in the state variable to a decision. These analytical functions come in three flavors:

Lookup tables - These are used when a discrete state S can be mapped to a discrete action, such as:

- If the patient is male, over 60 with high blood sugar, then prescribe metformin.
- If your car is at a particular intersection, turn left.

Parametric functions - These describe any analytical functions parameterized by a vector of parameters θ. Our order-up-to policy is a simple example. We might also write it as a linear model such as

$$X^{PFA}(S_t|\theta) = \theta_1\phi_1(S_t) + \theta_2\phi_2(S_t) + \theta_3\phi_3(S_t) + \theta_4\phi_4(S_t).$$

where $\phi_f(S_t)$ are features extracted from information in the state variable. Neural networks are another option.

Nonparametric functions - These include functions that might be locally linear approximations, or deep neural networks.

The second class of functions that can be optimized using policy search is called parametric *cost function approximations*, or CFAs, which are parameterized optimization problems. A simple CFA used in learning problems is called interval estimation and might be used to determine which ad gets the most clicks on a website. Let $\mathcal{X} = \{x_1, \ldots, x_M\}$ be the set of ads (there may be thousands of them), and let $\bar{\mu}_x^n$ be our current best estimate of the probability that ad x will be clicked on after we have run n observations (across all ads). Then let $\bar{\sigma}_x^n$ be the standard deviation of the estimate $\bar{\mu}_x^n$. Interval estimation would choose as the next ad

$$X^{CFA}(S^n|\theta) = \arg\max_{x\in\mathcal{X}} \left(\bar{\mu}_x^n + \theta\bar{\sigma}_x^n\right). \tag{1.7}$$

The distinguishing features of a CFA is that it requires solving an imbedded optimization problem (the max over ads), and there is a tunable parameter θ.

Once we introduce the idea of solving an optimization problem within the policy (as we did with the policy in (1.7)), we can solve *any* parameterized optimization problem. We are no longer restricted to the idea that x has to be one of a set of discrete choices; it can be a large integer program, such as those used to plan airline schedules with schedule slack inserted to handle possible weather delays, or planning energy generation for tomorrow with reserves in case a generator fails (both of these are real instances of CFAs used in practice).

1.4.2 Policies based on lookahead approximations

A natural strategy for making decisions is to consider the downstream impact of a decision you make now. There are two ways of doing this:

3) Value function approximations (VFAs) - One popular approach for solving sequential decision problems applies the principles of a field known as *dynamic programming* (or Markov decision processes). Imagine our state variable tells us where we are on a network where we have to make a decision, or the amount of inventory we are holding. Assume that someone tells us that if we are in state S_{t+1} at time $t+1$ (that is, we are at some node in the network or will have some level of inventory), that $V_{t+1}(S_{t+1})$ is the "value" of being in state S_{t+1}, which we can think of as the cost of the shortest path to the destination, or our expected profits from time $t+1$ onward if we start with inventory S_{t+1}.

Now assume we are in a state S_t at time t and trying to determine which decision x_t we should make. After we make the decision x_t, we observe the random variable(s) W_{t+1} that take us to $S_{t+1} = S^M(S_t, x_t, W_{t+1})$ (for example, our inventory equation (1.4) in our example above). Assuming we know $V_{t+1}(S_{t+1})$, we can find the value of being in state S_t by solving

$$V_t(S_t) = \max_{x_t} \big(C(S_t, x_t) + \mathbb{E}_{W_{t+1}}\{V_{t+1}(S_{t+1})|S_t\} \big), \tag{1.8}$$

where it is best to think of the expectation operator $\mathbb{E}_{W_{t+1}}$ as averaging over all outcomes of W_{t+1}. The value of x_t^* that optimizes equation (1.8) is then the optimal decision for state S_t. The first period contribution $C(S_t, x_t^*)$ plus the future contributions $\mathbb{E}_{W_{t+1}}\{V_{t+1}(S_{t+1})|S_t\}$ gives us the value $V_t(S_t)$ of being in state S_t now. When we know the values $V_t(S_t)$ for all time periods, and all states, we have a VFA-based policy given by

$$X_t^{VFA}(S_t) = \arg\max_{x_t} \big(C(S_t, x_t) + \mathbb{E}_{W_{t+1}}\{V_{t+1}(S_{t+1})|S_t\} \big), \tag{1.9}$$

where "$\arg\max_{x_t}$" returns the value x_t that maximizes (1.9).

Equation (1.9) is a powerful way of computing optimal policies, but it is rarely computable in practical problems. For this reason, a number of communities have developed ways of approximating the value function under names such as approximate dynamic programming, adaptive dynamic programming or, most visibly, reinforcement learning. These fields replace the exact value function $V_{t+1}(S_{t+1})$ with an approximation $\overline{V}_{t+1}(S_{t+1})$ estimated using machine learning.

VFA-based policies have attracted considerable attention from the research literature, and are possibly the most difficult of the four classes of policies. This is the reason why we cover it over the span of five chapters (chapters 14 - 18).

4) Direct lookahead approximations (DLAs) - The easiest example of a lookahead policy is a navigation system which plans a path to your destination, and then tells you which turn to take next. As new information arrives, the path is updated.

This is an example of a deterministic lookahead for a stochastic problem. While deterministic lookaheads are useful in some applications, there are many where we have to explicitly consider uncertainty as we make a decision, which means we have to solve a stochastic optimization problem within our direct lookahead policy! Direct lookahead policies are discussed in greater depth in chapter 19.

1.4.3 Mixing and matching

It is possible to create hybrids. We can create a lookahead policy H periods into the future, and then use a value function approximation. We can use a deterministic lookahead, but introduce tunable parameters to make it work better under uncertainty. We can combine a PFA (think of this as some analytical function that suggests a decision), and weight any deviation of the decision from the PFA and add it to any other optimization-based policy. When we get to stochastic lookaheads in chapter 19, we may end up using all four classes at the same time.

An example of a hybrid policy is determining both the path to drive to a destination, and the time of departure. Navigation systems use a deterministic lookahead, solving a shortest path problem using point estimates of the travel times on each link of a network. This path might produce an estimated travel time of 40 minutes, but when do you actually leave? Now you are aware of the uncertainty of traffic, so you might decide to add in a buffer. As you repeat the trip, you may adjust the buffer up or down as you evaluate the accuracy of the estimate. This is a combined direct lookahead (since it plans a path into the future) with a tunable parameter (making it a form of PFA).

As we said, we cannot tell you how to solve any particular problem (the diversity is simply too great), but we will give you a complete toolbox, with some guidelines to help in your choice.

1.4.4 Optimality of the four classes

There is a widespread misconception in the academic research literature that equation (1.8) (known either as Bellman's equation, or the Hamilton-Jacobi equation) is the basis for creating optimal policies, and that any path to designing good (that is, near optimal) policies have to start with Bellman's equation. This is simply not true.

Any of the four classes of policies can contain the optimal policy for specific problem classes. The problem that arises is purely computational. For example, for the vast majority of real applications, Bellman's equation (1.8) is simply not computable. Trying to replace the true value function $V_{t+1}(S_{t+1})$ in equation (1.8) with some approximation $\overline{V}_{t+1}(S_{t+1})$ may work quite well, but there are many settings where it is just not going to produce effective policies. In addition, once you start talking about using approximations of the value function, you open yourself up to the possibility that any of the other three classes of policies may work just as well or (often) better. This is the reason that there are so many people making decisions over time, in the presence of new information, and who do not use (and have not even heard of) Bellman's equation.

1.4.5 Pulling it all together

We claim that the four classes of policies (PFAs, CFAs, VFAs and DLAs) are universal, and cover every method that has been proposed by any of the communities listed earlier, as well as anything used in practice.

Of the four classes, the academic community has focused primarily on VFAs and various forms of DLAs (both deterministic and stochastic). By contrast, our belief is that PFAs and CFAs are much more widely used in practice. CFAs in particular have been largely overlooked in the academic community, but are widely used in practice in an ad hoc way (they are typically not tuned). PFAs and CFAs (that is, the policy search classes) are preferred in practice because they are simpler, but as we will see over and over again:

The price of simplicity is tunable parameters, and tuning is hard!

1.5 LEARNING

A significant part of decision analytics involves learning. Traditional machine learning involves being given a dataset consisting of inputs x^n and the associated response y^n, and then finding a function $f(x|\theta)$ which might be a linear model such as

$$f(x|\theta) = \theta_0 + \theta_1 \phi_f(x) + \theta_2 \phi_f(x) + \ldots + \theta_F \phi_F(x)$$

where the functions $\phi_f(x)$ extract features from the data in x. The inputs x might be the words in a document, a patient history, weather data, or customer data such as personal data and recent buying history. We might also look at nonlinear models, hierarchical models, and even a neural network. We then have to fit the model by solving the optimization problem

$$\min_\theta \frac{1}{N} \sum_{n=1}^{N} (y^n - f(x^n|\theta))^2.$$

This is classical batch learning.

When we are making decisions sequentially, we also learn sequentially. We might have a patient arrive with medical history h^n; we then decide on treatment $x^{treatment,n}$ using a policy $X^\pi(S^n)$ (where S^n includes the patient history h^n). After choosing the treatment, we wait to observe the response, which we would index by y^{n+1} for the same reason that after making decision x^n we observe W^{n+1}. The index "$n+1$" indicates that this is new information not contained in any variable indexed by n.

Our belief state B^n (within the state variable S^n) contains all the information we need to update our estimate θ^n using the new observation y^{n+1}. All of this updating is buried in the transition

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

just as y^{n+1} is contained within W^{n+1}. The methods for doing this adaptive updating are all covered in chapter 3 on online learning, which is the term the machine learning community uses for learning in a sequential, versus batch, setting.

There are a number of opportunities for using online learning in sequential decision analytics:

1) Approximating the expectation of a function $\mathbb{E}F(x, W)$ to be maximized.

2) Creating an approximate policy $X^\pi(S|\theta)$.

3) Approximating the value of being in a state S_t which we typically represent by $\overline{V}_t(S_t)$.

4) Learning any of the underlying models in a dynamic system. These include:

   4a) The *transition function* $S^M(S_t, x_t, W_{t+1})$ which might describe how a future activity depends on the past.

4b) The cost or contribution functions which might be unknown if we are trying to replicate human behavior.

5) Parametric cost function approximations, where we use learning to modify the objective function and/or constraints.

The tools for estimating these functions are covered in chapter 3, but we visit the specific settings of these different problems throughout the rest of the book.

1.6 THEMES

Our presentation features a series of themes that run through the book. This section reviews some of these.

1.6.1 Blending learning and optimization

We distinguish three major problem classes:

- Pure learning problems - Here we face an unknown function, say $F(x) = \mathbb{E}F(x, W)$, where W is a random variable with unknown distribution, and where we can only do noisy observations. Our goal is to learn an estimate of $F(x)$ so that we can find the best value of x. A key feature of $F(x)$, or more specifically $F(x, W)$, is that it is not changing over time. We call these *state-independent problems*, meaning that the underlying problem, the function $F(x) = \mathbb{E}F(x, W)$, is not changing over time, but our belief about the function is changing as we make observations and learn about the problem. These problems are studied in chapters 5 for derivative-based stochastic optimization, and chapter 7 for the richer derivative-free stochastic optimization. The algorithm (or method) for learning these functions can be described as sequential decision problems.

- State-dependent problems without learning - These are problems that depend on a dynamically changing state. These include any of a vast range of dynamic resource allocation problems (managing blood, energy, money, people, equipment, ...) in the presence of different sources of uncertainty (demands, prices, costs, times, ...), but which do not require learning any exogenous parameters..

- Hybrid problems - Once we have learned how to handle pure learning problems and more general state-dependent problems (such as resource allocation problems), we now have the framework and the tools to design creative solutions to problems such as those arise in disease mitigation, where we are dynamically managing resources (vaccines, medical technicians, supplies) while also learning about vaccination rates, drug efficacy, and disease spread. An important class of hybrid problems are multi-agent systems, where agents need to learn about each other while making effective decisions to manage themselves.

1.6.2 Bridging machine learning to sequential decisions

Finding the best policy is the same as finding the best function that achieves the lowest cost, highest profits or best performance. Analogs to this stochastic optimization problem appear in statistics and machine learning, where a common problem is to use a dataset

(x^n, y^n), where $x^n = (x_1^n, \ldots, x_K^n)$ is used to predict y^n. For example, we might specify a linear function of the form:

$$y^n = f(x^n|\theta) = \theta_0 + \theta_1 x_1^n + \ldots + \theta_K^n x_K + \epsilon^n, \tag{1.10}$$

where ϵ^n is a random error term that is often assumed to be normally distributed with mean 0 and some variance σ^2.

We can find the parameter vector $\theta = (\theta_1, \ldots, \theta_K)$ by solving

$$\min_\theta \frac{1}{N} \sum_{n=1}^{N} \left(y^n - f(x^n|\theta)\right)^2. \tag{1.11}$$

Our problem of fitting a model to the data, then, involves two steps. The first is to choose the function $f(x|\theta)$, which we have done by specifying the linear model in equation (1.10) (note that this model is called “linear” because it is linear in θ). The second step involves solving the optimization problem given in (1.11). The only difference is the specific choice of performance metric.

Now consider how we approach sequential decision problems. Assume we are minimizing costs $C(S^n, x^n)$ that depend on our decision x^n as well as other information that we carry in the state variable S^n. Decisions are made with a policy $x^n = X^\pi(S^n|\theta)$ parameterized by θ which is analogous to the statistical model $f(x^n|\theta)$ that is used to predict (or estimate) y^{n+1} before it becomes known. Our objective function would then be

$$\min_\theta \mathbb{E} \sum_{n=0}^{N-1} C(S^n, X^\pi(S^n|\theta)). \tag{1.12}$$

where $S^{n+1} = S^M(S^n, X^\pi(S^n), W^{n+1})$, and where we are given a source of the sequence $(S^0, W^1, \ldots, W^N)$.

When we compare (1.11) to (1.12), we see that both are minimizing some metric. In statistical modeling, the metric requires a dataset $(x^n, y^n)_{n=1}^N$, while our decision problem just requires a contribution (or cost) function $C(S, x)$, along with the transition function $S^{n+1} = S^M(S^n, x^n, W^{n+1})$ and a source of the exogenous information process $W^1, \ldots, W^N$. The tools for searching for θ to solve (1.11) or (1.12) are the same, but the input requirements (a training dataset, or a model of the physical problem) are quite different.

Our statistical model may take any of a wide range of forms, but they are all in the broad class of analytical models that might be a lookup table, parametric or nonparametric model. All of these classes of functions fall in just one of our four classes of policies that we refer to as policy function approximations.

Table 1.3 provides a quick comparison of some major problem classes in statistical learning, and corresponding problems in stochastic optimization. The first row compares the standard batch machine learning problem to our canonical stochastic optimization problem (for a state-independent problem). The second row compares online learning (where we have to adapt to data as it arrives) to online decision making. We use expectations in both cases since the goal is to make decisions now that work well in expectation after the next observation. Finally, the third row is making clear that we are searching for functions in both machine learning and stochastic optimization, where we use the canonical expectation-based form of the objective function. As of this writing, we feel that the research community has only begun to exploit these links, so we ask the reader to be on the lookout for opportunities to help build this bridge.

	Statistical learning	Stochastic optimization
(1)	Batch estimation: $\min_\theta \frac{1}{N}\sum_{n=1}^N (y^n - f(x^n\|\theta))^2$	Sample average approximation: $\min_{x\in\mathcal{X}} \frac{1}{N}\sum_{n=1}^N F(x, W(\omega^n))$
(2)	Online learning: $\min_\theta \mathbb{E}F(Y - f(X\|\theta))^2$	Stochastic search: $\min_\theta \mathbb{E}F(X, W)$
(3)	Searching over functions: $\min_{f\in\mathcal{F},\theta\in\Theta^f} \mathbb{E}F(Y - f(X\|\theta))^2$	Policy search: $\min_\pi \mathbb{E}\sum_{t=0}^T C(S_t, X^\pi(S_t))$

Table 1.3 Comparison of classical problems faced in statistics (left) versus similar problems in stochastic optimization (right).

1.6.3 From deterministic to stochastic optimization

Our approach shows how to generalize a deterministic problem to a stochastic one. Imagine we are making decisions $x_0, x_1, \ldots, x_t, \ldots$ over time (x_t may be a scalar or vector). Let $C_t(x_t)$ be our contribution in period t, given by

$$C_t(x_t) = p_t x_t$$

where p_t is a (known) price at time t. We also require that the following be satisfied:

$$A_t x_t = R_t, \tag{1.13}$$
$$x_t \geq 0, \tag{1.14}$$
$$R_{t+1} = B_t x_t + \hat{R}_{t+1}. \tag{1.15}$$

We wish to solve

$$\max_{x_0,\ldots,x_T} \sum_{t=0}^{T} C_t(x_t), \tag{1.16}$$

subject to equations (1.13)-(1.15).

Now assume that we wish to make $\hat{R}_{t+1}$ a random variable, which means it is not known until time $t+1$. In addition, assume that the price p_t varies randomly over time, which means we do not learn p_{t+1} until time $t+1$. These changes turn the problem into a sequential decision problem under uncertainty.

There are some simple steps to turn this deterministic optimization problem into a fully sequential one under uncertainty. To begin, we write the objective function as

$$\max_\pi \mathbb{E}\left\{\sum_{t=0}^{T} C_t(S_t, X^\pi(S_t))|S_0\right\}, \tag{1.17}$$

where $X^\pi(S_t)$ has to produce decisions that satisfy the constraints (1.13) - (1.14). Equation (1.15) is represented by the transition function $S^M(S_t, x_t, W_{t+1})$, where W_{t+1} includes $\hat{R}_{t+1}$ and the updated price p_{t+1}. We now have a properly modeled sequential decision problem.

We made the transition from deterministic optimization to a stochastic optimization formulation by making four changes:

- We replaced each occurrence of x_t with the function (policy) $X^\pi(S_t)$.
- We made the contribution function $C_t(x_t)$ depend on the state S_t to capture information (such as the price p_t) that is evolving randomly over time.
- We now take the expectation of the sum of the contributions since the evolution $S_{t+1} = S^M(S_t, x_t, W_{t+1})$ depends on the random variable W_{t+1}. It is helpful to think of the expectation operator $\mathbb{E}$ as averaging all the possible outcomes of the information process $W_1, \ldots, W_T$.
- We replace the $\max_{x_0,\ldots,x_T}$ with $\max_\pi$, which means we switch from finding the best set of *decisions*, to finding the best set of *policies*.

Care has to be taken when converting constraints for deterministic problems to the format we need when there is uncertainty. For example, we might be allocating resources and have to impose a budget over time that we can write as

$$\sum_{t=0}^{T} x_t \quad \leq \quad B,$$

where B is a budget for how much we use over all time periods. This constraint cannot be directly used in a stochastic problem since it assumes that we “decide” the variables $x_0, x_1, \ldots, x_T$ all at the same time. When we have a sequential decision problem, these decisions have to be made sequentially, reflecting the information available at each point in time. We would have to impose budget constraints recursively, as in

$$x_t \quad \leq \quad B - R_t, \tag{1.18}$$
$$R_{t+1} \quad = \quad R_t + x_t. \tag{1.19}$$

In this case, R_t would go into our state variable, and the policy $X^\pi(S_t)$ would have to be designed to reflect the constraint (1.18), while constraint (1.19) is captured by the transition function. Each decision $x_t = X^\pi(S_t)$ has to reflect what is known (captured by S_t) at the time the decision is made.

In practice, computing the expectation is hard (typically impossible) so we resort to methods known as *Monte Carlo simulation*. We introduce these methods in chapter 10. That leaves us with the usual problem of designing the policy. For this, we return to section 1.4.

All optimization problems involve a mixture of modeling and algorithms. With integer programming, modeling is important (especially for integer problems), but modeling has always taken a back seat to the design of algorithms. A testament of the power of modern algorithms is that they generally work well (for a problem class) with modest expertise in modeling strategy.

Sequential decision problems are different.

Figure 1.3 illustrates some of the major differences between how we approach deterministic and stochastic optimization problems:

Models - Deterministic models are systems of equations. Stochastic models are often complex systems of equations, numerical simulators, or even physical systems with unknown dynamics.

Figure 1.3 Deterministic vs. stochastic optimization

	Deterministic	Stochastic
Models	System of equations	Complex functions, numerical simulations, physical systems
Objective	Minimize cost	Performance metrics, risk measures
What we are searching for	Real-valued vectors	Functions (policies)
What is hard	Designing algorithms	1) Modeling 2) Designing policies

Objectives - Deterministic models minimize or maximize some well defined metric such as cost or profit. Stochastic models require that we deal with statistical performance measures and uncertainty operators such as risk. Many stochastic dynamic problems are quite complicated (think of managing supply chains, trucking companies, energy systems, hospitals, fighting diseases) and involve multiple objectives.

What we are searching for - In deterministic optimization, we are looking for a deterministic scalar or vector. In stochastic optimization, we are almost always looking for functions that we will refer to as policies.

What is hard - The challenge of deterministic optimization is designing an effective algorithm. The hardest part of stochastic optimization, by contrast, is the modeling. Designing and calibrating a stochastic model can be surprisingly difficult. Optimal policies are rare, and a policy is not optimal if the model is not correct.

1.6.4 From single to multiple agents

We close the book by extending these ideas to multiagent systems. Multiagent modeling is effective for breaking up complex systems such as supply chains (where different suppliers operate independently), as well as large transportation networks such as major carriers in trucking and rail. Multiagent modeling is essential in military applications, adversarial settings such as homeland security, oligopolies that describe markets with a small number of competitors, and a host of other applications.

Multiagent modeling is important in problems involving robots, drones and underwater vehicles, which are often used for distributed information collection. For example, a drone might be used to identify areas where wildfires are burning to guide planes and helicopters dropping fire retardant. Robots can be used to sense landmines, and underwater vehicles might be used to collect information about fish populations.

Multiagent settings almost always require learning, since there is an unavoidable compartmentalization of knowledge. This in turn introduces the dimension of communication and coordination, where coordination may be through a central agent, or where we wish to design policies that encourage agents to work together.

We use this chapter to compare our modeling strategy to the most widely used modeling and algorithmic framework for learning systems, known as partially observable Markov decision processes, or POMDPs. This is a mathematically sophisticated theory which does not lead to scalable algorithms. We are going to use our multiagent framework to clarify knowledge of the transition function, and then draw on all four classes of policies to develop practical, scalable, implementable solutions.

1.7 OUR MODELING APPROACH

The five elements in the modeling framework (section 1.3) can be used to model *any* sequential decision problem, recognizing that there are a variety of objective functions that can be used (these will be covered later). The four classes of policies in section 1.4 cover *any* method that might be used to make decisions in a sequential decision problem.

The four classes of policies are central to our modeling framework in section 1.3. We claim that *any* method used to make decisions for a sequential decision problem (and we mean *any* sequential decision problem) will be made with one of these four classes (or a hybrid of two or more). This represents a major change compared to the approaches used by the communities listed in section 1.2, which are typically associated with a particular solution approach (sometimes more than one).

We note that our approach precisely parallels that used in deterministic optimization, where people write out an optimization model (with decision variables, constraints and an objective) before searching for a solution. This is exactly what we are doing: we are writing out our model without specifying the policy, and then we search for effective policies. We call this approach:

Model first, then solve.

The generality of the four classes of policies is what allows us to separate the process of designing the model (in section 1.3) from the solution of the model (that is, finding an acceptable policy). We will first see this applied in the context of pure learning problems in chapter 7. Next, chapter 8 will present a much richer set of applications, followed by a greatly expanded version of the modeling framework given in chapter 9. Then, after touching on modeling uncertainty in chapter 10, chapter 11 revisits the four classes of policies in more detail. Chapters 12 - 19 describe each of the four classes of policies in depth.

1.8 HOW TO READ THIS BOOK

The book has been carefully designed to present topics in a logical order, with a progression from simpler to more sophisticated concepts. This section provides a guide to how to approach this material.

1.8.1 Organization of topics

The book is organized into six parts, as follows:

Part I - Introduction and foundations - We start by providing a summary of some of the most familiar canonical problems, followed by an introduction to approximation strategies which we draw on throughout the book.

- Canonical problems and applications (chapter 2) - We begin by listing a series of canonical problems that are familiar to different communities, primarily using the notation familiar to those communities. This is a chapter that can be skimmed by readers new to the general area of stochastic optimization.
- Online learning (chapter 3) - Most books on statistical learning focus on batch applications, where a model is fit to a static dataset. In our work, learning is primarily sequential, known as "online learning" in the machine learning community. Our use of online learning is purely endogenous, in that we do not need an external dataset for training.
- Introduction to stochastic search (chapter 4) - We begin with a problem we call the *basic stochastic optimization problem* which provides the foundation for most stochastic optimization problems. In this chapter we also provide examples of how some problems can be solved exactly. We then introduce the idea of solving sampled models before transitioning to adaptive learning methods, which will be the focus of the rest of the book.

Part II - State-independent problems - There is a wide range of optimization problems where the problem itself is not changing over time (for any reason). All "state-independent problems" are pure learning problems, since all that is changing as a result of our decisions is our belief about the problem. These are also known as stochastic search problems. We defer until Part III the study of more general state-dependent problems, which includes the massive class of dynamic resource allocation problems (where decisions change the allocation of resources), as well as other settings where the problem itself is evolving over time (e.g. changing weather, market prices, temperature in a room, ...).

- Derivative-based stochastic search (chapter 5) - Derivative-based algorithms represent one of the earliest adaptive methods proposed for stochastic optimization. These methods form the foundation of what is classically referred to as (derivative-based) stochastic search, or stochastic gradient algorithms.
- Stepsize policies (chapter 6) - Sampling-based algorithms need to perform smoothing between old and new estimates using what are commonly known as stepsizes (or learning rates). Stepsize policies play a critical role in derivative-based stochastic search, where the stochastic gradient determines the direction in which we move to improve a parameter vector, but the stepsize determines how far we move in the direction of the gradient.
- Derivative-free stochastic search (chapter 7) - We then transition to derivative-free stochastic search, which encompasses a variety of fields with names such as ranking and selection (for offline learning), response surface methods, and multiarmed bandit problems (for online, or cumulative reward, formulations). In this chapter that we demonstrate all four classes of policies for deciding where to next make a (typically noisy) observation of a function that we are trying to optimize.

Part III - State-dependent problems - Here we transition to the much richer class of sequential problems where the *problem* being optimized is evolving over time, which means the *problem* depends on information or parameters that are changing over time. This means the objective function and/or constraints depend on dynamic data in the state variable, where this dynamic data can depend on decisions being made (such as the inventory or location of a drone), or may just evolve exogenously (such as market prices or weather). These problems may or may not have a belief state.

- State-dependent applications (chapter 8) - We begin with a series of applications where the function is state dependent. State variables can arise in the objective function (e.g. prices), or in the constraints, which is typical of problems that involve the management of physical resources. We also illustrate problems that include evolving beliefs, which introduces the dimension of active learning (which we first encounter in chapter 7).
- Modeling sequential decision problems (chapter 9) - This chapter provides a comprehensive summary of how to model general (state-dependent) sequential decision problems.
- Uncertainty modeling (chapter 10) - To find good policies, you need a good model of uncertainty, which is arguably the most subtle dimension of modeling. In this chapter we identify different sources of uncertainty and discuss how to model them.
- Designing policies (chapter 11) - Here we provide a more comprehensive overview of the different strategies for creating policies, leading to the four classes of policies that we first introduced in part I for learning problems.

Part IV - Policies based on policy search - These chapters describe policies in the "policy search" class that have to be tuned, either in a simulator or in the field.

PFAs - Policy function approximations (chapter 12) - In this chapter we consider the use of parametric functions (plus some variations) which directly map from the state variable to a decision, without solving an imbedded optimization problem. This is the only class which does not solve an imbedded optimization problem. We search over a well-defined parameter space to find the policy that produces the best performance over time, in either offline or online settings. PFAs are well suited to problems with scalar action spaces, or low-dimensional continuous actions.

CFAs - Cost function approximations (chapter 13) - This strategy spans effective policies for solving optimal learning problems (also known as multiarmed bandit problems), to policies for high-dimensional problems that require the use of solvers for linear, integer or nonlinear programs. This policy class has been overlooked in the research literature, but is widely used (heuristically) in industry.

Part V - Policies based on lookahead approximations - Policies based on lookahead approximations are the counterpart to policies derived from policy search. Here, we design good policies by understanding the impact of a decision now on the future. We can do this by finding (usually approximately) the value of being in a state, or by planning over some horizon.

VFAs - Policies based on value function approximations - This class covers a very rich literature that span exact methods for special cases, and an extensive literature based on approximating value functions that are described by terms such as approximate dynamic programming, adaptive (or neuro) dynamic programming, and (initially) reinforcement learning. Given the depth and breadth of the work in this area, we cover this class of policy in five chapters:

  – Exact dynamic programming (chapter 14) - There are certain classes of sequential decision problems that can be solved exactly. One of the best known is characterized by discrete states and actions (known as discrete Markov decision processes), a topic we cover in considerable depth. We also briefly cover an important problem from the optimal controls literature known as *linear quadratic regulation*, as well as some simple problems that can be solved analytically.
  – Backward approximate dynamic programming (chapter 15) - Backward approximate dynamic programming parallels classical backward dynamic programming (from chapter 14), but avoids the need to enumerate states or compute expectations through Monte Carlo sampling and using machine learning to estimate value functions approximately.
  – Forward approximate dynamic programming I: The value of a policy (chapter 16) - This is the first step using machine learning methods to approximate the value of policy as a function of the starting state. This is the foundation of a broad class of methods known as approximate (or adaptive) dynamic programming, or reinforcement learning.
  – Forward approximate dynamic programming II: Policy optimization (chapter 17) - In this chapter we build on foundational algorithms such as Q-learning, value iteration and policy iteration, first introduced in chapter 14, to try to find high quality policies based on value function approximations.
  – Forward approximate dynamic programming III: Convex functions (chapter 18) - This chapter focuses on convex problems, with special emphasis on stochastic linear programs with applications in dynamic resource allocation. Here we exploit convexity to build high quality approximations of value functions.

DLAs - Policies based on direct lookahead approximations (chapter 19) - A direct lookahead policy optimizes over a horizon, but instead of optimizing the original model, we allow ourselves to introduce a variety of approximations to make it more tractable. A standard approximation is to make the model deterministic, which can work well in some applications. For those where it does not, we revisit the entire process of solving a stochastic optimization problem, but with considerably more emphasis on computation.

Part VI - Multiagent systems and learning - We close by showing how our framework can be extended to handle multiagent systems, which inherently requires learning.

- Multiagent systems and learning (chapter 20) - We start by showing how to model learning systems as two agent problems (a controlling agent observing an environment agent), and show how this produces an alternative framework to partially observable Markov decision processes (known as POMDPs). We then extend to problems with multiple controlling agents, in particular the need to model communication.

1.8.2 Organization of exercises

Each chapter is accompanied by a series of exercises at the end of the chapter, divided into the following categories:

- Review questions - These are relatively simple questions drawn directly from the chapter, without any need for creative problem solving.
- Modeling questions - These will be questions that describe an application which you then have to put into the modeling framework given above.
- Computational exercises - These are exercises that require that you perform specific calculations related to methods described in the chapter.
- Theory questions - From time to time we will pose classical theory questions. Most texts on stochastic optimization emphasize these questions. This book emphasizes modeling and computation, so theory questions play a relatively minor role.
- Problem solving questions - These questions will pose a setting and require that you go through modeling and policy design.
- Diary problem - This is a single problem of your choosing that you will use as a context to answer a question at the end of each chapter. It is like "keeping a diary" since you will accumulate answers that draw from the material throughout the book, but using the setting of a problem that is relevant to you.

Not all of these topics will be included in the exercises for each chapter.

1.8.3 How to read each chapter

This book covers a lot of material, which should not be surprising given the scope of the topic. However, it has been written to "read short." In every chapter, there are sections marked by "*" - this is our indication of material that can be skipped on a first pass. For example, chapter 9 is a substantial chapter on modeling that captures the richness of this problem class. However, we present modeling in two passes: First, for a relatively simple problem, and then for the much richer modeling framework needed for real-world problems.

Readers new to the entire topic of sequential decision problems (and by this we mean any form of dynamic programming, stochastic programming and stochastic control) should start with the relatively simpler "starter" models. It is quite easy to learn how to model the simpler problems. By contrast, complex problems can become quite rich, especially when it comes to developing stochastic models. It is important to find the problems that you are comfortable with, and then grow from there.

The book will talk at length about the four classes of policies. Of these, two are relatively simple (PFAs and CFAs) and two are much richer (VFAs and stochastic DLAs). You should not assume that you need to become an expert in all of them right away. Everyone makes decisions over time in the presence of evolving information, and the vast majority of these people have never heard of Bellman's equation (VFA-based policies). Also, while deterministic DLAs (think of navigation systems planning a path) are also relatively easy to understand, stochastic DLAs are another matter. It is much more important to get an understanding of the concept of a policy and tuning a policy (which you can do using PFAs and CFAs) than it is to jump into the more complex policies that are popular in the academic literature (VFAs and stochastic DLAs).

There are a few sections marked with ** which is our indication of mathematically advanced material. For mathematically sophisticated readers (especially those with a measure-theoretic probability background), there are many opportunities to approach this material using the full range of this training. This book is not designed for these readers, although we will occasionally hint at this material. We will say, however, that much of our notational style has been designed with an understanding of how probabilists (in particular) think of and approach sequential decision problems. This book will lay a proper foundation for readers who want to use this as a launching pad into more theoretical research.

1.9 BIBLIOGRAPHIC NOTES

Section 1.2 - We defer to chapter 2 for a discussion of the different communities of stochastic optimization, and review the literature there. It cannot be emphasized enough how much our universal framework draws on all these communities.

Section 1.3 - We first articulated the five elements of the universal framework in Powell (2011) (Chapter 5, which has always been available at `http://adp.princeton.edu`), which built on the initial model from the first edition which had six elements (Powell (2007)). Our framework draws heavily from the framework that has long been used in optimal control (there are many books, but see Lewis et al. (2012) which is a popular reference in this field), but there are some differences. Our framework is compared to the optimal control framework and that used in Markov decision processes (and now reinforcement learning) in Powell (2019*b*). Some key differences is that the optimal control framework, which is originally based on deterministic control, often optimizes over the controls $u_0, u_1, \ldots, u_T$, even when the problem is stochastic. Our notation makes it explicit that if the problem is stochastic, u_t is a function which we call a policy (the controls people will call it a control law), and we always optimizes over policies π.

Section 1.4 - Powell (2011) appears to be the first published reference to "four classes of policies" for solving dynamic programs, but it did not list the four classes used here (one class was myopic policies, and cost function approximations were overlooked). The first reference to list the four classes of policies used here was the tutorial Powell (2014*a*) (*Clearing the Jungle of Stochastic Optimization*, without recognizing that the four classes can (and should) be divided into two major strategies. The first paper to identify the two strategies of "policy search" and "lookahead policies" was given in the tutorial Powell (2016*a*).

EXERCISES

Review questions

1.1 What are the three classes of state variables?

1.2 What are the five elements of a sequential decision problem?

1.3 What is the price of simplicity? Give an example, either from the chapter or a problem of your own choosing..

1.4 What are the two strategies for designing policies for sequential decision problems? Briefly describe the principles behind each one.

1.5 What are the four classes of policies? Briefly describe each one.

Modeling questions

1.6 Pick three examples of sequential decision problems. Provide a brief narrative describing the context, and list a) the decision being made, b) information that arrives after the decision is made that is likely to be relevant to the decision, and c) at least one metric that can be used to evaluate how well the decision has performed.

1.7 For each of the three types of state variables, do the following:

a) Give three examples of physical state variables.

b) Give three examples of information about parameters or quantities that we know perfectly, but which would not be considered a physical state variable.

c) Give three examples of parameters or quantities that we would not know perfectly, but could approximate with a probability distribution.

1.8 Section 1.3 shows how to model a simple inventory problem. Repeat this model assuming that we sell our product at a price p_t that changes from time period to time period according to the equation

$$p_{t+1} = p_t + \varepsilon_{t+1},$$

where ε_{t+1} is a normally distributed random variable with mean 0 and variance σ^2.

1.9 Consider an asset selling problem where you need to decide when to sell an asset. Let p_t be the price of the asset if it is sold at time t, and assume that you model the evolution of the price of the asset using

$$p_{t+1} = p_t + \theta(p_t - 60) + \varepsilon_{t+1},$$

We assume that the noise terms $\varepsilon_t,\ t = 1, 2, \ldots$ are independent and identically distributed over time, where $\varepsilon_t \sim N(0, \sigma_\varepsilon^2)$. Let

$$R_t = \begin{cases} 1 & \text{If we are still holding the asset at time } t, \\ 0 & \text{Otherwise.} \end{cases}$$

Further let

$$x_t = \begin{cases} 1 & \text{If we sell the asset at time } t, \\ 0 & \text{Otherwise.} \end{cases}$$

Of course, we can only sell the asset if we are still holding it. We now need a rule for deciding if we should sell the asset. We propose

$$X^\pi(S_t|\rho) = \begin{cases} 1 & \text{If } p_t \geq \bar{p}_t + \rho \text{ and } R_t = 1, \\ 0 & \text{Otherwise.} \end{cases}$$

where

$$\begin{aligned} S_t &= \text{The information we have available to make a decision (we have to, design this),} \\ \bar{p}_t &= .9\bar{p}_{t-1} + .1p_t. \end{aligned}$$

a) What are the elements of the state variable S_t for this problem?

b) What is the uncertainty?

c) Imagine running a simulation in a spreadsheet where you are given a sample realization of the noise terms over T time periods as $(\hat{\varepsilon})_{t=1}^T = (\hat{\varepsilon}_1, \hat{\varepsilon}_2, \ldots, \hat{\varepsilon}_T)$. Note that we treat $\hat{\varepsilon}_t$ as a number, such as $\hat{\varepsilon}_t = 1.67$ as opposed to ε_t which is a normally distributed random variable. Write an expression for computing the value of the policy $X^\pi(S_t|\rho)$ given the sequence $(\hat{\varepsilon})_{t=1}^T$. Given this sequence, we could evaluate different values of ρ, say $\rho = 0.75, 2.35$ or 3.15 to see which performs the best.

d) In reality, we are not going to be given the sequence $(\hat{\varepsilon})_{t=1}^T$. Assume that $T = 20$ time periods, and that

$$\begin{aligned} \sigma_\varepsilon^2 &= 4^2, \\ p_0 &= \$65, \\ \theta &= 0.1. \end{aligned}$$

Write out the value of the policy as an expectation (see section 1.3).

e) Develop a spreadsheet to create 10 sample paths of the sequence $(\varepsilon_t),\ t = 1, \ldots, 20)$ using the parameters above. You can generate a random observation of ε_t using the function `NORM.INV(RAND(),0,`σ`)`. Let the performance of our decision rule $X^\pi(S_t|\rho)$ be given by the price that it decides to sell (if it decides to sell), averaged over all 10 sample paths. Now test $\rho = 1, 2, 3, 4, ..., 10$ and find the value of ρ that seems to work the best.

f) Repeat (e), but now we are going to solve the problem

$$\max_{x_0,\ldots,x_T} \mathbb{E} \sum_{t=0}^{T} p_t x_t.$$

We do this by picking the time t when we are going to sell (that is, when $x_t = 1$) before seeing any information. Evaluate the solutions $x_2 = 1, x_4 = 1, \ldots, x_{20} = 1$. Which is best? How does its performance compare to the performance of $X^\pi(S_t|\rho)$ for the best value of ρ?

g) Finally, repeat (f), but now you get to see all the prices and then pick the best one. This is known as a *posterior bound* because it gets to see all the information in the future to make a decision now. How do the solutions in parts (e) and (f) compare to the posterior bound? (There is an entire field of stochastic optimization that uses this strategy as an approximation.)

h) Classify the policies in (e), (f) and (g) (yes, (g) is a class of policy) according to the classification described in section 1.5 of the text.

Problem solving questions

1.10 The inventory problem describes a policy where an order is made if the inventory falls below θ^{min}, where we order up to θ^{max}. Which of the four classes does this represent? Write out the objective function we would have to use to find the best value of θ.

Diary problem

The diary problem is a single problem you chose (see chapter 1 for guidelines). Answer the following for your diary problem.

1.11 For this chapter, you need to pick a problem context. The ideal problem is one with some richness (e.g. different types of decisions and sources of uncertainty), but the best problem is one that you are familiar with, or have a special interest in. To bring out the richness of our modeling and algorithmic framework, it would help if your sequential decision problem involved learning in some form. For now, prepare a 1-2 paragraph summary of the context. You will be providing additional details in later chapters.