REINFORCEMENT LEARNING AND STOCHASTIC OPTIMIZATION

A unified framework for sequential decisions

Warren B. Powell

August 22, 2021

WILEY-INTERSCIENCE

A JOHN WILEY & SONS, INC., PUBLICATION

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

Library of Congress Cataloging-in-Publication Data:

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

10 9 8 7 6 5 4 3 2 1

CHAPTER 4

INTRODUCTION TO STOCHASTIC SEARCH

Our most basic optimization problem can be written

$$\max_{x \in \mathcal{X}} \mathbb{E}_W F(x, W), \tag{4.1}$$

where x is our decision and W is any form of random variable. A simple example of this problem is the newsvendor problem which we might write

$$\max_{x \in \mathcal{X}} \mathbb{E}_W \big(p \min(x, W) - cx\big),$$

where x is a quantity of product we order at cost c, W is the demand, and we sell the smaller of x and W to the market at a price p.

This problem is the one most identified with the field that goes under the name of "stochastic search." It is typically presented as a "static" stochastic optimization problem because it consists of making a single decision x, then observing an outcome W allowing us to assess the performance $F(x, W)$, at which point we stop. However, this all depends on how we interpret "$F(x, W)$," "x," and "W."

For example, we can use $F(x, W)$ to represent the results of running a simulation, a set of laboratory experiments, or the profits from managing a fleet of trucks. The input x could be the set of controllable inputs that govern the behavior of the simulator, the materials used in the laboratory experiments, or the size of our fleet of trucks. In addition, x could also be the parameters of a policy for making decisions, such as the order-up-to parameters $\theta = (\theta^{min}, \theta^{max})$ in the inventory problem we introduced in section 1.3 (see equation (1.5)).

Reinforcement Learning and Stochastic Optimization. By Warren B. Powell

At the same time, the variable W could be the sequence $W = (W^1, W^2, \ldots, W^N)$ representing the events within the simulator, the outcomes of individual laboratory experiments, or the loads that arrive while dispatching our fleet of trucks. Finally, $F(x, W)$ could be the performance of the simulation or set of experiments or our fleet of trucks over a week. This means that we could write $F(x, W)$ as

$$F(x, W) = \sum_{t=0}^{T} C(S_t, X^\pi(S_t))$$

where "x" is our policy π and our state variable evolves according to $S_{t+1} = S^M(S_t, X^\pi(S_t), W_{t+1})$ given the sequence $W = (W_1, \ldots, W_T)$.

While equation (4.1) is the most standard way of writing this problem, we are going to use the expanded form as our default statement of the problem, which is written

$$\max_{x \in \mathcal{X}} \mathbb{E}_{S^0} \mathbb{E}_{W|S^0} \{F(x, W)|S^0\}, \tag{4.2}$$

which allows us to express the expectation on information in an initial state S^0, which can include deterministic parameters as well as probabilistic information (which we need when we use Bayesian belief models). For example, our problem may depend on an unknown physical parameter $\boldsymbol{\theta}$ which we believe may be one of a set $\theta_1, \ldots, \theta_K$ with probability $p_k^0 = \mathbb{P}[\boldsymbol{\theta} = \theta_k]$.

There are three core strategies for solving our basic stochastic optimization problem (4.2):

Deterministic methods - There are some problems with sufficient structure that allows us to compute any expectations exactly, which reduces a stochastic problem to a deterministic one (more precisely, it reduces a stochastic problem to one that can be solved using deterministic mathematics). In some cases problems can be solved analytically, while others require the use of deterministic optimization algorithms.

Sampled approximations - This is a powerful and widely used approach for turning computationally intractable expectations into tractable ones. We note that sampled problems, while solvable, may not be easily solvable, and as a result have attracted considerable research interest, especially for problems where x is high-dimensional and possibly integer. However, we will also make the argument that a sampled stochastic problem is, fundamentally, a problem that can be solved with deterministic mathematics, although the analysis of the properties of the resulting solution may require stochastic tools.

Adaptive learning methods - The vast majority of stochastic optimization problems will end up requiring adaptive learning methods, which are fundamentally stochastic, and require stochastic tools. These are the approaches that will attract most of our attention in this volume. We will be particularly interested in the performance of these methods using finite learning budgets.

We begin our presentation by discussing different perspectives of our basic stochastic optimization problem, which encompasses fully sequential problems when we interpret "x" as a policy π. We then observe that there are examples of stochastic optimization problems that can be solved using standard deterministic methods, either by directly exploiting the structure of the uncertainty (which allows us to compute the expectation directly), or by using the powerful idea of sampled models.

We then close by setting up some preliminary discussions about adaptive learning methods, which are then discussed in more detail in chapters 5, 6, and 7. As we point out below, adaptive learning methods represent a form of sequential decision problem where the state variable S^n captures only what we know (or believe). There is no other physical process (such as inventory) or informational process (such as a time series) which links decisions over time. We defer until Part III of the book the handling of these more complex problems.

The perspectives presented in this chapter appear to be new, and set up the approach we use throughout the rest of the book.

4.1 ILLUSTRATIONS OF THE BASIC STOCHASTIC OPTIMIZATION PROBLEM

There is no shortage of applications of our basic stochastic optimization problem. Some examples that illustrate applications in different settings include:

EXAMPLE 4.1

Engineering design - Here x is the design of an airplane wing where we have to create a design that minimizes costs over a range of different conditions. We can learn from numerical simulations, laboratory strength tests, and examining actual aircraft for stress fractures.

EXAMPLE 4.2

Let $(y^n, x^n)_{n=1}^N$ be a set of explanatory variables x^n and response variables y^n. We would like to fit a statistical model (this might be a linear parametric model, or a neural network) where θ is the parameters (or weights) that characterize the model. We want to find θ that solves

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

This problem, which is very familiar in statistics, is a sampled approximation of

$$\min_\theta \mathbb{E}(Y - f(X|\theta))^2,$$

where X is a random input and Y is the associated random response.

EXAMPLE 4.3

We would like to design an energy system where R is a vector of energy investments (in wind farms, solar fields, battery storage, gas turbines), which we have to solve subject to random realizations of energy from wind and solar (which we represent using the vector W) defined over a year. Let $C^{cap}(R)$ be the capital cost of these investments, and let $C^{op}(R, W)$ be the net operating revenue given W (computed from a numerical simulator). Now we want to solve

$$\max_R \mathbb{E}(-C^{cap}(R) + C^{op}(R, W)).$$

EXAMPLE 4.4

A bank uses a policy $X^\pi(S|\theta)$ that covers how much to move into or out of cash given the state S which describes how much cash is on hand, the forward price/earnings ratio of the S&P 500 (an important index of the stock market), and current 10-year bond rates. The vector θ captures upper and lower limits on each variable that triggers decisions to move money into or out of cash. If $C(S_t, X^\pi(S_t|\theta), W_{t+1})$ is the cash flow given the current state S_t and the next-period returns W_{t+1}, then we want to find the policy control parameters θ that solves

$$\max_\theta \mathbb{E} \sum_{t=0}^{T} e^{-rt} C(S_t, X^\pi(S_t|\theta), W_{t+1}).$$

Each of these examples involve making some decision: The design of the airplane wing, the model parameter θ, the energy investment R, or the parameters θ of a cash transfer policy. In each case, we have to choose a design either to optimize a deterministic function, a sampled approximation of a stochastic problem, or by adaptive learning (either from a simulator, laboratory experiments or field observations).

While there are some settings where we can solve (4.2) directly (possibly with an approximation of the expectation), most of the time we are going to turn to iterative learning algorithms. We will start with a state S^n that captures our belief state about the function $F(x) = \mathbb{E}\{F(x, W)|S_0\}$ after n experiments (or observations). We then use this knowledge to make a decision x^n after which we observe W^{n+1} which leads us to a new belief state S^{n+1}. The problem is designing a good rule (or policy) that we call $X^\pi(S^n)$ that determines x^n. For example, we might want to find the best answer that we can with a budget of N iterations.

We pose this as one of finding the best policy to determine a solution $x^{\pi,N}$, which is a random variable that might depend on any initial distributions S^0 (if necessary), and the sequence of observations $(W^1, \ldots, W^N)$ that, combined with our policy (algorithm) π produces $x^{\pi,N}$. We can think of $(W^1, \ldots, W^N)$ as the training observations. Then, we let $\widehat{W}$ be observations we make to perform testing of $x^{\pi,N}$. This can all be written (using our expanded form of the expectation) as

$$\max_\pi \mathbb{E}_{S^0} \mathbb{E}_{W^1,\ldots,W^N|S^0} \mathbb{E}_{\widehat{W}|S^0} \{F(x^{\pi,N}, \widehat{W})|S_0\}. \tag{4.3}$$

We ask the reader to contrast our original version of this problem in equation (4.1) with (4.3). The version in (4.1) can be found throughout the research literature. But the version in (4.3) is the problem we are actually solving in practice.

The formulations in (4.1), (4.2) and (4.3) all focus on finding the best decision (or design) to maximize some function. We refer to these as *final reward* formulations. This distinction is important when we use adaptive learning policies $X^\pi(S)$, since this involves optimizing using intelligent trial and error.

When we use adaptive learning (which is a widely used strategy), then we have to think about our attitude toward the intermediate decisions x^n for $n < N$. If we have to "count" the results of these intermediate experiments, then we would write our objective as

$$\max_\pi \mathbb{E}_{S^0} \mathbb{E}_{W^1,\ldots,W^N|S^0} \left\{ \sum_{n=0}^{N-1} F(X^\pi(S^n), W^{n+1})|S_0 \right\}. \tag{4.4}$$

When we are using an adaptive learning strategy, we are going to refer to (4.3) as the *final reward* formulation, while the objective function in (4.4) is the *cumulative reward* formulation.

It is not by accident that the function $F(x, W)$ does not depend on our evolving state variable S^n (or S_t), while the policy $X^\pi(S^n)$ does. We are assuming here that our function $F(x, W)$ itself is not evolving over time; all that is changing are the inputs x and W. When we want our performance to depend on the state, we will use $C(S, x)$ to indicate this dependence.

The number of applications that fit the basic model given in equation (4.2) is limitless. For discussion purposes, it is helpful to recognize some of the major problem classes that arise in this setting:

- Discrete problems, where $\mathcal{X} = \{x_1, \ldots, x_M\}$. Examples might be where x_m is a set of features for a product, catalysts for a type of material, drug cocktails, or even paths over a network.

- Concave problems, where $F(x, W)$ is concave in x (often x is a vector in this case).

- Linear programs, where $F(x, W)$ is a linear cost function and $\mathcal{X}$ is a set of linear constraints.

- Continuous, nonconcave problems, where x is continuous.

- Expensive functions - There are many settings where computing $F(x, W)$ involves running time consuming computer simulations or laboratory experience that may take hours to days to weeks, or field experiments that may take weeks or months).

- Noisy functions - There are many problems where the measurement or observation errors in the function are extremely high, which introduces the need to develop methods that manage this level of noise.

For these problems, the decision x may be finite, continuous scalar, or a vector (that may be discrete or continuous).

As we progress, we are going to see many instances of (4.1) (or (4.2)) where we sequentially guess at a decision x^n, then observe W^{n+1}, and use this information to make a better guess x^{n+1}, with the goal of solving (4.1). In fact, before we are done, we are going to show that we can reduce our formulations of fully sequential problems such as our inventory problem to the same form as in (4.3) (or (4.4)). For this reason, we have come to refer to (4.2) as the basic stochastic optimization model.

4.2 DETERMINISTIC METHODS

There are a handful of stochastic optimization problems that can be solved to optimality using purely deterministic methods. We are going to provide a brief illustration of some examples as an illustration, but in practice, exact solutions of stochastic problems will be quite rare. The discussion in this section is relatively advanced, but the point is important since the research community often overlooks that there are a number of so-called "stochastic optimization problems" that are solved using purely deterministic mathematics.

4.2.1 A "stochastic" shortest path problem

In section 2.3.3, we introduced a stochastic shortest path problem where a traveler arriving to node i would see the sample realizations of the random costs C_{ij} to each node j that can be reached from i. Assume that on the n^{th} day we arrive to node i and observe the sample realization $\hat{c}_{ij}^n$ of the random variable C_{ij}. We would then get a sampled observation of the value of being at node i from

$$\hat{v}_i^n = \min_{j \in \mathcal{I}_i^+} \left(\hat{c}_{ij}^n + \overline{V}_j^{n-1} \right),$$

where $\mathcal{I}_i^+$ is the set of all nodes that we can reach from node i. Now assume that we do not see the sample realization of the random variable C_{ij} before we make our decision. Assume we have to make the decision before we see the realization. In this case, we have to use the expected value $\bar{c}_{ij} = \mathbb{E}C_{ij}$, which means we are solving

$$\begin{aligned} \hat{v}_i^n &= \min_{j \in \mathcal{I}_i^+} \mathbb{E}\left(C_{ij} + \overline{V}_j^{n-1} \right), \\ &= \min_{j \in \mathcal{I}_i^+} \left(\bar{c}_{ij} + \overline{V}_j^{n-1} \right), \end{aligned}$$

which is just what we would solve if we had a deterministic shortest path problem. In other words, when we have a linear objective, if we have to make decisions before we see information, then the resulting problem reduces to a deterministic optimization problem which can (generally) be solved exactly.

The key difference between this "stochastic" shortest path problem and the one in section 2.3.3 is how information is revealed. The problem in section 2.3.3 is harder (and more interesting) because information is revealed just before we make the decision of the next link to traverse. Here, information is revealed after we make a decision, which means decisions have to be made using distributional information. Since the problem is linear in the costs, then all we need are the means, turning our stochastic problem into a deterministic problem.

4.2.2 A newsvendor problem with known distribution

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

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

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

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

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

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

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

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

Under the (reasonable) assumption that the unit purchase cost c is less than the sales price p, we see that the optimal solution x^* corresponds to the point where the probability that x^* is less than the demand W is the ratio of the cost over the price. Thus if the cost is low, the probability that the demand is greater than the supply (which means we lose sales) should be low.

Equation (4.7) gives the optimal solution of the newsvendor problem. It requires that we know the distribution of demand, and also requires that we be able to take the expectation of the gradient and solve for the optimal probability analytically. Not surprisingly, these conditions are rarely met in practice.

4.2.3 Chance-constrained optimization

There are some problems where we can compute the expectation exactly, but the result is (typically) a nonlinear problem that can only be solved numerically. A good illustration of this is a method known as chance constrained programming, which is itself a rich area of study. A classical formulation (which we first saw in section 2.1.12) poses the problem

$$\min_x f(x), \tag{4.8}$$

subject to the constraint

$$p(x) \quad \leq \quad \alpha, \tag{4.9}$$

where

$$p(x) \quad = \quad \mathbb{P}[C(x, W) \geq 0] \tag{4.10}$$

is the probability that a constraint violation, captured by $C(x, W)$, is violated. Thus, $C(x, W)$ might be the uncovered demand for energy, or the degree to which two driverless cars get closer than an allowed tolerance. If we can compute $p(x)$ (analytically or numerically), we can draw on powerful nonlinear programming algorithms to solve (4.8) directly.

4.2.4 Optimal control

In section 2.1.4, we formulated an optimal control problem of the form

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

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

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

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

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

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

Now assume that the loss function has the quadratic form

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

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

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

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

This solution depends on three critical features of this problem:

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

Despite these limitations, this result has proved quite important for many problems in engineering.

4.2.5 Discrete Markov decision processes

As with the field of stochastic control, there is an incredibly rich body of literature that has grown up around the basic problem of discrete dynamic programs, a problem that we first introduced in section 2.1.3, but address in much more depth in chapter 14. Imagine that we have a contribution $C(s, x)$ when we are in state $s \in \mathcal{S}$ and take discrete action $x \in \mathcal{X} = \{x_1, \ldots, x_M\}$, and a one-step transition matrix $P(s'|s, x)$ which gives the probability that we evolve to state $S_{t+1} = s'$ given that we are in state $S_t = s$ and take action x. It is possible to show that the value of being in a state $S_t = s$ at time t is given by

$$V_t(S_t) = \max_{x \in \mathcal{X}} \left(C(S_t, x) + \sum_{s' \in \mathcal{S}} P(s'|S_t, x) V_{t+1}(s') \right). \tag{4.13}$$

We can compute (4.13) if we start at time T with some initial value, say $V_T(s) = 0$, and then step backward in time. This produces the optimal policy $X_t^*(S_t)$ given by

$$X_t^*(S_t) = \arg\max_{x \in \mathcal{X}} \left(C(S_t, x) + \sum_{s' \in \mathcal{S}} P(s'|S_t, x) V_{t+1}(s') \right). \tag{4.14}$$

Again, we have found our optimal policy purely using deterministic mathematics. The critical element of this formulation is the assumption that the one-step transition matrix $P(s'|s, x)$ is known (and computable). This requirement also requires that the state space $\mathcal{S}$ and action space $\mathcal{X}$ be discrete and not too large.

4.2.6 Remarks

These are a representative list of the very small handful of stochastic optimization problems that can be solved either analytically or numerically using deterministic methods. While we have not covered every problem that can be solved this way, the list is not long. This is not to minimize the importance of these results, which sometimes serve as the foundation for algorithms for more general problems.

Often, the most difficult aspect of a stochastic optimization problem is the expectation (or other operators such as risk metrics to deal with uncertainty). It should not be surprising, then, that the techniques used to solve more general stochastic optimization problems tend to focus on simplifying or breaking down the representation of uncertainty. The next section introduces the concept of sampled models, a powerful strategy that is widely used in stochastic optimization. We then transition to a discussion of adaptive sampling-based methods that is the focus of most of the rest of this book.

4.3 SAMPLED MODELS

One of the most powerful and widely used methods in stochastic optimization is to replace the expectation in the original model in equation (4.1), which is typically computationally intractable, with a sampled model. For example, we might represent the possible values of W (which might be a vector) using the set $\hat{\mathcal{W}} = \{w^1, \ldots, w^N\}$. Assume that each w^n can happen with equal probability. We can then approximate the expectation in equation (4.1) using

$$\mathbb{E}F(x, W) \approx \bar{F}(x) = \frac{1}{N}\sum_{n=1}^{N} F(x, w^n).$$

The use of samples can transform intractable expectations into relatively easy calculations. More difficult is understanding the properties of the resulting approximation $\bar{F}(x)$, and the effect of sampling errors on the solution of

$$\max_x \bar{F}(x). \tag{4.15}$$

These questions have been addressed under the umbrella of a method called the *sample average approximation*, but the idea has been applied in a variety of settings.

Our newsvendor problem is a nice example of a stochastic optimization problem where the uncertain random variable is a scalar, but real applications can feature random inputs W that are very high dimensional. A few examples illustrate how large random variables can be:

■ **EXAMPLE 4.5**

A blood management problem requires managing eight blood types, which can be anywhere from 0 to 5 weeks old, and may or may not be frozen, creating $6 \times 8 \times 2 = 96$ blood types. Patients needing blood create demands for eight different types of blood. Each week there are random supplies (96 dimensions) and random demands (8 dimensions), creating an exogenous information variable W_t with 104 dimensions.

■ **EXAMPLE 4.6**

A freight company is moving parcels among 1,000 different terminals. Since each parcel has an origin and destination, the vector of new demands has 1,000,000 dimensions.

■ **EXAMPLE 4.7**

Patients arriving to a doctor's office may exhibit as many as 300 different characteristics. Since each patient may or may not have any of these characteristics, there are as many as $2^{300} \sim 2 \times 10^{90}$ different types of patients (far more than the population of planet Earth!)

This section provides a brief introduction to what has evolved into an incredibly rich literature. We start by addressing the following questions:

- How do we formulate a sampled model?
- How good is the quality of the sampled solution (and how fast does it approach the optimal as K is increased)?
- For large problems (high dimensional x), what are strategies for solving (4.15)?
- Again for large problems, what are the best ways of creating the sample $w^1, \ldots, w^N$?

We are going to return to sampled models from time to time since they represent such a powerful strategy for handling expectations.

4.3.1 Formulating a sampled model

Assume that W is one of these multidimensional (and possibly very high dimensional) random variables. Further assume that we have some way of generating a set of samples $w^1, \ldots, w^N$. These may be generated from a known probability distribution, or perhaps from a historical sample. We can replace our original stochastic optimization problem (4.1) with

$$\max_x \frac{1}{N} \sum_{n=1}^{N} F(x, w^n). \tag{4.16}$$

Solving (4.16) as an approximation of the original problem in (4.1) is known as the *sample average approximation*. It is important to realize that both our original stochastic optimization problem (4.1) and the sampled problem (4.16) are deterministic optimization problems. The challenge is computation.

Below we illustrate several uses of sampled models.

4.3.1.1 A sampled stochastic linear program As with W, the decision variable x can be a scalar, or a very high-dimensional vector. For example, we might have a linear program where we are optimizing the flows of freight x_{ij} from location i to location j by solving

$$\min_x F(x, W) = \sum_{i,j \in \mathcal{I}} c_{ij} x_{ij},$$

subject to a set of linear constraints

$$\begin{aligned} Ax &= b, \\ x &\geq 0. \end{aligned}$$

A common application of this model arises when making a decision to allocate a resource such as blood inventories from central blood banks to hospitals, before knowing the results of weekly donations of blood, and the schedule of operations that need blood, at each hospital for the following week.

Now assume that the random information is the cost vector c (which might reflect the types of surgeries that require blood transfusions), the coefficient matrix A (which might capture travel times between inventory locations and hospitals), and the vector b (which captures blood donations and surgeries). Thus, $W = (A, b, c)$.

If we have one sample of W, then we have a straightforward linear program which may not be too hard to solve. But now imagine that we have $N = 100$ samples of the data, given by $(A^n, b^n, c^n)_{n=1}^N$. We could then solve

$$\min_x \frac{1}{N} \sum_{n=1}^{N} c_{ij}^n x_{ij},$$

subject to, for $n = 1, \ldots, 100$,

$$\begin{aligned} A^n x &= b^n, \\ x &\geq 0. \end{aligned}$$

If we choose a sample of $N = 100$ outcomes, then our sampled problem in (4.16) becomes a linear program that is 100 times larger (remember we have just one vector x, but 100 samples of A, b and c). This may be computationally difficult (in fact, coming up with a single vector x that is feasible for all 100 samples of the data (A, b, c) may not even be possible).

4.3.1.2 Sampled chance-constrained models

We can use our idea of sampling to solve chance constrained programs. We begin by noting that a probability is like an expectation. Let $\mathbb{1}_{\{E\}} = 1$ if event E is true. Then we can write our probability as

$$\mathbb{P}[C(x, W) \leq 0] = \mathbb{E}_W \mathbb{1}_{\{C(x,W)\leq 0\}}.$$

We can replace the chance constraint in (4.10) with a sampled version, where we basically average the random indicator variable to obtain

$$\mathbb{P}[C(x, W) \leq 0] \approx \frac{1}{N} \sum_{n=1}^{N} \mathbb{1}_{\{C(x,w^n)\leq 0\}}.$$

If x is discrete, then each $\mathbb{1}_{\{C(x,w^n)\}}$ can be calculated in advance for each w^n. If x is continuous, then it is likely that these indicator functions can be written as linear constraints.

4.3.1.3 Sampled parametric models

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

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

We want to pick a price that maximizes revenue using

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

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

$$\max_p \mathbb{E}_\theta pD(p|\theta), \tag{4.17}$$

but computing the expectation may be hard. However, perhaps we are willing to say that θ may take on one of a set of values $\theta^1, \ldots, \theta^N$, each with probability q^n. Now we can solve

$$\max_p \sum_{n=1}^{N} pD(p|\theta^n)q^n. \tag{4.18}$$

Whereas equation (4.17) may be intractable, (4.18) may be much easier.

Both (4.16) and (4.18) are examples of sampled models. However, the representation in (4.16) is used in settings where $(w^1, \ldots, w^N)$ is a sample drawn from a typically large (often infinite) set of potential outcomes. The model in (4.18) is used when we have an uncertain belief about parameters, and are using the set $\theta^1, \ldots, \theta^N$, with a probability vector q that may evolve over time.

4.3.2 Convergence

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

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

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

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

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

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

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

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

These results tell us that we will eventually achieve the best possible objective function (note that there may be more than one optimal solution). The most interesting and important result is the rate at which we achieve this result. We start by assuming that our feasible

region $\mathcal{X}$ is a set of discrete alternatives $x_1, \ldots, x_M$. This might be a set of discrete choices (e.g. different product configurations or different drug cocktails), or a discretized continuous parameter such as a price or concentration. Or, it could be a random sample of a large set of possibly vector-valued decisions.

Now, let ϵ be some small value (whatever that means). The amazing result is that as N increases, the probability that the optimal solution to the approximate problem, X^N, is more than ϵ from the optimal shrinks at an *exponential rate*. We can write this statement mathematically as

$$\mathbb{P}[F(x^N) < F(x^*) - \epsilon] < |\mathcal{X}|e^{-\eta N}, \tag{4.19}$$

for some constant $\eta > 0$. What equation (4.19) is saying is that the probability that the quality of our estimated solution x^N, given by $F(x^N)$, is more than ϵ away from the optimal $F(x^*)$, decreases at an exponential rate $e^{-\eta N}$ with a constant, $|\mathcal{X}|$, that depends on the size of the feasible region. The coefficient $\mathcal{X}$ is quite large, of course, and we have no idea of the magnitude of η. However, the result suggests that the probability that we do worse than $F(x^*) - \epsilon$ (remember that we are maximizing) declines exponentially with the same size N, which is comforting.

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

$$\mathbb{P}[F(x^N) < F(x^*) - \epsilon] < Ce^{-\eta N}, \tag{4.20}$$

for given constants $C > 0$ and $\eta > 0$. Note that unlike (4.19), equation (4.20) does not depend on the size of the feasible region, although the practical effect of this property is unclear.

The convergence rate results (4.19) (for discrete decisions) or (4.20) (for convex functions) tell us that as we allow our sample size N to increase, the optimal objective function $F(x^N)$ approaches the optimal solution $F(x^*)$ at an exponential rate, which is a very encouraging result. Of course, we never know the parameters η, or C and η, so we have to depend on empirical testing to get a sense of the actual convergence rate. However, knowing that the rate of convergence is exponential (regardless of the values of C and η) is exceptionally important. We would also note that while solving a sampled model is fundamentally deterministic (since the sample gives us an approximate expectation that can be calculated exactly), the analysis of the rate of convergence with respect to the sample size N is pure stochastic analysis.

The exponential convergence rates are encouraging, but there are problems such as linear (or especially integer) programs that are computationally challenging even when $N = 1$. We are going to see these later in the context of models where we use sampling to look into the future. There are two computational issues that will need to be addressed:

Sampling - Rather than just doing random sampling to obtain $W^1, \ldots, W^N$, it is possible to choose these samples more carefully so that a smaller sample can be used to produce a more realistic representation of the underlying sources of uncertainty.

Decomposition - The sampled problem (4.16) can still be quite large (it is N times bigger than the problem we would obtain if we just used expectations for uncertain quantities), but the sampled problem has structure we can exploit using decomposition algorithms.

We defer until chapter 10 a more complete description of sampling methods to represent uncertainty. We then wait until chapter 19 to show how decomposition methods can be used in the setting of lookahead policies.

4.3.3 Creating a sampled model

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

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

In some instances we have an interest in creating a reasonable representation of the underlying uncertainty with the smallest possible sample. For example, imagine that we are replacing the original problem $max_x \mathbb{E}F(x, W)$ with a sampled representation

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

Now imagine that x is a (possibly large) vector of integer variables, which might arise if we are trying to schedule aircraft for an airline, or to design the location of warehouses for a large logistics network. In such settings, even a deterministic version of the problem might be challenging, whereas we are now trying to solve a problem that is N times as large. Instead of solving the problem over an entire sample $W^1, \ldots, W^N$, we may be interested in using a good representative subset $(W^j),\ j \in \mathcal{J}$. Assume that W^n is a vector with elements $W^n = (W_1^n, \ldots, W_k^n, \ldots, W_K^n)$. One way to compute such a subset is to compute a distance metric $d^1(n, n')$ between W^n and $W^{n'}$ which we might do using

$$d^1(n, n') = \sum_{k=1}^{K} |W_k^n - W^{n'_k}|.$$

This would be called an "L_1-norm" because it is measuring distances by the absolute value of the distances between each of the elements. We could also use an "L_2-norm" by computing

$$d^2(n, n') = \sqrt{\left(\sum_{k=1}^{K} (W_k^n - W^{n'_k})^2\right)}.$$

The L_2-norm puts more weight on large deviations in an individual element, rather than a number of small deviations spread over many dimensions. We can generalize this metric using

$$d^p(n, n') = \left(\sum_{k=1}^{K} (W_k^n - W^{n'_k})^p\right)^{\frac{1}{p}}.$$

However, other than the L_1 and L_2 metrics, the only other metric that is normally interesting is the L_∞-norm, which is the same as setting $d^\infty(n, n')$ equal to the absolute value of the largest difference across all the dimensions.

Using the distance metric $d^p(n, n')$, we choose a number of clusters J and then organize the original set of observations $W^1, \ldots, W^n$ into J clusters. This can be done using a popular family of algorithms that go under names such as k-means clustering or k-nearest neighbor clustering. There are different variations of these algorithms which can be found in standard libraries. The core idea in these procedures can be roughly described as:

Step 0 - Use some rule to pick J centroids. This might be suggested by problem structure, or you can pick J elements out of the set $W^1, \ldots, W^N$ at random.

Step 1 - Now step through each $W^1, \ldots, W^N$ and assign each one to the centroid that minimizes the distance $d^p(n, j)$ over all centroids $j \in \mathcal{J}$.

Step 2 - Find the centroids of each of the clusters and return to Step 1 until you find that your clusters are the same as the previous iteration (or you hit some limit).

A nice feature of this approach is that it can be applied to high-dimensional random variables W, as might arise when W represents observations (wind speed, prices) over many time periods, or if it represents observations of the attributes of groups of people (such as medical patients).

The challenge of representing uncertain events using well-designed samples is growing into a mature literature. We refer the reader to the bibliographic notes for some guidance as of the time that this book is being written.

4.3.4 Decomposition strategies*

Let $\overline{W} = \mathbb{E}W$ be a point estimate of our random variable W. From time to time, we encounter problems where the deterministic problem

$$\max_{x \in \mathcal{X}} F(x, \overline{W}),$$

is reasonably difficult to solve. For example, it might be a large integer program which might arise when scheduling airlines or planning when energy generators should turn on and off. In this case, $F(x, \overline{W})$ would be the contribution function and $\mathcal{X}$ would contain all the constraints, including integrality. Imagine that we can solve the deterministic problem, but it might not be that easy (integer programs might have 100,000 integer variables). If we want to capture the uncertainty of W using a sample of, say, 20 different values of W, then we create an integer program that is 20 times larger. Even modern solvers on today's computers have difficulty with this.

Now imagine that we decompose the problem so that there is a different solution for each possible value of W. Assume we have N sample outcomes $\omega^1, \omega^2, \ldots, \omega^N$ where $W^n = W(\omega^n)$ is the set of sample realizations of W corresponding to outcome ω^n. Let $x(\omega^n)$ be the optimal solution corresponding to this outcome.

We might start by rewriting our sampled stochastic optimization problem (4.16) as

$$\max_{x(\omega^1),\ldots,x(\omega^N)} \frac{1}{N} \sum_{n=1}^{N} F(x(\omega^n), W(\omega^n)). \tag{4.21}$$

We can solve this problem by creating N parallel problems and obtaining a different solution $x^*(\omega^n)$ for each ω. That is,

$$x^*(\omega^n) = \arg \max_{x(\omega^n) \in \mathcal{X}} F(x(\omega^n), W(\omega^n)).$$

This is a much smaller problem, but it also means choosing x assuming you know the outcome W. This would be like allowing an aircraft to arrive late to an airport because we already knew that the crew for the next leg was also going to be late.

The good news is that this is a starting point. What we really want is a solution where all the $x(\omega)$ are the same. We can introduce a constraint, often known as a *nonanticipativity constraint*, that looks like

$$x(\omega^n) - \bar{x} \quad = \quad 0, \;\; n = 1, \ldots, N. \tag{4.22}$$

If we introduce this constraint, we are just back to our original (and very large) problem. But what if we relax this constraint and add it to the objective function with a penalty λ. This produces the relaxed problem

$$\max_{x(\omega^1),\ldots,x(\omega^N)} \frac{1}{N} \sum_{n=1}^{N} \big(F(x(\omega^n), W(\omega^n)) + \lambda^n (x(\omega^n) - \bar{x})\big). \tag{4.23}$$

What is nice about this new objective function is that, just as with the problem in (4.21), it decomposes into N problems, which makes the overall problem solvable. Now the difficulty is that we have to coordinate the different subproblems by tuning the vector $\lambda^1, \ldots, \lambda^N$ until our nonanticipativity constraint (4.22) is satisfied.. We are not going to address this problem in detail, but this hints at a path for solving large scale problems using sampled means.

4.4 ADAPTIVE LEARNING ALGORITHMS

When we cannot calculate the expectation exactly, either through structure or resorting to a sampled model, we have to turn to adaptive learning algorithms. This transition fundamentally changes how we approach stochastic optimization problems, since any adaptive algorithm can be modeled as a sequential decision problem, otherwise known as a dynamic program.

We separate our discussion of adaptive learning algorithms between derivative-based algorithms, discussed in chapter 5, and derivative-free algorithms, presented in chapter 7. In between, chapter 6 discusses the problem of adaptively learning a signal, a problem that introduces the annoying but persistent problem of stepsizes that we first encountered in chapter 5, but which pervades the design of adaptive learning algorithms.

We begin by offering a general model of adaptive learning problems, which are basically a simpler example of the dynamic programs that we consider later in the book. As we illustrate in chapters 5 and 7, adaptive learning methods can be viewed as sequential decision problems (dynamic programs) where the state variable captures only what we know about the state of the search algorithm. This gives us an opportunity to introduce some of the core ideas of sequential decision problems, without all the richness and complexity that come with this problem class.

Below, we sketch the core elements of any sequential decision problem, and then outline the fundamental class of policies (or algorithms) that are used to solve them.

4.4.1 Modeling adaptive learning problems

Whether we are solving a derivative-based or derivative-free problem, any adaptive learning algorithm is going to have the structure of a sequential decision problem, which has five core components:

State S^n - This will capture the current point in the search, and other information required by the algorithm. The nature of the state variable depends heavily on how we are structuring our search process. The state variable may capture beliefs about the function (this is a major issue in derivative-free stochastic search), as well as the state of the algorithm itself. In chapter 9, we tackle the problem of modeling general dynamic programs which include states that are directly controllable (most often, these are physical problems).

Decision x^n - While this is sometimes x^n, the precise "decision" being made within an adaptive learning algorithm depends on the nature of the algorithm, as we see in chapter 5. Depending on the setting, decisions are made by a decision rule, an algorithm, or (the term we primarily use), a policy. If x is our decision, we designate $X^\pi(S)$ as the policy (or algorithm).

Exogenous information W^{n+1} - This is the new information that is sampled during the n^{th} iteration (but after making decision x^n), either from a Monte Carlo simulation or observations from an exogenous process (which could be a computer simulation, or the real world).

Transition function - The transition function includes the equations that govern the evolution from S^n to S^{n+1}. Our default notation used throughout this volume is to write

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

Objective function - This is how we evaluate how well the policy is performing. The notation depends on the setting. We may have a problem where we make a decision x^n at the end of iteration n, then observe information W^{n+1} in iteration $n+1$, from which we can evaluate our performance using $F(x^n, W^{n+1})$. This is going to be our default notation for learning problems.

When we make the transition to more complex problems with a physical state, we are going to encounter problems where the contribution (cost if minimizing) depends on the state S^n and decision x^n, which we would write as $C(S^n, x^n)$, but there are other variations. We return to the objective function below.

We are going to be able to model any sequential learning algorithm as a sequential decision process that can be modeled as the sequence

$$(S^0, x^0 = X^\pi(S^0), W^1, S^1, x^1 = X^\pi(S^1), W^2, \ldots).$$

Thus, all sequential learning algorithms, for *any* stochastic optimization problem, can ultimately be reduced to a sequential decision problem.

For now (which is to say, chapters 5 and 7), we are going to limit our attention to where decisions only affect what we learn about the function we are optimizing. In chapter 8, we are going to introduce the complex dimension of controllable physical states.

Mathematically, there is no difference in how we formulate a problem where the state consists only of what we know about a function, versus problems where the state captures the locations of people, equipment and inventory. However, pure learning problems are much simpler, and represent a good starting point for modeling and solving stochastic optimization problems using sequential (adaptive) methods. In addition, we will be using these methods throughout the remainder of the book. For example, policy search methods (chapters 12 and 13) both require that we solve stochastic search problems, which we may approach using either derivative-based or derivative-free methods.

4.4.2 Online vs. offline applications

The terms "online" and "offline" are terms that are widely used in both machine learning and stochastic optimization settings, but they take on different interpretations which can be quite important, and which have created considerable confusion in the literature. Below we explain the terms in the context of these two communities, and then describe how these terms are used in this volume.

Machine learning - Machine learning is an optimization problem that involves minimizing the error between a proposed model (typically parametric) and a dataset. We can represent the model by $f(x|\theta)$ where the model may be linear or nonlinear in θ (see chapter 3). The most traditional representation is to assume that we have a set of input variables $x^1, \ldots, x^n$ with a corresponding set of observations $y^1, \ldots, y^n$, to which we are going to fit our model by solving

$$\min_{\theta} \sum_{i=1}^{n} (y^i - f(x^i|\theta))^2, \tag{4.24}$$

where we might represent the optimal solution to (4.24) by θ^*. This problem is solved as a batch optimization problem using any of a set of deterministic optimization algorithms. This process is classically known as offline learning in the machine learning. Once we find θ^*, we would presumably use our model $f(x|\theta^*)$ to make an estimate of something, such as a forecast of the future, or a product recommendation.

In online learning, we assume that data is arriving sequentially over time. In this case, we are going to assume that we see x^n and then observe y^{n+1}, where the use of $n+1$ is our way of showing that y^{n+1} is observed after seeing $x^0, \ldots, x^n$. Let D^n be our dataset at time n where

$$D^n = \{x^0, y^1, x^1, y^2, \ldots, x^{n-1}, y^n\}.$$

We need to estimate a new value of θ, which we call θ^n, for each new piece of information which includes (x^{n-1}, y^n). We would call any method we use to compute θ^n a *learning policy*, but one obvious example would be

$$\theta^n = \arg\min_{\theta} \sum_{i=0}^{n-1} (y^{i+1} - f(x^i|\theta))^2. \tag{4.25}$$

More generally, we could write our learning policy as $\theta^n = \Theta^\pi(D^n)$. As our dataset evolves $D^1, D^2, \ldots, D^n, D^{n+1}, \ldots$, we update our estimate θ^n sequentially.

In the eyes of the machine learning community, the difference between the offline problem in equation (4.24) and the online learning problem in (4.25) is that the first is a single, batch optimization problem, while the second is implemented sequentially.

Optimization - Imagine that we are trying to design a new material to maximize the conversion of solar energy to electricity. We will go through a series of experiments testing different materials, as well as continuous parameters such as the thickness of a layer of a material. We wish to sequence our experiments to try to create a surface that maximizes energy conversion within our experimental budget. What we care about is how well we do in the end; trying a design that does not work is not a problem as long as the final design works well.

Now consider the problem of actively tilting solar panels to maximize the energy production over the course of the day, where we have to handle not just the changing angle of the sun during the day (and over seasons), but also with changes in cloud cover. Again, we may have to experiment with different angles, but now we need to maximize the total energy created while we are trying to learn the best angle.

We would treat the first problem as an offline problem since we are learning in the lab, while the second is an online problem since we are optimizing in the field. When we are in the lab, we do not mind failed experiments as long as we get the best result in the end, which means we would maximize final reward. By contrast, when we are learning in the field we want to optimize the cumulative reward. Note that both problems are fully sequential, which means the machine learning community would view both as forms of online learning.

We show how to write out the objective functions for our offline and online settings next.

4.4.3 Objective functions for learning

In contrast with the exact methods for solving stochastic optimization problems, there are different ways to formulate the objective function for adaptive learning problems. For learning problems, we are going to let $F(x, W)$ be the function that captures our performance objective when we make decision x and then observe random information W. In an iterative setting, we will write $F(x^n, W^{n+1})$; in a temporal setting, we will write $F(x_t, W_{t+1})$. Our choice $x^n = X^\pi(S^n)$ will be made by a policy that depends on the state, but otherwise the contribution $F(x, W)$ depends only on the action and random information.

The function $\mathbb{E}F(x, W)$ captures the performance of our implementation decision x. To make a good decision, we need to design an algorithm, or more precisely, a learning policy $X^\pi(S)$, that allows us to find the best x. There are different objective functions for capturing the performance of a learning policy:

Final reward - Let $x^{\pi,n} = X^\pi(S^n)$ be our solution at iteration n while following policy π. We may analyze the policy π in two ways:

Finite time analysis - Here, we want to solve

$$\max_\pi \mathbb{E}\{F(x^{\pi,N}, W)|S^0\} = \mathbb{E}_{S^0}\mathbb{E}_{W^1,\ldots,W^N|S^0}\mathbb{E}_{\widehat{W}|S^0}\mathbb{E}_W F(x^{\pi,N}, \widehat{W}) \quad (4.26)$$

where:

- S^0 might include a distribution of belief about unknown parameters (such as whether a patient is allergic to a drug),
- $W^1, \ldots, W^N$ are the observations we make while running our search policy π for N iterations (these are the training iterations),
- $\widehat{W}$ is the sampling done to test the performance of the final design $x^{\pi,N}$.

Asymptotic analysis - In this setting, we are trying to establish that

$$\lim_{N\rightarrow\infty} x^{\pi,N} \rightarrow x^*$$

where x^* solves $\max_x \mathbb{E}F(x,W)$. In both of these settings, we are only interested in the quality of the final solution, whether it is $x^{\pi,N}$ or x^*. We do not care about the solutions obtained along the way.

Cumulative reward - Cumulative reward objectives arise when we are interested not just in the performance after we have finished learning the best asymptotic design x^*, or the best design in a finite budget N, $x^{\pi,N}$, or finite time T, x^π_T. We divide these problems into two broad classes:

Deterministic policy - The most common setting is where we want to design a single policy that optimizes the cumulative reward over some horizon. We can further divide deterministic policies into two classes:

Stationary policy - This is the simplest setting, where we wish to find a single policy $X^\pi(S_t)$ to solve:

$$\max_\pi \mathbb{E}\sum_{t=0}^{T-1} F(X^\pi(S_t),W_{t+1}), \tag{4.27}$$

within a finite time horizon T. We may write this in either a discounted objective,

$$\max_\pi \mathbb{E}\sum_{t=0}^{T} \gamma^t C(S_t,X^\pi(S_t)), \tag{4.28}$$

or average reward,

$$\max_\pi \mathbb{E}\frac{1}{T}\sum_{t=0}^{T} C(S_t,X^\pi(S_t)). \tag{4.29}$$

Both (4.28) and (4.29) can be extended to infinite horizon, where we would replace (4.29) with

$$\max_\pi \lim_{T\rightarrow\infty} \mathbb{E}\frac{1}{T}\sum_{t=0}^{T} C(S_t,X^\pi(S_t)). \tag{4.30}$$

Time-dependent policy - There are many problems where we need a time-dependent policy $X^\pi_t(S_t)$, either because the behavior needs to vary by time of day, or because we need different behaviors based on how close the decisions are to the end of horizon. We denote the policy by time t as $X^\pi_t(S_t)$, but let π_t refer to the choices (type of function, parameters) we need to make for each time period. These problems would be formulated

$$\max_{\pi_0,\ldots,\pi_{T-1}} \mathbb{E}\sum_{t=0}^{T-1} F(X^\pi_t(S_t),W_{t+1}). \tag{4.31}$$

Although the policies are time dependent, they are in the class of static policies because they are designed before we start the process of making observations.

Adaptive policy - Now we allow our policies to learn over time, as would often happen in an online setting. Modeling this is a bit subtle, and it helps to use an example. Imagine that our policy is of the form

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

This would be an example of a stationary policy parameterized by $\theta = (\theta_0, \theta_1, \theta_2)$. Now imagine that θ is a function of time, so we would write our policy as

$$X_t^{\pi}(S_t|\theta_t) = \theta_{t0} + \theta_{t1} S_t + \theta_{t2} S_t^2,$$

where we have now written the *policy* $X_t^{\pi}(S_t)$ as being time-dependent, since the function depends on time (through the parameter vector θ_t).

Finally, imagine that we have an adaptive policy that updates θ_t after computing $x_t = X^{\pi}(S_t|\theta_t)$ and observing W_{t+1}. Just as we have to make a decision x_t, we have to "decide" on how to set θ_{t+1} given S_{t+1} (which depends on S_t, x_t and W_{t+1}). In this case, θ_t becomes a part of the state variable (along with any other statistics needed to compute θ_{t+1} given what we know at time t).

We refer to the policy for learning θ as our *learning policy*, which we designate $\Theta^{\pi^{lrn}}$, where we would write

$$\theta_t = \Theta^{\pi^{lrn}}(S_t).$$

We refer to $\Theta^{\pi^{lrn}}(S_t)$ as the *learning policy* (also known as the "behavior policy") while $X^{\pi^{imp}}(S_t|\theta_t)$ is the *implementation policy*, which is the policy that makes the decisions that are implemented (this is also known as the "target policy"). This problem is formulated as

$$\max_{\pi^{imp}} \max_{\pi^{lrn}} \mathbb{E} \sum_{t=0}^{T-1} F(X^{\pi^{imp}}(S_t|\theta_t), W_{t+1}).$$

For learning problems (problems where the *function* $F(x, W)$ does not depend on the state), we are going to use (4.26) (the final reward) or (4.27) (the cumulative reward for stationary policies) as our default notation for the objective function.

It is common, especially in the machine learning community, to focus on *regret* rather than the total reward, cost or contribution. Regret is simply a measure of how well you do relative to how well you could have done (but recognize that there are different ways of defining the best we could have done). For example, imagine that our learning policy has produced the approximation $\bar{F}^{\pi,N}(x)$ of the function $\mathbb{E}F(x, W)$ by following policy π after N samples, and let

$$x^{\pi,N} = \arg\max_x \bar{F}^{\pi,N}(x)$$

be the best solution based on the approximation. The regret $\mathcal{R}^{\pi,N}$ would be given by

$$\mathcal{R}^{\pi,N} = \max_x \mathbb{E}F(x, W) - \mathbb{E}F(x^{\pi,N}, W). \tag{4.32}$$

Of course, we cannot compute the regret in a practical application, but we can study the performance of algorithms in a setting where we assume we know the true function (that is, $\mathbb{E}F(x,W)$), and then compare policies to try to discover this true value. Regret is popular in theoretical research (for example, computing bounds on the performance of policies), but it can also be used in computer simulations comparing the performance of different policies.

4.4.4 Designing policies

Now that we have presented a framework for modeling our learning problems, we need to address the problem of designing policies (we will sometimes refer to these as algorithms), especially in chapter 7 when we deal with derivative-free optimization.

We originally introduced the different classes of policies in section 1.4. As a brief reminder, there are two fundamental strategies for designing policies, each of which break down into two subclasses, creating four classes of policies:

Policy search - These are functions that are tuned to work well over time without directly modeling the effect of a decision now on the future. Policies designed using policy search fall into two styles:

- **Policy function approximations (PFAs)** - PFAs are analytical functions that map directly from state to a decision.
- **Cost function approximations (CFAs)** - CFAs involve maximizing (or minimizing) a parameterized optimization problem that returns a decision.

Lookahead policies - These are policies that are designed by estimating, directly or indirectly, the impact of a decision now on the future. There are again two ways of creating these policies:

- **Value function approximations (VFAs)** - If we are in a state S^n, make a decision x^n, that leads (with the introduction of new information) to a new state S^{n+1}, assume we have a function $V^{n+1}(S^{n+1})$ that estimates (exactly or, more often, approximately) the value of being in state S^{n+1}. The value function $V^{n+1}(S^{n+1})$ captures the downstream impact of decision x^n, and can be used to help us make the best decision now.
- **Direct lookahead policies (DLAs)** - These are policies where we model the downstream trajectory of each decision, and the optimizing across decisions now as well as decisions in the future (which may have to incorporate uncertainty).

The importance of each of these four classes depends on the characteristics of the problem. We are going to see all four of these classes used in the setting of derivative-free optimization in chapter 7. By contrast, derivative-based search strategies reviewed in chapter 5 have historically been more limited, although this perspective potentially introduces new strategies that might be pursued. When we transition to problems with physical states starting in chapter 8, we are going to see that we will need to draw on all four classes. For this reason, we discuss these four classes in more depth in chapter 11.

4.5 CLOSING REMARKS

This chapter offers three fundamental perspectives of stochastic optimization problems. Section 4.2 is basically a reminder that any stochastic optimization problem can be solved

as a deterministic optimization problem if we are able to compute the expectation exactly. While this will not happen very often, we offer this section as a reminder to readers not to overlook this path.

Section 4.3 then introduces the powerful approach of using sampled models, where we overcome the complexity of computing an expectation by replacing the underlying uncertainty model with a small sampled set, which is much easier to model. This strategy should always be in your toolbox, even when it will not solve the entire problem.

When all else fails (which is most of the time), we are going to need to turn to adaptive learning strategies, which are increasingly being grouped under the umbrella known as *reinforcement learning*. These approaches have evolved into substantial areas of research, which we divide into derivative-based methods in chapter 5, and derivative-free methods in chapter 7. In chapter 5, we are going to see that we need a device called "stepsizes" (which we cover in chapter 6), which can be viewed as a type of decision, where different stepsize rules are actually types of policies.

4.6 BIBLIOGRAPHIC NOTES

- Section 4.2.2 - The newsvendor problem where the distribution of W is known can be found in any standard textbook on inventory theory (see, e.g. Porteus (2002)), and is also a standard canonical problem in many books on stochastic optimization (see, for example, Shapiro et al. (2014)).
- Section 4.2.3 - See the bibliographic notes for section 2.1.12 for references on chance-constrained programming.
- Section 4.2.4 - See the bibliographic notes for section 2.1.4 for references on optimal control.
- Section 4.2.5 - We address Markov decision processes in detail in chapter 14 and the references cited there.
- Section 4.3.1 - See the references for section 2.1.8 for some references on stochastic programming.
- Section 4.3.2 - The convergence rate results given in equations (4.19) and (4.20) are presented in Shapiro et al. (2014), based on work in Shapiro & Wardi (1996) and Shapiro & Homem-de Mello (2000). An excellent presentation of sampled methods and the convergence rates is given in Kim, Pasupathy and Henderson's chapter in Fu (2014)[Chapter 8], as well as Ghadimi and Lan's chapter on finite time convergence properties [Chapter 7].
- Section 4.3.4 - The decomposition of stochastic programs was exploited in Rockafellar & Wets (1991) using a technique called "progressive hedging." Mulvey et al. (1995) implemented the method and performed numerical testing.
- Section 4.3.3 - The use of scenarios to approximate the future can make what are already large problems much larger, so there has been considerable attention given to the process of sampling efficient scenarios; see Dupacova et al. (2003) and Heitsch & Romisch (2009) for early, but important, contributions to this field.
- Section 4.4.1 - Every adaptive problem, whether it be a sequential decision problem or a stochastic algorithm, can be modeled using the five elements listed here. This

structure was first presented in this style in Powell (2011). This framework follows the style of deterministic math programs, which consist of three core elements: decision variables, constraints, and the objective function. Our framework builds off the modeling framework used in stochastic, optimal control (see, for example, Kirk (2012), Stengel (1986), Sontag (1998), Sethi (2019), and Lewis & Vrabie (2012)). Powell (2021) contrasts the modeling framework used in this volume to the modeling style of Markov decision processes (which has been adopted in reinforcement learning) to that used in optimal control.

- Section 4.4.3 - We first used the finite time formulation of the stochastic search problem given in equation (4.3) in Powell (2019); we have not seen this formulation used elsewhere, since the asymptotic formulation in (4.1) is so standard in the stochastic optimization literature.

- Section 4.4.4 - Powell (2011)[Chapter 6] is the first reference to discuss different classes of policies, but overlooked cost function approximations. Powell (2014) was the first time the four classes of policies (as listed here) were given, without recognizing that they belonged in two classes. Powell (2016) presented the four classes of policies, divided between the two strategies: policy search and lookahead policies. Powell (2019) summarized these again, introducing additional modeling insights such as final and cumulative reward (equation (4.3) is written as final reward, but it can also be stated in a cumulative reward format, as we will do in chapter 7). This book is the first to present these ideas formally.

EXERCISES

Review questions

4.1 In your own words, explain why $\min_x \mathbb{E}_W F(x, W)$ is properly viewed as a deterministic optimization problem if we can compute the expectation W.

4.2 How would we compute $\mathbb{E}_W F(x, W)$ using a sampled approximation? Does this meet the conditions for a deterministic optimization problem? Explain (briefly!).

4.3 Assume we take a sample $\{w^1, \ldots, w^N\}$ and then solve the sampled representation

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

to obtain an optimal solution x^N. Let x^* solve $\max_x \mathbb{E}F(x, W)$ (if we could compute this). What is the rate that $F(x^N)$ approaches $F(x^*)$? when $F(x)$ is concave?

4.4 What is the difference between offline and online learning in the machine learning community?

4.5 Write out the objective functions for final reward and cumulative reward? Be sure to use the expanded form of the expectation, which means you need to indicate what random variables each expectation is being taken over.

Modeling questions

4.6 Our basic newsvendor problem

$$F(x, W) = p \max\{0, x - W\} - cx,$$

can be written as different forms of optimization problems:

a) Write out the asymptotic form of the optimization problem to maximize the final reward.

b) Write out the final reward version of the newsvendor problem assuming we can only perform N observations of the newsvendor problem.

c) Assume that we have to perform our learning in the field, which means we need to maximize the sum of the rewards over N observations. Write out the objective function for this problem.

4.7 We illustrated $F(x, W)$ above using our basic newsvendor problem

$$F(x, W) = p \max\{0, x - W\} - cx,$$

but this is general notation that can be used to represent an entire range of sequential decision problems. Imagine that we have an asset selling problem, where we are determining when to sell an asset. Let W be a sequence of prices $p_1, p_2, \ldots, p_t, \ldots, p_T$. Assume we are going to sell our stock when $p_t \geq x$, which means that "x" defines a policy. Write out what $F(x, W)$ means for this problem, and formulate the objective function to optimize over policies.

Problem solving questions

4.8 In a flexible spending account (FSA), a family is allowed to allocate x pretax dollars to an escrow account maintained by the employer. These funds can be used for medical expenses in the following year. Funds remaining in the account at the end of the following year are given back to the employer. Assume that you are in a 35 percent tax bracket (sounds nice, and the arithmetic is a bit easier).

Let W be the random variable representing total medical expenses in the upcoming year, and let $P^W(S) = Prob[W \leq w]$ be the cumulative distribution function of the random variable W.

a) Write out the objective function $F(x)$ that we would want to solve to find x to minimize the total cost (in pretax dollars) of covering your medical expenses next year.

b) If x^* is the optimal solution and $\nabla_x F(x)$ is the gradient of your objective function if you allocate x to the FSA, use the property that $\nabla_x F(x) = 0$ to derive the critical ratio that gives the relationship between $x*$ and the cumulative distribution function $P^W(w)$.

c) Given your 35 percent tax bracket, what percentage of the time should you have funds left over at the end of the year?

4.9 Consider the problem faced by a mutual fund manager who has to decide how much to keep in liquid assets versus investing to receive market returns. Assume he has R_t dollars to invest at the end of day t, and needs to determine the quantity x_t to put in cash at the end of day t to meet the demand $\hat{D}_{t+1}$ for cash in day $t+1$. The remainder, $R_t - x_t$, is to be invested and will receive a market return of $\hat{\rho}_{t+1}$ (for example, we might have $\hat{\rho}_{t+1} = 1.0002$, implying a dollar invested is worth 1.0002 tomorrow). Assume there is nothing earned for the amount held in cash.

If $\hat{D}_t > x_{t-1}$, the fund manager has to redeem stocks. Not only is there a transaction cost of 0.20 percent (redeeming \$1000 costs \$2.00), the manager also has to pay capital gains. His fund pays taxes on the average gain of the total assets he is holding (rather than the gain on the money that was just invested). At the moment, selling assets generates a tax commitment of 10 percent which is deducted and held in escrow. Thus, selling \$1000 produces net proceeds of 0.9(1000 - 2). As a result, if he needs to cover a cash request of \$10,000, he will need to sell enough assets to cover both the transaction costs (which are tax deductible) and the taxes, leaving \$10,000 net proceeds to cover the cash request.

a) Formulate the problem of determining the amount of money to hold in cash as a stochastic optimization problem. Formulate the objective function $F(x)$ giving the expected return when holding x dollars in cash.

b) Give an expression for the stochastic gradient $\nabla_x F(x)$.

c) Find the optimal fraction of the time that you have to liquidate assets to cover cash redemption. For example, if you manage the fund for 100 days, how many days would you expect to liquidate assets to cover cash redemptions?

4.10 Independent system operators (ISOs) are companies that manage our power grid by matching generators (which create the energy) with customers. Electricity can be generated via steam, which takes time, or gas turbines which are fast but expensive. Steam generation has to be committed in the day-ahead market, while gas turbines can be brought on line with very little advance notification.

Let x_t be the amount of steam generation capacity (measured in megawatt-hours) that is requested on day t to be available on day $t+1$. Let $p^{steam}_{t,t+1}$ be the price of steam on day $t+1$ that is bid on day t (which is known on day t). Let D_{t+1} be the demand for electricity (also measured in megawatt-hours) on day $t+1$, which depends on temperature and other factors that cannot be perfectly forecasted. However, we do know the cumulative distribution function of D_{t+1} , given by $F^D(d) = Prob[D_{t+1} < d]$. If the demand exceeds the energy available from steam (planned on day t), then the balance has to be generated from gas turbines. These are bid at the last minute, and therefore we have to pay a random price p^{GT}_{t+1}. At the same time, we are not able to store energy; there is no inventory held over if $D_{t+1} < x_t$. Assume that the demand D_{t+1} and the price of electricity from gas turbines p^{GT}_{t+1} are independent.

a) Formulate the objective function $F(x)$ to determine x_t as an optimization problem.

b) Compute the stochastic gradient of your objective function $F(x)$ with respect to x_t. Identify which variables are known at time t, and which only become known at time $t+1$.

c) Find an expression that characterizes the optimal value of x_t in terms of the cumulative probability distribution $F^D(d)$ of the demand D_T.

4.11 We are going to illustrate the difference between

$$\max_x \mathbb{E}F(x, W) \tag{4.33}$$

and

$$\max_x F(x, \mathbb{E}W) \tag{4.34}$$

using a sampled belief model. Assume we are trying to price a product where the demand function is given by

$$D(p|\theta) = \theta^0 \frac{e^{U(p|\theta)}}{1 + e^{U(p|\theta)}}, \tag{4.35}$$

where

$$U(p|\theta) = \theta_1 + \theta_2 p.$$

Our goal is to find the price that maximizes total revenue given by

$$R(p|\theta) = pD(p|\theta). \tag{4.36}$$

Here, our random variable W is the vector of coefficients $\theta = (\theta_0, \theta_1, \theta_2)$ which can take one of four possible values of θ given by the set $\Theta = \{\theta^1, \theta^2, \theta^3, \theta^4\}$.

θ	$P(\theta)$	θ_0	θ_1	θ_2
θ^1	0.20	50	4	-0.2
θ^2	0.35	65	4	-0.3
θ^3	0.30	75	4	-0.4
θ^4	0.15	35	7	-0.25

Table 4.1 Data for exercise 4.11.

a) Find the price $p(\theta)$ that maximizes

$$\max_p R(p|\theta), \tag{4.37}$$

for each of the four values of θ. You may do this analytically, or to the nearest integer (the relevant range of prices is between 0 and 40). Either way, it is a good idea to plot the curves (they are carefully chosen). Let $p^*(\theta)$ be the optimal price for each value of θ and compute

$$R^1 = \mathbb{E}_\theta \max_{p(\theta)} R(p^*(\theta)|\theta). \tag{4.38}$$

b) Find the price p that maximizes

$$R^2 = \max_p \mathbb{E}_\theta R(p|\theta), \tag{4.39}$$

where $R(p|\theta)$ is given by equation (4.36).

c) Now find the price p that maximizes

$$R^3 = \max_p R(p|\mathbb{E}\theta).$$

d) Compare the optimal prices and the optimal objective functions R^1, R^2 and R^3 produced by solving (4.37), (4.39) and (4.40). Use the relationships among the revenue functions to explain as much as possible about the relevant revenues and prices.

Theory questions

4.12 Recall our newsvendor problem

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

where $F(x, W) = p \min(x, W) - cx$. Assume that W is given by a known distribution $f^W(w)$ with cumulative distribution

$$F^W(w) = \mathbb{P}[W \leq w].$$

You are going to show that the optimal solution x^* satisfies

$$F^W(x^*) = \frac{p - c}{p}. \tag{4.40}$$

Do this by first finding the stochastic gradient $\nabla_x F(x, W)$ which will give you a gradient that depends on whether $W < x$ or $W > x$. Now take the expected value of this gradient and set it equal to zero, and use this to show (4.40).

4.13 The newsvendor problem is given by

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

where

$$F(x, W) = p \min\{x, W\} - cx,$$

where we assume that our sales price p is strictly greater than the purchase cost c. An important property of the newsvendor problem is that $F(x)$ is concave in x. This means, for example, that

$$\lambda F(x_1) + (1 - \lambda)F(x_2) \leq F(\lambda x_1 + (1 - \lambda)x_2), \tag{4.41}$$

for $0 \leq \lambda \leq 1$, and where $x_1 \leq x_2$. This property is illustrated in figure 4.1.

a) To begin, fix the random variable W, and show that $F(x, W)$ is concave (this should be apparent just by plotting the graph of $F(x, W)$ for a fixed W).

b) Now assume that W can only take on a fixed set of values $w^1, \ldots, w^N$, where each occurs with probability $p^n = Prob[W = w^n]$. Let $F(x) = \sum_{n=1}^N p^n F(x, w^n)$. Substitute this into equation (4.41) to show that the weighted sum of concave functions is concave.

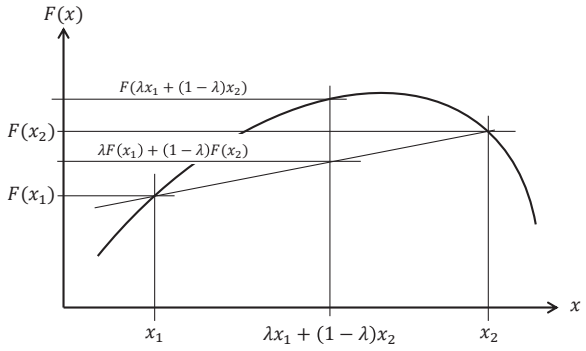

Figure 4.1 Concave function, showing that $\lambda F(x_1) + (1-\lambda)F(x_2) \leq F(\lambda x_1 + (1-\lambda)x_2)$.

c) Finally argue that (b) implies that the newsvendor problem is concave in x.

Diary problem

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

4.14 For your diary problem:

a) Do you think you could reduce your problem to a deterministic problem as described in section 4.2? If not, could you approximate it using the sampled methods described in section 4.3? For each of these two approaches, explain why or why not. Assuming neither of these work, can you sketch how an adaptive search algorithm might work?

b) Identify whether you would formulate your decision problem using a final reward or cumulative reward objective?

Bibliography

Dupacova, J., Growe-Kuska, N. & Romisch, W. (2003), 'Scenario reduction in stochastic programming: An approach using probability metrics', *Math. Program.,* **Ser. A 95**, 493–511.

Fu, M. C. (2014), *Handbook of Simulation Optimization*, Springer, New York.

Heitsch, H. & Romisch, W. (2009), 'Scenario tree modeling for multistage stochastic programs', *Mathematical Programming* **118**, 371–406.

Kirk, D. E. (2012), *Optimal Control Theory: An introduction*, Dover, New York.

Lewis, F. L. & Vrabie, D. (2012), *Design Optimal Adaptive Controllers*, 3 edn, John Wiley & Sons, Hoboken, NJ.

Mulvey, J. M., Vanderbei, R. J. & Zenios, S. A. (1995), 'Robust Optimization of Large-Scale Systems', *Operations Research* **43**(2), 264–281.

Porteus, E. L. (2002), *Foundations of Stochastic Inventory Theory*, Stanford University Press, Stanford.

Powell, W. B. (2011), *Approximate Dynamic Programming: Solving the Curses of Dimensionality*, 2 edn, John Wiley & Sons.

Powell, W. B. (2014), 'Clearing the Jungle of Stochastic Optimization', *Informs TutORials in Operations Research 2014*.

Powell, W. B. (2016), A Unified Framework for Optimization under Uncertainty, *in* 'Informs TutORials in Operations Research', pp. 45–83.

Powell, W. B. (2019), 'A unified framework for stochastic optimization', *European Journal of Operational Research* **275**(3), 795–821.

Powell, W. B. (2021), 'From reinforcement learning to optimal control: A unified framework for sequential decisions', *Handbook on Reinforcement Learning and Optimal Control, Studies in Systems, Decision and Control* pp. 29–74.

Rockafellar, R. T. & Wets, R. J.-B. (1991), 'Scenarios and policy aggregation in optimization under uncertainty', *Mathematics of Operations Research* **16**(1), 119–147.

Sethi, S. P. (2019), *Optimal Control Theory: Applications to Management Science and Economics*, 3 edn, Springer-Verlag, Boston.

Shapiro, A. & Homem-de Mello, T. (2000), 'On the rate of convergence of optimal solutions of Monte Carlo approximations of stochastic programs', *SIAM Journal on Optimization* **11**, 70–86.

Shapiro, A. & Wardi, Y. (1996), 'Convergence Analysis of Stochastic Algorithms', *Mathematics of Operations Research* **21**, 615–628.

Shapiro, A., Dentcheva, D. & Ruszczyński, A. (2014), *Lectures on Stochastic Programming: Modeling and theory*, 2 edn, SIAM, Philadelphia.

Sontag, E. (1998), 'Mathematical Control Theory, 2nd ed.', *Springer* pp. 1–544.

Stengel, R. F. (1986), *Stochastic optimal control: theory and application*, John Wiley & Sons, Hoboken, NJ.