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

PART II - STOCHASTIC SEARCH

Stochastic search covers a broad class of problems that are typically grouped under names such as stochastic approximation methods (derivative-based stochastic search), ranking and selection (derivative-free stochastic search), simulation optimization, and multiarmed bandit problems. We include in this part problems that are often solved using iterative algorithms, where the only information carried from one iteration to the next is what we have learned about the function. This is the defining characteristic of a learning problem.

Chapter 5 begins with derivative-based algorithms, where we describe the difference between asymptotic and finite-time analysis. This chapter identifies the importance of stepsizes, which are actually "decisions" in derivative-based methods. Chapter 6 provides an in-depth discussion of stepsize policies.

We then transition to derivative-free problems in chapter 7, where there is a much richer tradition of designing policies compared to derivative-based methods. This will be the first time we fully explore our canonical framework and the four classes of policies. Derivative-free stochastic search is a sequential decision problem characterized by a pure belief state which captures our approximation of the underlying problem. This allows us to build a bridge to the multiarmed bandit community. We also introduce the idea of *active learning*, where we make decisions specifically to improve our knowledge of the function we are optimizing.

By the end of Part II, we will have laid the foundation for the much richer class of sequential decision problems that involve controllable physical states that link decisions and dynamics from one time period to the next. However, we will use the tools of these three chapters throughout the rest of the book, especially in the context of tuning parameters for policies.

CHAPTER 5

DERIVATIVE-BASED STOCHASTIC SEARCH

We begin our discussion of adaptive learning methods in stochastic optimization by addressing problems where we have access to derivatives (or gradients, if x is a vector) of our function $F(x, W)$. It is common to start with the asymptotic form of our basic stochastic optimization problem

$$\max_{x \in \mathcal{X}} \mathbb{E}\{F(x, W)|S^0\}, \tag{5.1}$$

but soon we are going to shift attention to finding the best algorithm (or policy) for finding the best solution within a finite budget. We are going to show that with any adaptive learning algorithm, we can define a state S^n that captures what we know after n iterations. We can represent any algorithm as a "policy" $X^\pi(S^n)$ which tells us the next point $x^n = X^\pi(S^n)$ given what we know, S^n, after n iterations. Eventually we complete our budget of N iterations, and produce a solution that we call $x^{\pi,N}$ to indicate that the solution was found with policy (algorithm) π after N iterations.

After we choose x^n, we observe a random variable W^{n+1} that is not known when we chose x^n. We then evaluate the performance through a function $F(x^n, W^{n+1})$ which can serve as a placeholder for a number of settings, including the results of a computer simulation, how a product works in the market, the response of a patient to medication, or the strength of a material produced in a lab. The initial state S^0 might contain fixed parameters (say the boiling point of a material), the attributes of a patient, the starting point of an algorithm, and beliefs about any uncertain parameters.

Reinforcement Learning and Stochastic Optimization. By Warren B. Powell

When we focus on this finite-budget setting, the problem in (5.1) becomes

$$\max_{\pi} \mathbb{E}\{F(x^{\pi,N}, W)|S^0\}, \tag{5.2}$$

but this way of writing the problem hides what is actually happening. Starting with what we know in S^0, we are going to apply our policy $X^\pi(S^n)$ while we generate the sequence

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

where the observations $W^1, \ldots, W^N$ might be called *training data* to produce the solution $x^{\pi,N}$. Once we have $x^{\pi,N}$, we evaluate it using a new random variable that we denote by $\widehat{W}$ which is what we use for testing. We then use $\widehat{W}$ to evaluate the performance of $x^{\pi,N}$ which is computed using

$$\bar{F}^{\pi,N} = \mathbb{E}_{\widehat{W}} F(x^{\pi,N}, \widehat{W}). \tag{5.3}$$

We are almost there. The problem with $\bar{F}^{\pi,N}$ is that it is a random variable that depends on the specific sequence $W^1, \ldots, W^N$, as well as any distributional information in S^0 (we return to this issue later). We have potentially three sources of uncertainty:

The initial state S^0 - The initial state S^0 might include a probability distribution describing our belief (say) of the mean of a random variable.

The training sequence $W^1, \ldots, W^N$ - These are our observations while we are computing $x^{\pi,N}$.

The testing process - Finally, we are going to repeatedly sample from W, using a random variable we call $\widehat{W}$ to make the distinction with the random variable W that we use for training $x^{\pi,N}$.

The value F^π of our policy (algorithm) $X^\pi(S)$ can now be written as (using our expanded form of the expectation)

$$F^\pi = \mathbb{E}_{S^0}\mathbb{E}_{W^1,\ldots,W^N|S^0}\mathbb{E}_{\widehat{W}|S^0}\{F(x^{\pi,N}, \widehat{W})|S^0\}. \tag{5.4}$$

These expectations can be a little frightening. In practice we are going to simulate them, but we defer this to later in the chapter.

The objective in (5.2) would be the natural finite-budget version of (5.1) (which we also call the *final reward* objective), but we should keep an open mind and recognize that we may also be interested in the *cumulative reward* formulation given by

$$\max_{\pi} \mathbb{E}_{S^0}\mathbb{E}_{W^1,\ldots,W^N|S^0}\left\{\sum_{n=0}^{N-1} F(x^n, W^{n+1})|S^0\right\}. \tag{5.5}$$

where $x^n = X^\pi(S^n)$ is our search policy (typically known as an "algorithm"). Note that when we maximize cumulative reward, we add up our performance as we go, so we do not have that final training step with $\widehat{W}$ that we did above with our final reward objective.

The transition from searching for a solution x to finding a function π is one of the central differences between deterministic and stochastic optimization problems. We are moving from looking for the best *solution* x to finding the best *algorithm* (or policy) π.

In this chapter, we assume that we can compute the gradient $\nabla F(x, W)$ once the random information W becomes known. This is most easily illustrated using the newsvendor

problem. Let x be the number of newspapers placed in a bin, with unit cost c. Let W be the random demand for newspapers (which we learn after choosing x), which are sold at price p. We wish to find x that solves

$$\max_x F(x) = \mathbb{E}F(x,W) = \mathbb{E}\big(p\min\{x,W\} - cx\big). \tag{5.6}$$

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

$$\nabla_x F(x,W) = \begin{cases} p-c & \text{if } x \leq W, \\ -c & \text{if } x > W. \end{cases} \tag{5.7}$$

The gradient $\nabla_x F(x,W)$ is known as a stochastic gradient because it depends on the random demand W, which is to say that we calculate it after we have observed W.

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

We start by summarizing a variety of applications.

5.1 SOME SAMPLE APPLICATIONS

Derivative-based problems exploit our ability to use the derivative after the random information has been observed (but remember that our decision x must be made before we have observed this information). These derivatives, known as stochastic gradients, require that we understand the underlying dynamics of the problem. When this is available, we have access to some powerful algorithmic strategies that have been developed since these ideas where first invented in 1951 by Robbins and Monro.

Some examples of problems where derivatives can be computed directly are:

- Cost minimizing newsvendor problem - A different way of expressing the newsvendor problem is one of minimizing overage and underage costs. Using the same notation as above, our objective function would be written

$$\min_x \mathbb{E}F(x,W) = \mathbb{E}[c^o \max\{0, x-W\} + c^u \max\{0, W-x\}]. \tag{5.8}$$

  We can compute the derivative of $F(x,\hat{D})$ with respect to x after W becomes known using

$$\nabla_x F(x,W) = \begin{cases} c^0 & \text{if } x > W, \\ -c^u & \text{if } x \leq W. \end{cases}$$

- Nested newsvendor - This hints at a multidimensional problem which would be hard to solve even if we knew the demand distribution. Here there is a single random demand D that we can satisfy with products $1,\ldots,K$ where we use the supply of products $1,\ldots,k-1$ before using product k. The profit maximizing version is given by

$$\max_{x_1,\ldots,x_K} = \sum_{k=1}^{K} p_k \mathbb{E}\min\left\{x_k, \left(D - \sum_{j=1}^{k-1} x_j\right)^+\right\} - \sum_{k=1}^{K} c_k x_k. \tag{5.9}$$

Although more complicated than the scalar newsvendor, it is still fairly straightforward to find the gradient with respect to the vector x once the demand becomes known.

- Statistical learning - Let $f(x|\theta)$ be a statistical model which might be of the form

$$f(x|\theta) = \theta_0 + \theta_1\phi_1(x) + \theta_2\phi_2(x) + \ldots.$$

Imagine we have a dataset of input variables $x^1, \ldots, x^N$ and corresponding response variables $y^1, \ldots, y^N$. We would like to find θ to solve

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

- Finding the best inventory policy - Let R_t be the inventory at time t. Assume we place an order x_t according to the rule

$$X^\pi(R_t|\theta) = \begin{cases} \theta^{max} - R_t & \text{If } R_t < \theta^{min} \\ 0 & \text{Otherwise.} \end{cases}$$

Our inventory evolves according to

$$R_{t+1} = \max\{0, R_t + x_t - D_{t+1}\}.$$

Assume that we earn a contribution $C(R_t, x_t, D_{t+1})$

$$C(R_t, x_t, D_{t+1}) = p\min\{R_t + x_t, D_{t+1}\} - cx_t.$$

We then want to choose θ to maximize

$$\max_\theta \mathbb{E}\sum_{t=0}^{T} C(R_t, X^\pi(R_t|\theta), D_{t+1}).$$

If we let $F(x, W) = \sum_{t=0}^{T-1} C(R_t, X^\pi(R_t|\theta), D_{t+1})$ where $x = (\theta^{min}, \theta^{max})$ and $W = D_1, D_2, \ldots, D_T$, then we have the same problem as our newsvendor problem in equation (5.6). In this setting, we simulate our policy, and then look back and determine how the results would have changed if θ is perturbed for the same sample path. It is sometimes possible to compute the derivative analytically, but if not, we can also do a numerical derivative (but using the same sequence of demands).

- Maximizing e-commerce revenue - Assume that demand for a product is given by

$$D(p|\theta) = \theta_0 - \theta_1 p + \theta_2 p^2.$$

Now, find the price p to maximize the revenue $R(p) = pD(p|\theta)$ where θ is unknown.

- Optimizing engineering design - An engineering team has to tune the timing of a combustion engine to maximize fuel efficiency while minimizing emissions. Assume the design parameters x include the pressure used to inject fuel, the timing of the beginning of the injection, and the length of the injection. From this the engineers observe the gas consumption $G(x)$ for a particular engine speed, and the emissions $E(x)$, which are combined into a utility function $U(x) = U(E(x), G(x))$ which

combines emissions and mileage into a single metric. $U(x)$ is unknown, so the goal is to find an estimate $\bar{U}(x)$ that approximates $U(x)$, and then maximize it.

- Derivatives of simulations - In the previous section we illustrated a stochastic gradient algorithm in the context of a simple newsvendor problem. Now imagine that we have a multiperiod simulation, such as we might encounter when simulating flows of jobs around a manufacturing center. Perhaps we use a simple rule to govern how jobs are assigned to machines once they have finished a particular step (such as being drilled or painted). However, these rules have to reflect physical constraints such as the size of buffers for holding jobs before a machine can start working on them. If the buffer for a downstream machine is full, the rule might specify that a job be routed to a different machine or to a special holding queue.

  This is an example of a policy that is governed by static variables such as the size of the buffer. We would let x be the vector of buffer sizes. It would be helpful, then, if we could do more than simply run a simulation for a fixed vector x. What if we could compute the derivative with respect to each element of x, so that after running a simulation, we obtain all the derivatives?

  Computing these derivatives from simulations is the focus of an entire branch of the simulation community. A class of algorithms called *infinitesimal perturbation analysis* was developed specifically for this purpose. It is beyond the scope of our presentation to describe these methods in any detail, but it is important for readers to be aware that the field exists.

5.2 MODELING UNCERTAINTY

Before we progress too far, we need to pause and say a few words about how we are modeling uncertainty, and the meaning of what is perhaps the most dangerous piece of notation in stochastic optimization, the expectation operator $\mathbb{E}$.

We are going to talk about uncertainty from three perspectives. The first is the random variable W that arises when we evaluate a solution, which we refer to as *training uncertainty*. The second is the initial state S^0, where we express *model uncertainty*, typically in the form of uncertainty about parameters (but sometimes in the structure of the model itself). The third addresses testing uncertainty. In final-reward problems, we use the random variable $\widehat{W}$ for testing. In cumulative-reward settings, we test as we proceed.

5.2.1 Training uncertainty $W^1, \ldots, W^N$

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

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

Table 5.1 illustrates six sample paths for the sequence $W^1, \ldots, W^{10}$. We often let ω to represent an outcome of a random variable, or an entire sample path (as we would here). We might let Ω be the set of all the sample paths, which for this problem we would write as

ω	W^1	W^2	W^3	W^4	W^5	W^6	W^7	W^8	W^9	W^{10}
1	0	1	6	3	6	1	6	0	2	4
2	3	2	2	1	7	5	4	6	5	4
3	5	2	3	2	3	4	2	7	7	5
4	6	3	7	3	2	3	4	7	3	4
5	3	1	4	5	2	4	3	4	3	1
6	3	4	4	3	3	3	2	2	6	1

Table 5.1 Illustration of six sample paths for the random variable W.

$$\Omega = (\omega_1, \omega_2, \omega_3, \omega_4, \omega_5, \omega_6).$$

We could then let $W_t(\omega)$ be the outcome of the random variable W_t at time t for sample path ω. Thus, $W_5(\omega_2) = 7$. If we are following sample path ω using policy π, we obtain the final design $x^{\pi,N}(\omega)$. By running policy π for each outcome $\omega \in \Omega$, we would generate a population of designs $x^{\pi,N}$ which provide a nice way to represent $x^{\pi,N}$ as a random variable.

5.2.2 Model uncertainty S^0

We illustrate model uncertainty using our newsvendor problem, where we make a decision x, then observe a random demand $W = \hat{D}$, after which we calculate our profit using equation (5.6). Imagine that our demand follows a Poisson distribution given by

$$\mathbb{P}[W = w] = \frac{\mu^w e^{-\mu}}{w!},$$

where $w = 0, 1, 2, \ldots$. In this setting, our expectation would be over the possible outcomes of W, so we could write the optimization problem in equation (5.6) as

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

This does not look too hard, but what happens if we do not know μ? This parameter would be carried by our initial state S^0. If we are uncertain about μ, we may feel that we can describe it using an exponential distribution given by

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

where the parameter λ is known as a hyperparameter, which is to say it is a parameter that determines a distribution that describes the uncertainty of a problem parameter. The assumption is that even if we do not know λ precisely, it still does a good job of describing the uncertainty in the mean demand μ. In this case, S^0 would include both λ and the assumption that μ is described by an exponential distribution.

We would now write our expectation of $F(x, W)$ as

$$\begin{aligned} F(x) &= \mathbb{E}\{F(x, W)|S^0\}, \\ &= \mathbb{E}_{S^0}\mathbb{E}_{W|S^0}\{F(x, W)|S^0\}. \end{aligned}$$

For our example, this would be translated as

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

The notation $\mathbb{E}_{W|\mu}$ means the conditional expectation of W given μ. Using our distributions where the random demand W follows a Poisson distribution with mean μ which is itself random with an exponential distribution with mean λ, we would write the expectation as

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

In practice, we are rarely using explicit probability distributions. One reason is that we may not know the distribution, but we may have an exogenous source for generating random outcomes. The other is that we may have a distribution, but it might be multidimensional and impossible to compute.

5.2.3 Testing uncertainty

When we finally obtain our solution $x^{\pi,N}$, we then have to evaluate the quality of the solution. For the moment, let's fix $x^{\pi,N}$. We let $\widehat{W}$ denote the random observations we use when testing the performance of our final solution $x^{\pi,N}$. We use $\widehat{W}$ to represent the random observations while testing to avoid confusion with the random observations W we use while training.

We write the value of the solution $x^{\pi,N}$ using

$$F(x^{\pi,N}) \quad = \quad \mathbb{E}_{\widehat{W}}\{F(x^{\pi,N},\widehat{W})|S^0\}. \tag{5.10}$$

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

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

The estimate $\bar{F}(x^{\pi,N})$ evaluates a single decision $x^{\pi,N}$, which hints to the performance of the learning policy π.

5.2.4 Policy evaluation

If we wish to evaluate a policy $X^\pi(S^n)$, we have to combine all three types of uncertainty. This is done by computing

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

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

5.2.5 Closing notes

This section is hardly a comprehensive treatment of modeling uncertainty. Given the richness of this topic, chapter 10 is dedicated to describing the process of modeling uncertainty. The discussion here was to bring out the basic forms of uncertainty when evaluating algorithms for stochastic search.

We mention only in passing the growing interest in replacing the expectation $\mathbb{E}$ with some form of risk measure that recognizes that the possibility of extreme outcomes is more important that is represented by their likelihood (which may be low). Expectations average over all outcomes, so if extreme events occur with low probability, they do not have much effect on the solution. Also, expectations may have the effect of letting high outcomes cancel low outcomes, when in fact one tail is much more important than the other. We discuss risk in more detail in section 9.8.5. Replacing the expectation operator with some form of risk measure does not change the core steps when evaluating a policy.

5.3 STOCHASTIC GRADIENT METHODS

One of the oldest and earliest methods for solving our basic stochastic optimization problem

$$\max_x \mathbb{E}F(x,W). \tag{5.11}$$

uses the fact that we can often compute the gradient of $F(x,W)$ with respect to x after the random variable W becomes known. For example, assume that we are trying to solve a newsvendor problem, where we wish to allocate a quantity x of resources ("newspapers") before we know the demand W. The optimization problem is given by

$$\max_x F(x) = \mathbb{E}p\min\{x,W\} - cx. \tag{5.12}$$

If we could compute $F(x)$ exactly (that is, analytically), and its derivative, then we could find x^* by taking its derivative and setting it equal to zero as we did in section 4.2.2. If this is not possible, we could still use a classical steepest ascent algorithm

$$x^{n+1} \quad = \quad x^n + \alpha_n \nabla_x F(x^n), \tag{5.13}$$

where α_n is a stepsize. For deterministic problems, we typically choose the best stepsize by solving the one-dimensional optimization problem

$$\alpha^n = \arg\max_{\alpha\geq 0} F\big(x^n + \alpha\nabla F(x^n)\big). \tag{5.14}$$

For stochastic problems, we would have to be able to compute $F(x) = \mathbb{E}F(x,W)$ in (5.14), which is computationally intractable (otherwise we return to the techniques in chapter 4). This means that we cannot solve the one-dimensional search for the best stepsize in equation (5.14).

Instead, we resort to an algorithmic strategy known as stochastic gradients, where we use the gradient $\nabla_x F(x^n, W^{n+1})$, which means we wait until we observe W^{n+1} and then take the gradient of the function. This is not possible for all problems (hence the reason for chapter 7), but for problems where we can find the gradient, this overcomes the issues associated with computing the derivative of an expectation. The idea that we are allowed to wait until *after* we observe W^{n+1} before computing the gradient is the magic of stochastic gradient algorithms.

5.3.1 A stochastic gradient algorithm

For our stochastic problem, we assume that we either cannot compute $F(x)$, or we cannot compute the gradient exactly. However, there are many problems where, if we fix $W = W(\omega)$, we can find the derivative of $F(x, W(\omega))$ with respect to x. Then, instead of using the deterministic updating formula in (5.13), we would instead use

$$x^{n+1} = x^n + \alpha_n \nabla_x F(x^n, W^{n+1}). \tag{5.15}$$

Here, $\nabla_x F(x^n, W^{n+1})$ is called a *stochastic gradient* because it depends on a sample realization of W^{n+1}.

It is important to note our indexing. A variable such as x^n or α_n that is indexed by n is assumed to be a function of the observations $W^1, W^2, \ldots, W^n$, but not W^{n+1}. Thus, our stochastic gradient $\nabla_x F(x^n, W^{n+1})$ depends on our current solution x^n and the next observation W^{n+1}.

To illustrate, consider the simple newsvendor problem with the profit maximizing objective

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

In this problem, we order a quantity $x = x^n$ (determined at the end of day n), and then observe a random demand W^{n+1} that was observed the next day $n+1$. We earn a revenue given by $p \min\{x^n, W^{n+1}\}$ (we cannot sell more than we bought, or more than the demand), but we had to pay for our order, producing a negative cost $-cx$. Let $\nabla F(x^n, W^{n+1})$ be the sample gradient, taken when $W = W^{n+1}$. In our example, this is given by

$$\frac{\partial F(x^n, W^{n+1})}{\partial x} = \begin{cases} p - c & \text{If } x^n < W^{n+1}, \\ -c & \text{If } x^n > W^{n+1}. \end{cases} \tag{5.16}$$

The quantity x^n is the estimate of x computed from the previous iteration (using the sample realization ω^n), while W^{n+1} is the sample realization in iteration $n+1$ (the indexing tells us that x^n was computed without knowing W^{n+1}). When the function is deterministic, we would choose the stepsize by solving the one-dimensional optimization problem determined by (5.14).

5.3.2 Introduction to stepsizes

Now we face the problem of finding the stepsize α_n when we have to work with the stochastic gradient $\nabla F(x^n, W^{n+1})$. Unlike our deterministic algorithm, we cannot solve a one-dimensional search (as we did in (5.14)) to find the best stepsize after seeing W^{n+1}, simply because we cannot compute the expectation.

We overcome our inability to compute the expectation by working with stochastic gradients. While the computational advantages are tremendous, it means that the gradient is now a random variable. This means that the stochastic gradient can even point away from the optimal solution such that any positive stepsize actually makes the solution worse. Figure 5.1 compares the behavior of a deterministic search algorithm, where the solution improves at each iteration, and a stochastic gradient algorithm.

This behavior is easily illustrated using our newsvendor problem. It might be that the optimal order quantity is 15. However, even if we order $x = 20$, it is possible that the demand is 24 on a particular day, pushing us to move our order quantity to a number larger than 20, which is even further from the optimum.

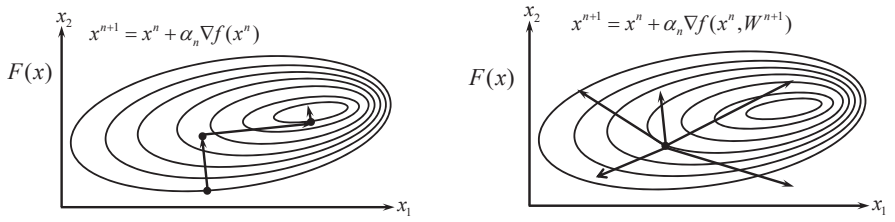


Figure 5.1 Illustration of gradient ascent for a deterministic problem (a), and stochastic gradients (b).

The major challenge when using stochastic gradients is the stepsize; we can no longer use the one-dimensional search as we did with our deterministic application in equation (5.14). Interestingly, when we are working on stochastic problems, we overcome our inability to solve the one-dimensional search problem by using relatively simple stepsize rules that we are going to call stepsize policies. For example, a classic formula is nothing more than

$$\alpha_n = \frac{1}{n+1} \tag{5.17}$$

for $n = 0, 1, \ldots$. With this formula, we can show that

$$\lim_{n\to\infty} x^n \to x^*, \tag{5.18}$$

where x^* is the optimal solution of our original optimization problem (5.1). But note - we did not promise that convergence was fast, only that it would converge (eventually). (See section 5.10 for proofs of this convergence.) Note that there is a *very* large literature that proves asymptotic convergence, but then runs the algorithm for a finite number of iterations and then just assumes the resulting solution is good.

There are many applications where the units of the gradient, and the units of the decision variable, are different. This happens with our newsvendor example, where the gradient is in units of dollars, while the decision variable x is in units of newspapers. This is a significant problem that causes headaches in practice.

A problem where we avoid this issue arises if we are trying to learn the mean of a random variable W. We can formulate this task as a stochastic optimization problem using

$$\min_x \mathbb{E}\frac{1}{2}(x - W)^2. \tag{5.19}$$

Here, our function $F(x, W) = \frac{1}{2}(x - W)^2$, and it is not hard to see that the value of x that minimizes this function is $x = \mathbb{E}W$. Now assume that we want to produce a sequence of estimates of $\mathbb{E}W$ by solving this problem using a sequential (online) stochastic gradient algorithm, which looks like

$$\begin{aligned} x^{n+1} &= x^n - \alpha_n \nabla F_x(x^n, W^{n+1}), && (5.20)\\ &= x^n - \alpha_n (x^n - W^{n+1}), \\ &= (1 - \alpha_n)x^n + \alpha_n W^{n+1}. && (5.21) \end{aligned}$$

Equation (5.20) illustrates α_n as the stepsize in a stochastic gradient algorithm, while equation (5.21) is exponential smoothing (see section 3.2). In this context, α_n is widely known as a smoothing factor or "learning rate."

There are going to be problems where our "one over n" stepsize formula (5.17) is very slow. However, for the problem of estimating the mean of a random variable, we are going to show in chapter 6 that "one over n" is actually *the optimal stepsize formula*!! That is, no other stepsize formula will give faster convergence. This is just a hint of the richness we are going to encounter with stepsize rules.

There are problems where we may start with a prior estimate of $\mathbb{E}W$ which we can express as x^0. In this case, we would want to use an initial stepsize $\alpha^0 < 1$. However, we often start with no information, in which case an initial stepsize $\alpha^0 = 1$ gives us

$$\begin{aligned} x^1 &= (1-\alpha_0)x^0 + \alpha_0 W^1 \\ &= W^1, \end{aligned}$$

which means we do not need the initial estimate for x^0. Smaller initial stepsizes would only make sense if we had access to a reliable initial guess, and in this case, the stepsize should reflect the confidence in our original estimate (for example, we might be warm starting an algorithm from a previous iteration).

This section is just a peek into stepsizes. We cover this rich topic in considerably more detail in chapter 6.

5.3.3 Evaluating a stochastic gradient algorithm

In section 5.10 we are going to provide two proofs of asymptotic optimality. The problem is that we never run these algorithms to the limit, which means we are only interested in our finite time performance. If we are only interested in the quality of our final solution $x^{\pi,N}$, then we want to use the final reward objective given by (5.4), but this raises the issue: How do we compute this? The answer is that we have to simulate it.

Let ω^ℓ be a sample realization of our random variables $W^1(\omega^\ell), \ldots, W^N(\omega^\ell)$ that we use for training (estimating) $x^{\pi,N}(\omega^\ell)$ for $\ell = 1, 2, \ldots, L$. Then let ψ^k be a sample realization of our testing information $\widehat{W}(\psi^k)$, for $k = 1, 2, \ldots, K$. Assume that there is no probabilistic information in S^0. We can estimate the performance of our algorithm π using

$$\bar{F}^\pi = \frac{1}{L}\sum_{\ell=1}^{L}\left(\frac{1}{K}\sum_{k=1}^{K} F(x^{\pi,N}(\omega^\ell), \widehat{W}(\psi^k))\right), \tag{5.22}$$

where $x^n(\omega^\ell) = X^\pi(S^n(\omega^\ell))$ is determined by our stochastic gradient formula (5.20) and α_n comes from our stepsize formula (say, equation (5.17)). For this problem our state variable $S^n = x^n$, which means that our state transition equation $S^{n+1}(\omega^\ell) = S^M(S^n(\omega^\ell), x^n(\omega^\ell), W^{n+1}(\omega^\ell))$ is just the stochastic gradient (5.20). We then let $x^{\pi,N} = x^N$ be the ending point.

The final reward objective in (5.22) is easily the most classical way of evaluating a stochastic search algorithm, but there are several arguments to be made for using the cumulative reward, which we would simulate using

$$\bar{F}^\pi = \frac{1}{L}\sum_{\ell=1}^{L}\left(\sum_{n=0}^{N-1} F(x^n(\omega^\ell), W^{n+1}(\omega^\ell))\right). \tag{5.23}$$

It is possible that we have to apply this algorithm in a field situation such as a real newsvendor problem, where we have to live with the results of each solution x^n. However, we may simply be interested in the overall rate of convergence, which would be better captured by (5.23).

5.3.4 A note on notation

Throughout this book, we index variables (whether we are indexing by iterations or time) to clearly identify the *information content* of each variable. Thus, x^n is the decision made after W^n becomes known. When we compute our stochastic gradient $\nabla_x F(x^n, W^{n+1})$, we use x^n which was determined after observing W^n. If the iteration counter refers to an experiment, then it means that x^n is determined after we finish the n^{th} experiment. If we are solving a newsvendor problem where n indexes days, then it is like determining the amount of newspapers to order for day $n+1$ after observing the sales for day n. If we are performing a laboratory experiment, we use the information up through the first n experiments to choose x^n, which specifies the design settings for the $n+1^{st}$ experiment. This indexing makes sense when you realize that the index n reflects the information content, not when it is being implemented.

In chapter 6, we are going to present a number of formulas to determine stepsizes. Some of these are deterministic, such as $\alpha_n = 1/n$, and some are stochastic, adapting to information as it arrives. Our stochastic gradient formula in equation (5.15) communicates the property that the stepsize α_n that is multiplied times the gradient $\nabla_x F(x^n, W^{n+1})$ is allowed to see W^n and x^n, but not W^{n+1}.

We return to this issue in chapter 9, but we urge readers to adopt this notational system.

5.4 STYLES OF GRADIENTS

There are a few variants of the basic stochastic gradient method. Below we introduce the idea of gradient smoothing and describe a method for approximating a second-order algorithm.

5.4.1 Gradient smoothing

In practice, stochastic gradients can be *highly* stochastic, which is the reason why we have to use stepsizes. However, it is possible to mitigate some of the variability by smoothing the gradient itself. If $\nabla F(x^n, W^{n+1})$ is our stochastic gradient, computed after the $n+1^{st}$ experiment, we could then smooth this using

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

where η is a smoothing factor where $0 < \eta \leq 1$. We could replace this with a declining sequence η_n, although common practice is to keep this process as simple as possible. Regardless of the strategy, gradient smoothing has the effect of introducing at least one more tunable parameter. The open empirical question is whether gradient smoothing adds anything beyond the smoothing produced by the stepsize policy used for updating x^n.

5.4.2 Second-order methods

Second order methods for deterministic optimization have proven to be particularly attractive. For smooth, differentiable functions, the basic update step looks like

$$x^{n+1} = x^n + (H^n)^{-1}\nabla_x f(x^n), \tag{5.24}$$

where H^n is the Hessian, which is the matrix of second derivatives. That is,

$$H^n_{xx'} = \left.\frac{\partial^2 f(x)}{\partial x \partial x'}\right|_{x=x^n}.$$

The attraction of the update in equation (5.24) is that there is no stepsize. The reason (and this requires that $f(x)$ be smooth with continuous first derivatives) is that the inverse Hessian solves the problem of scaling. In fact, if $f(x)$ is quadratic, then equation (5.24) takes us to the optimal solution in one step!

Since functions are not always as nice as we would like, it is sometimes useful to introduce a constant "stepsize" α, giving us

$$x^{n+1} = x^n + \alpha(H^n)^{-1}\nabla_x f(x^n),$$

where $0 < \alpha \leq 1$. Note that this smoothing factor does not have to solve any scaling problems (again, this is solved by the Hessian).

If we have access to second derivatives (which is not always the case), then our only challenge is inverting the Hessian. This is not a problem with a few dozen or even a few hundred variables, but there are problems with thousands to tens of thousands of variables. For large problems, we can strike a compromise and just use the diagonal of the Hessian. This is both much easier to compute, as well as being easy to invert. Of course, we lose some of the fast convergence (and scaling).

There are many problems (including all stochastic optimization problems) where we do not have access to Hessians. One strategy to overcome this is to construct an approximation of the Hessian using what are known as rank-one updates. Let $\bar{H}^n$ be our approximate Hessian which is computed using

$$\bar{H}^{n+1} = \bar{H}^n + \nabla f(x^n)(\nabla f(x^n))^T. \tag{5.25}$$

Recall that $\nabla f(x^n)$ is a column vector, so $\nabla f(x^n)(\nabla f(x^n))^T$ is a matrix with the dimensionality of x. Since it is made up of an outer product of two vectors, this matrix has rank 1.

This methodology could be applied to a stochastic problem. As of this writing, we are not aware of any empirical study showing that these methods work, although there has been recent interest in second-order methods for online machine learning.

5.4.3 Finite differences

It is often the case that we do not have direct access to a derivative. Instead, we can approximate gradients using finite differences which requires running the simulation multiple times with perturbed inputs.

Assume that x is a P-dimensional vector, and let e_p be a P-dimensional column vector of zeroes with a 1 in the p^{th} position. Let $W_p^{n+1,+}$ and $W_p^{n+1,-}$ be sequences of random variables that are generated when we run each simulation, which would be run in the $n+1^{st}$ iteration. The subscript p only indicates that these are the random variables for the p^{th} run.

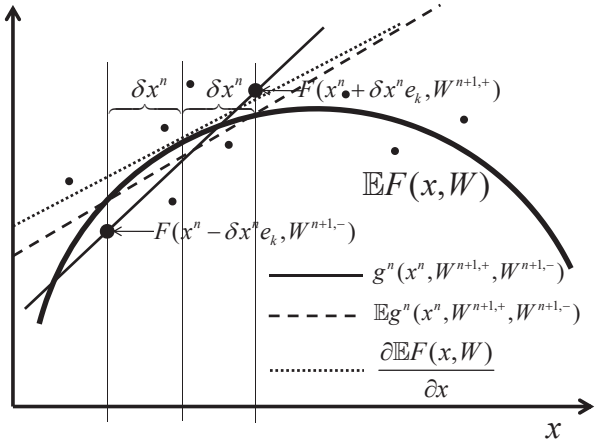


Figure 5.2 Different estimates of the gradient of $F(x, W)$ with a) the stochastic gradient $g^n(x^n, W^{n+1,+}, W^{n+1,-})$ (solid line), the expected finite difference $\mathbb{E}g^n(x^n, W^{n+1,+}, W^{n+1,-})$ (dashed line), and the exact slope at x^n, $\partial \mathbb{E}F(x^n, W^{n+1})/\partial x^n$.

Now assume that we can run two simulations for each dimension, $F(x^n+\delta x^n e_p, W_p^{n+1,+})$ and $F(x^n - \delta x^n e_p, W_p^{n+1,-})$ where $\delta x^n e_p$ is the change in x^n, multiplied by e_p so that we are only changing the p^{th} dimension. Think of $F(x^n + \delta x^n e_p, W_p^{n+1,+})$ and $F(x^n - \delta x^n e_p, W_p^{n+1,-})$ as calls to a black-box simulator where we start with a set of parameters x^n, and then perturb it to $x^n + \delta x^n e_p$ and $x^n - \delta x^n e_p$ and run two separate, independent simulations. We then have to do this for each dimension p, allowing us to compute

$$g_p^n(x^n, W^{n+1,+}, W^{n+1,-}) = \frac{F(x^n + \delta x^n e_p, W_p^{n+1,+}) - F(x^n - \delta x^n e_p, W_p^{n+1,-})}{2\delta x_p^n}, \tag{5.26}$$

where we divide the difference by the width of the change, given by $2\delta x_p^n$, to get the slope.

The calculation of the derivative (for one dimension) is illustrated in figure 5.2. We see from figure 5.2 that shrinking δx can introduce a lot of noise in the estimate of the gradient. At the same time, as we increase δx, we introduce bias, which we see in the difference between the dashed line showing $\mathbb{E}g^n(x^n, W^{n+1,+}, W^{n+1,-})$, and the dotted line that depicts $\partial \mathbb{E}F(x^n, W^{n+1})/\partial x^n$. If we want an algorithm that converges asymptotically in the limit, we need δx^n decreasing, but in practice it is often set to a constant δx, which is then handled as a tunable parameter.

Finite differences can be expensive. Running a function evaluation can require seconds to minutes, but there are computer models that can take hours or days (or more) to run. Equation (5.26) requires $2P$ function evaluations, which can be especially problematic when $F(x, W)$ is an expensive simulation, as well as when the number of dimensions P is large. Fortunately, these simulations can often be run in parallel. In the next section we introduce a strategy for handling multidimensional parameter vectors.

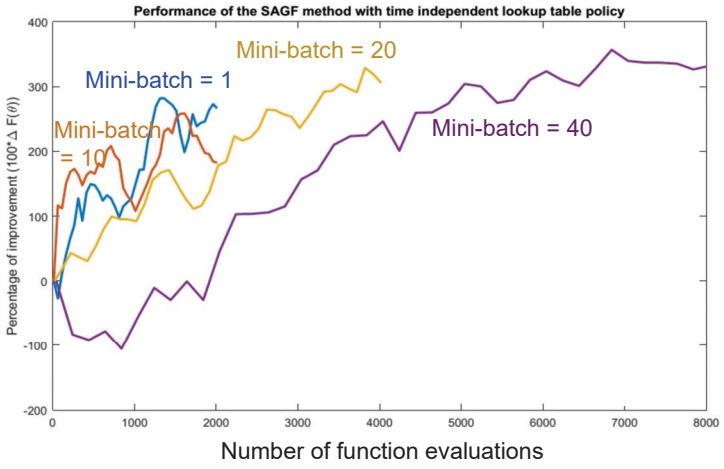


Figure 5.3 Convergence of SPSA for different mini-batch sizes, showing the slower convergence to a better with larger mini-batches.

5.4.4 SPSA

A powerful method for handling higher-dimensional parameter vectors is *simultaneous perturbation stochastic approximation* (or SPSA). SPSA computes gradients in the following way. Let $Z_p, p = 1, \ldots, P$ be a vector of zero-mean random variables, and let Z^n be a sample of this vector at iteration n. We approximate the gradient by perturbing x^n by the vector Z using $x^n + \eta^n Z^n$ and $x^n - \eta^n Z^n$, where η^n is a scaling parameter that may be a constant over iterations, or may vary (typically it will decline). Now let $W^{n+1,+}$ and $W^{n+1,-}$ represent two different samples of the random variables driving the simulation (these can be generated in advance or on the fly). We then run our simulation twice: once to find $F(x^n + \eta^n Z^n, W^{n+1,+})$, and once to find $F(x^n - \eta^n Z^n, W^{n+1,-})$. The estimate of the gradient is then given by

$$g^n(x^n, W^{n+1,+}, W^{n+1,-}) = \begin{bmatrix} \frac{F(x^n+\eta^n Z^n, W^{n+1,+}) - F(x^n-\eta^n Z^n, W^{n+1,-})}{2\eta^n Z_1^n} \\ \frac{F(x^n+\eta^n Z^n, W^{n+1,+}) - F(x^n-\eta^n Z^n, W^{n+1,-})}{2\eta^n Z_2^n} \\ \vdots \\ \frac{F(x^n+\eta^n Z^n, W^{n+1,+}) - F(x^n-\eta^n Z^n, W^{n+1,-})}{2\eta^n Z_P^n} \end{bmatrix} \quad (5.27)$$

Note that the numerator of each element of g^n in equation (5.27) is the same, which means we only need two function evaluations: $F(x^n + \eta^n Z^n, W^{n+1,+})$ and $F(x^n - \eta^n Z^n, W^{n+1,-})$. The only difference is the Z_p^n in the denominator for each dimension p.

The real power of SPSA arises in applications where simulations are noisy, and these can be *very* noisy in many settings. A way to overcome this is with the use of "mini-batches" where the simulations to compute $F(x^n + \eta^n Z^n, W^{n+1,+})$ and $F(x^n - \eta^n Z^n, W^{n+1,-})$ are run, say, M times and averaged. Keep in mind that these can be done in parallel; this does not mean they are free, but if you have access to parallel processing capability (which is quite common), it means that repeated simulations may not add to the completion time for your algorithm.

Figure 5.3 illustrates the effect of mini-batches; larger mini-batches produce slower initial performance, but better performance over more iterations. Note that figure shows performance in terms of function evaluations, not CPU time, so the benefits of parallel computing are ignored. This graphic suggests a strategy of using increasing mini-batch sizes. Smaller mini-batches work well in the beginning, while larger mini-batches help as the algorithm progresses.

SPSA seems like magic: we are getting a P-dimensional gradient from just two function evaluations, regardless of the value of P. The open question is the rate of convergence, which will depend very much on the characteristics of the problem at hand. A reader will naturally ask: "Does it work?" The unqualified answer is: "It *can* work," but you will need to spend time understanding the characteristics of your problem, and tuning the algorithmic choices of SPSA, notably:

- Choice of stepsize formula, and tuning of any stepsize parameters (there is always at least one). Be careful with tuning, as it may depend on the starting point of your algorithm x^0, as well as other problem characteristics.
- The choice of mini-batch size. SPSA is trying to get a lot of information from just two function evaluations, so there is going to be a price to be paid in terms of convergence rates. A key issue here is whether you have access to parallel computing resources.
- You may also experiment with gradient smoothing, which is another way to stabilize the algorithm but without the price of repeated simulations required by mini-batches. This introduces the additional dimension of tuning the smoothing factor for gradient smoothing.
- Don't forget that all gradient-based methods are designed for maximizing concave functions (minimizing convex functions), but your function may not be concave. For complex problems, it is not necessarily easy (or even possible) to verify the behavior of the function, especially for higher dimensional problems (three or more).

5.4.5 Constrained problems

There are problems where x has to stay in a feasible region $\mathcal{X}$, which might be described by a system of linear equations such as

$$\mathcal{X} = \{x | Ax = b, x \geq 0\}.$$

When we have constraints, we would first compute

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

which might produce a solution y^{n+1} which does not satisfy the constraints. To handle this we project y^{n+1} using a projection step that we write using

$$x^{n+1} \leftarrow \Pi_{\mathcal{X}}[y^{n+1}].$$

The definition of the projection operator $\Pi_{\mathcal{X}}[\cdot]$ is given by

$$\Pi_{\mathcal{X}}[y] = \arg\min_{x \in \mathcal{X}} \|x - y\|_2, \tag{5.28}$$

where $\|x - y\|_2$ is the "L_2 norm" defined by

$$\|x - y\|_2 = \sum_i (x_i - y_i)^2.$$

The projection operator $\Pi_{\mathcal{X}}[\cdot]$ can often be solved easily by taking advantage of the structure of a problem. For example, we may have box constraints of the form $0 \leq x_i \leq u_i$. In this case, any element x_i falling outside of this range is just mapped back to the nearest boundary (0 or u_i).

5.5 PARAMETER OPTIMIZATION FOR NEURAL NETWORKS*

In section 3.9.3 we descibed how to produce an estimate from a neural network given the set of parameters. Now we are going to show how to estimate these parameters using the stochastic gradient concepts that we presented in this chapter.

We are going to show how to derive the gradient for the three-layer network in figure 5.4. We will use the following relationships that we first derived in section 3.9.3 from the forward pass:

$$\bar{f}(x^n|\theta) = \sum_{i\in\mathcal{I}^{(1)}} \sum_{j\in\mathcal{I}^{(2)}} \theta_{ij}^{(1)} x_i^{(1,n)}, \tag{5.29}$$

$$y_j^{(2,n)} = \sum_{i\in\mathcal{I}^{(1)}} \theta_{ij}^{(1)} x_i^{(1,n)}, \quad j \in \mathcal{I}^{(2)}, \tag{5.30}$$

$$x_i^{(2,n)} = \sigma(y_i^{(2,n)}), \ i \in \mathcal{I}^{(2)}, \tag{5.31}$$

$$y_j^{(3,n)} = \sum_{i\in\mathcal{I}^{(2)}} \theta_{ij}^{(2)} x_i^{(2,n)}, \quad j \in \mathcal{I}^{(3)}, \tag{5.32}$$

$$x_i^{(3,n)} = \sigma(y_i^{(3,n)}), \ i \in \mathcal{I}^{(3)}. \tag{5.33}$$

Recall that $\sigma(y)$ is the sigmoid function

$$\sigma(y) = \frac{1}{1 + e^{-\beta y}}, \tag{5.34}$$

and that $\sigma'(y) = \frac{\partial \sigma(y)}{\partial y}$.

We are going to start by showing how to compute the gradient. Then, we will present the stochastic gradient algorithm and discuss some issues that arise in the context of neural networks.

5.5.1 Computing the gradient

We compute the stochastic gradient $\nabla_\theta F(\theta)$ for a given input x^n and observed response $\hat{f}^{n+1}$. $\hat{f}^{n+1}$ plays the role of W^{n+1} in our original derivation.

Assume that we are given an input x^n. If we follow the forward instructions above, our final estimate is produced by

$$\bar{f}(x^n|\theta) = \sum_{i\in\mathcal{I}^{(3)}} \theta_i^{(3)} x_i^{(3,n)}. \tag{5.35}$$

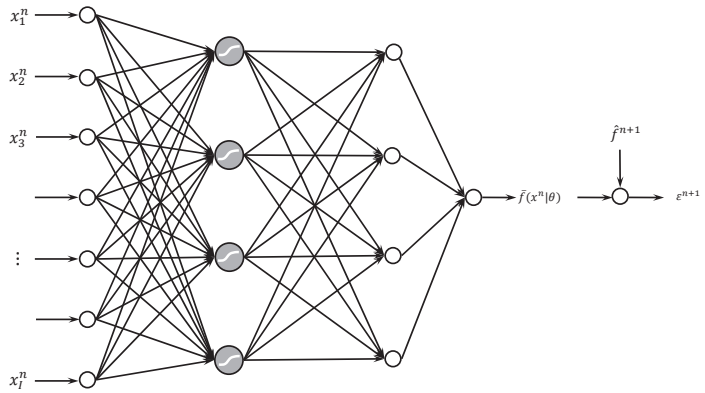


Figure 5.4 A three-layer neural network.

The goal is to find θ that solves

$$\min_{\theta} F(\theta) = \mathbb{E}\frac{1}{2}\sum_{n=1}^{N-1}(\bar{f}(x^n|\theta) - \hat{f}^{n+1})^2. \tag{5.36}$$

We want to understand the effect of a change in θ given an input x^n and response $\hat{f}^{n+1}$. Specifically, we want the gradient $\nabla_\theta F(\theta)$. First, since we cannot compute the expectation, we are going to compute a *stochastic gradient* which means we are going to replace $F(\theta)$ with the function evaluated for a particular x^n, given the response $\hat{f}^{n+1}$ which we write as

$$F(x^n, \hat{f}^{n+1}|\theta) = \frac{1}{2}(\bar{f}(x^n|\theta) - \hat{f}^{n+1})^2.$$

Computing $\nabla_\theta F(x^n, \hat{f}^{n+1}|\theta)$ will prove to be a nice exercise in applying the chain rule. We are going to compute the gradient by stepping backward through the neural network in figure 5.4. Our hope is that by illustrating how to do it for this network, the process of extending this to other neural networks will be apparent.

We start with the derivative with respect to $\theta^{(3)}$:

$$\frac{\partial F(\theta|x^n, \hat{f}^{n+1})}{\partial \theta_i^{(3)}} = (\bar{f}(x^n|\theta) - \hat{f}^{n+1})\frac{\partial \bar{f}(x^n|\theta)}{\partial \theta_i^{(3)}}, \tag{5.37}$$

$$= (\bar{f}(x^n|\theta) - \hat{f}^{n+1})x_i^{(3,n)}. \tag{5.38}$$

where (5.38) comes from differentiating (5.29). The derivation of the gradient with respect to $\theta^{(2)}$ is given by

$$\frac{\partial F(\theta|x^n, \hat{f}^{n+1})}{\partial \theta_{ij}^{(2)}} = (\bar{f}(x^n|\theta) - \hat{f}^{n+1})\frac{\partial \bar{f}(x^n|\theta)}{\partial \theta_{ij}^{(2)}}, \tag{5.39}$$

$$\frac{\partial \bar{f}(x^n|\theta)}{\partial \theta_{ij}^{(2)}} = \frac{\partial \bar{f}(x^n|\theta)}{\partial x_j^{(3,n)}}\frac{\partial x_j^{(3,n)}}{\partial y_j^{(3)}}\frac{\partial y_j^{(3,n)}}{\partial \theta_{ij}^{(2)}}, \tag{5.40}$$

$$= \theta_j^{(3)}\sigma'(y_j^{(3,n)})x_i^{(2,n)}. \tag{5.41}$$

Remember that $\sigma'(y)$ is the derivative of our sigmoid function (equation (3.58)) with respect to y.

Finally, the gradient with respect to $\theta^{(1)}$ is found using

$$\frac{\partial F(\theta|x^n,\hat{f}^{n+1})}{\partial\theta_{ij}^{(1)}} = (\bar{f}(x^n|\theta)-\hat{f}^{n+1})\frac{\partial\bar{f}(x^n|\theta)}{\partial\theta_{ij}^{(1)}}, \tag{5.42}$$

$$\frac{\partial\bar{f}(x^n|\theta)}{\partial\theta_{ij}^{(1)}} = \sum_k \frac{\partial\bar{f}(x^n|\theta)}{\partial x_k^{(3)}}\frac{\partial x_k^{(3,n)}}{\partial y_k^{(3)}}\frac{\partial y_k^{(3)}}{\partial\theta_{ij}^{(1)}}, \tag{5.43}$$

$$= \sum_k \theta_k^{(3)}\sigma'(y_k^{(3)})\frac{\partial y_k^{(3)}}{\partial\theta_{ij}^{(1)}}, \tag{5.44}$$

$$\frac{\partial y_k^{(3)}}{\partial\theta_{ij}^{(1)}} = \frac{\partial y_k^{(3)}}{\partial x_j^{(2)}}\frac{\partial x_j^{(2)}}{\partial\theta_{ij}^{(1)}}, \tag{5.45}$$

$$= \frac{\partial y_k^{(3)}}{\partial x_j^{(2)}}\frac{\partial x_j^{(2)}}{\partial y_j^{(2)}}\frac{\partial y_j^{(2)}}{\partial\theta_{ij}^{(1)}}, \tag{5.46}$$

$$= \theta_{jk}^{(2)}\sigma'(y_j^{(2)})x_i^{(1)}. \tag{5.47}$$

Combining the above gives us

$$\frac{\partial F(\theta|x^n,\hat{f}^{n+1})}{\partial\theta_i^{(1)}} = (\bar{f}(x^n|\theta)-\hat{f}^{n+1})\left(\sum_k \theta_k^{(3)}\sigma'(y_k^{(3)})\theta_{jk}^{(2)}\right)\sigma'(y_j^{(2)})x_i^{(1)},$$

$$\frac{\partial F(\theta|x^n,\hat{f}^{n+1})}{\partial\theta_i^{(2)}} = (\bar{f}(x^n|\theta)-\hat{f}^{n+1})\theta_j^{(3)}\sigma'(y_j^{(3,n)})x_i^{(2,n)},$$

$$\frac{\partial F(\theta|x^n,\hat{f}^{n+1})}{\partial\theta_i^{(3)}} = (\bar{f}(x^n|\theta)-\hat{f}^{n+1})x_i^{(3,n)}.$$

The complete stochastic gradient is then given by

$$\nabla_\theta F(\theta|x^n,\hat{f}^{n+1}|\theta) = \begin{pmatrix} \nabla_{\theta^{(1)}}F(x^n,\hat{f}^{n+1}|\theta) \\ \nabla_{\theta^{(2)}}F(x^n,\hat{f}^{n+1}|\theta) \\ \nabla_{\theta^{(3)}}F(x^n,\hat{f}^{n+1}|\theta) \end{pmatrix}.$$

We are now ready to execute our parameter search using a stochastic gradient algorithm.

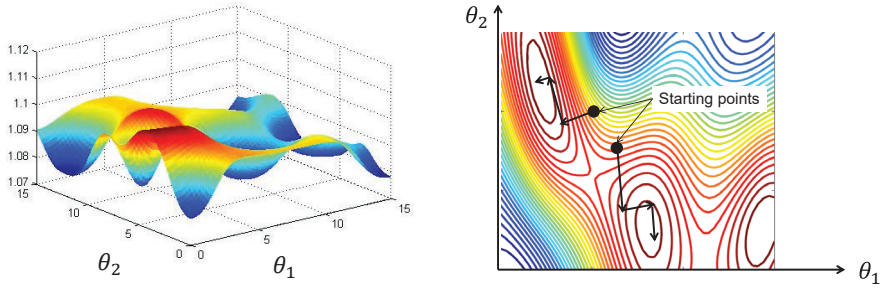
5.5.2 The stochastic gradient algorithm

The search for θ is done using a basic stochastic gradient algorithm given by

$$\theta^{n+1} = \theta^n - \alpha_n\nabla_\theta F(\theta^n,\hat{f}^{n+1}). \tag{5.48}$$

We return to this method in considerably more detail in chapter 5. In particular, we have an entire chapter devoted to the design of the stepsize α_n, although for now we note that we could use a formula as simple as

$$\alpha_n = \frac{\theta^{step}}{\theta^{step}+n-1}.$$



(a) (b)

Figure 5.5 (a) Illustration of nonconvex behavior of the response surface for $F(\theta)$; (b) the path to local minima from two different starting points.

For now, we are going to focus on the properties of the function $F(\theta)$ in equation (5.36). In particular, readers need to be aware that the function $F(\theta)$ is highly nonconvex, as illustrated in figure 5.5(a) for a two-dimensional problem. Figure 5.5(b) shows that when we start from two different starting points, we can end up at two different local minima. This behavior is typical of nonlinear models, but it especially true of neural networks.

The lack of convexity in the objective function $F(\theta)$ is well known to the neural network community. One strategy is to try a number of different starting points, and then use the best of the optimized values of θ. The real issue, of course, is not which θ produces the lowest error for a particular dataset, but which θ produces the best performance with new data.

This behavior also complicates using neural networks in an online setting. If we have fitted a neural network and then add one more data point, there is not a natural process for incrementally updating the estimate of θ. Simply doing one iteration of the gradient update in (5.48) accomplishes very little, since we are never truly at the optimum.

5.6 STOCHASTIC GRADIENT ALGORITHM AS A SEQUENTIAL DECISION PROBLEM

We think of stochastic gradient algorithms as methods for solving a problem such as our basic optimization problem in (5.1). However, designing a stochastic gradient algorithm can itself be formulated as a sequential decision problem and modelled using the canonical framework we presented in section 2.2.

We start by restating our stochastic gradient algorithm

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

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

$$\alpha_n(\theta^{kest}) = \frac{\theta^{kest}}{\theta^{kest} + N^n}, \tag{5.49}$$

State variables $S^n = (x^n, N^n)$.

Decision variables The stepsize α_n. This is determined by Kesten's stepsize policy in equation (5.49) which is parameterized by θ.

Exogenous information W^{n+1}, which depends on the motivating problem. This could involve observing the demand in a newsvendor problem, or it may involve running a simulator and observing $\hat{F}^{n+1} = F(x^n, W^{n+1})$.

Transition function These consist of equations for each state variable:

$$\begin{aligned} x^{n+1} &= x^n + \alpha_n \nabla_x F(x^n), \\ N^{n+1} &= \begin{cases} N^n + 1 & \text{if } (\nabla_x F(x^{n-1}))^T \nabla_x F(x^n) < 0, \\ N^n & \text{otherwise.} \end{cases} \end{aligned}$$

Note that $\nabla_x F(x^n)$ may be approximated using numerical derivatives such as SPSA.

Objective function We wish to maximize the performance of the final solution that we call $x^{\pi,N}$, which means we wish to optimize:

$$\max_{\pi} \mathbb{E}_{S^0} \mathbb{E}_{W^1,\ldots,W^N|S^0} \mathbb{E}_{\widehat{W}|S^0} F(x^{\pi,N}, \widehat{W}). \tag{5.50}$$

If we limit ourselves to Kesten's rule, then we can write the objective in terms of optimizing over θ^{kest}. More generally, we might want to search over different classes of stepsize rules, each of which is likely to have its own tunable parameter.

Figure 5.6 A stochastic gradient algorithm as a sequential decision problem.

where θ^{kest} is a tunable parameter and N^n counts the number of times that the gradient $\nabla_x F(x^n)$ has changed sign, which means that

$$(\nabla_x F(x^{n-1}))^T \nabla_x F(x^n) < 0.$$

When this inner product is negative, it means that the algorithm is starting to criss-cross which is an indication that it is in the vicinity of the optimum.

The stochastic gradient algorithm is stated as a sequential decision problem in figure 5.6. Formulating the stochastic gradient algorithm as a sequential decision problem produces an optimization problem that looks for the best *algorithm*. Here, we have limited ourselves to a) the use of a stochastic gradient algorithm, and b) the use of Kesten's stepsize policy. This means that we are just optimizing over the tunable parameter θ^{kest} which implies that we are optimizing within a class of algorithms, which is quite common.

There is a substantial literature on stochastic optimization algorithms which prove asymptotic convergence, and then examine the rate of convergence empirically. The goal stated in (5.50) of finding an *optimal algorithm* is aspirational, but formalizes what people are trying to achieve in practice. In chapter 6, we are going to review a variety of stepsize policies. We could search over all of these, although this is never done.

Even if we limit ourselves to a single class of stepsize policy, it is easy to overlook that tuning θ^{kest} depends on problem parameters such as the starting point x^0. It is easy to overlook the dependence of tuned parameters on problem data. This issue has been largely overlooked by the research community.

In section 1.4, we introduced four classes of policies that will describe any search over policies. Virtually all stochastic gradient algorithms, however, use policies from the first class, policy function approximations. This raises the question whether any of the other three classes might work well. As of this writing, we are not aware of any work to explore these options.

5.7 EMPIRICAL ISSUES

Invariably, the process of actually implementing these algorithms raises issues that are often ignored when describing the algorithms. To help mitigate this transition, below are some of the challenges an experimentalist is likely to encounter.

Tunable parameters - Arguably one of the most frustrating aspects of any algorithm is the need to tune parameters. For gradient-based algorithms, this typically refers to the tunable parameters in the stepsize policy, but could include a smoothing factor for gradient smoothing. These tunable parameters are a direct result of the use of first-order algorithms, which are easy to compute but which exploit very little information about the underlying function. Particularly frustrating is that this tuning really matters. A poorly tuned stepsize algorithm may decrease too quickly, creating apparent convergence. It is completely possible that a poorly tuned stepsize policy can result in a conclusion that an algorithm is not working. Stepsizes that are too large can introduce too much noise.

Scaling - In most (but not all) applications, the units of the gradient $\nabla F(x, W)$ are different than the units of x. A rough rule is that the initial stepsize should be chosen so that the initial change in x is on the order of 30 to 50 percent of the starting value.

Benchmarking - Whenever possible, it helps to run an algorithm on a simpler problem where the optimal solution can be found using other means, either analytically or numerically. For example, it might be possible to apply the stochastic gradient algorithm on a deterministic sequence that can be solved using deterministic algorithms.

Robustness - A desirable property of any algorithm is that it work reliably, on any problem instance (that is, within a problem class). For example, tuning parameters in the stepsize policy is annoying, but bearable if it only has to be done once.

5.8 TRANSIENT PROBLEMS*

There are many applications where we are trying to solve our basic stochastic optimization problem in an online setting, where the random variable W comes from field observations. In these settings, it is not unusual to find that the underlying distribution describing W is changing over time. For example, the demands in our newsvendor application may be changing as the purchasing patterns of the market change.

We tend to design algorithms so they exhibit asymptotic convergence. For example, we would insist that the stepsize α_n decline to zero as the algorithm progresses. In a transient setting, this is problematic because it means we are putting decreasing emphasis on the latest information, which is more important than older information. Over time, as α_n approaches zero, the algorithm will stop responding to new information. If we use a

stepsize such as $\alpha_n = 1/n$, it is possibly to show that the algorithm will eventually adapt to new information, but the rate of adaptation is so slow that the results are not useful.

Practitioners avoid this problem by either choosing a constant stepsize, or one that starts large but converges to a constant greater than zero. If we do this, the algorithm will start bouncing around the optimum. While this behavior may seem undesirable, in practice this is preferable, partly because the optimum of stochastic optimization problems tend to be smooth, but mostly because it means the algorithm is still adapting to new information, which makes it responsive to a changing signal.

5.9 THEORETICAL PERFORMANCE*

In practice, finding and tuning search algorithms tends to be ad hoc. Formal analysis of search algorithms tends to fall in one of three categories:

Asymptotic convergence - Probably the most standard result for an algorithm is a proof that the solution will asymptotically approach the optimal solution (that is, as the number of iterations $N \rightarrow \infty$). The criticism of asymptotic convergence is that it says nothing about rate of convergence, which means it is not telling us anything about the quality of the solution after N iterations. See section 5.10.2 and 5.10.3 in the appendix for samples of asymptotic convergence proofs.

Finite-time bounds - These are results that suggest that the quality of the solution after N iterations is within some limit. These bounds tend to be quite weak, and almost always feature unknown coefficients.

Asymptotic rate of convergence - It is often possible to provide high quality estimates of the rate of convergence, but only when the solution is in the vicinity of the optimal.

The holy grail of theoretical analysis of algorithms is tight bounds for the performance after n iterations. These are rare, and are limited to very simple problems. For this reason, empirical analysis of algorithms remains an important part of the design and analysis of search algorithms. Frustratingly, the performance of a search algorithm on one dataset may not guarantee good performance on a different dataset, even for the same problem class. We anticipate that this is typically due to a failure to properly tune the algorithm.

5.10 WHY DOES IT WORK?

Stochastic approximation methods have a rich history starting with the seminal paper Robbins & Monro (1951) and followed by Blum (1954*b*) and Dvoretzky (1956). The serious reader should see Kushner & Yin (1997) for a modern treatment of the subject. Wasan (1969) is also a useful reference for fundamental results on stochastic convergence theory. A separate line of investigation was undertaken by researchers in eastern European community focusing on constrained stochastic optimization problems (Gaivoronski (1988), Ermoliev (1988), Ruszczyński (1980), Ruszczyński (1987)). This work is critical to our fundamental understanding of Monte Carlo-based stochastic learning methods.

The theory behind these proofs is fairly deep and requires some mathematical maturity. For pedagogical reasons, we start in section 5.10.1 with some probabilistic preliminaries, after which section 5.10.2 presents one of the original proofs, which is relatively more accessible and which provides the basis for the universal requirements that stepsizes must

satisfy for theoretical proofs. Section 5.10.3 provides a more modern proof based on the theory of martingales.

5.10.1 Some probabilistic preliminaries

The goal in this section is to prove that these algorithms work. But what does this mean? The solution $\bar{x}^n$ at iteration n is a random variable. Its value depends on the sequence of sample realizations of the random variables over iterations 1 to n. If $\omega = (W^1, W^2, \ldots, W^n, \ldots)$ represents the sample path that we are following, we can ask what is happening to the limit $\lim_{n\rightarrow\infty} \bar{x}^n(\omega)$. If the limit is x^*, does x^* depend on the sample path ω?

In the proofs below, we show that the algorithms converge *almost surely*. What this means is that

$$\lim_{n\rightarrow\infty} \bar{x}^n(\omega) = x^*$$

for all $\omega \in \Omega$ that can occur with positive measure. This is the same as saying that we reach x^* with probability 1. Here, x^* is a deterministic quantity that does not depend on the sample path. Because of the restriction $p(\omega) > 0$, we accept that in theory, there could exist a sample outcome that can never occur that would produce a path that converges to some other point. As a result, we say that the convergence is "almost sure," which is universally abbreviated as "*a.s.*" Almost sure convergence establishes the core theoretical property that the algorithm will eventually settle in on a single point. This is an important property for an algorithm, but it says nothing about the rate of convergence (an important issue in approximate dynamic programming).

Let $x \in \Re^n$. At each iteration n, we sample some random variables to compute the function (and its gradient). The sample realizations are denoted by W^n. We let $\omega = (W^1, W^2, \ldots,)$ be a realization of all the random variables over all iterations. Let Ω be the set of all possible realizations of ω, and let $\mathfrak{F}$ be the σ-algebra on Ω (that is to say, the set of all possible events that can be defined using Ω). We need the concept of the history up through iteration n. Let

$$H^n = \text{a random variable giving the history of all random variables up through iteration } n.$$

A sample realization of H^n would be

$$\begin{aligned} h^n &= H^n(\omega) \\ &= (W^1, W^2, \ldots, W^n). \end{aligned}$$

We could then let W^n be the set of all outcomes of the history (that is, $h^n \in H^n$) and let $\mathcal{H}^n$ be the σ-algebra on W^n (which is the set of all events, including their complements and unions, defined using the outcomes in W^n). Although we could do this, this is not the convention followed in the probability community. Instead, we define a sequence of σ-algebras $\mathfrak{F}^1, \mathfrak{F}^2, \ldots, \mathfrak{F}^n$ as the sequence of σ-algebras on Ω that can be generated as we have access to the information through the first $1, 2, \ldots, n$ iterations, respectively. What does this mean? Consider two outcomes $\omega \neq \omega'$ for which $H^n(\omega) = H^n(\omega')$. If this is the case, then any event in $\mathfrak{F}^n$ that includes ω must also include ω'. If we say that a function is $\mathfrak{F}^n$-measurable, then this means that it must be defined in terms of the events in $\mathfrak{F}^n$, which is in turn equivalent to saying that we cannot be using any information from iterations $n+1, n+2, \ldots$.

We would say, then, that we have a standard probability space $(\Omega, \mathfrak{F}, \mathcal{P})$ where $\omega \in \Omega$ represents an elementary outcome, $\mathfrak{F}$ is the σ-algebra on $\mathfrak{F}$ and $\mathcal{P}$ is a probability measure on Ω. Since our information is revealed iteration by iteration, we would also then say that we have an increasing set of σ-algebras $\mathfrak{F}^1 \subseteq \mathfrak{F}^2 \subseteq \ldots \subseteq \mathfrak{F}^n$ (which is the same as saying that $\mathcal{F}^n$ is a filtration).

5.10.2 An older proof*

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

$$\max_x \mathbb{E}F(x,\omega) \tag{5.51}$$

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

$$g(x,\omega)^T \nabla F(x,\omega) \geq 0. \tag{5.52}$$

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

$$g(x,\omega) = \nabla F(x,\omega) \tag{5.53}$$

which clearly satisfies (5.52).

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

$$-M \leq g(x,\omega)^T \nabla^2 F(x) g(x,\omega) \leq M. \tag{5.54}$$

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

$$\bar{x}^n = \bar{x}^{n-1} + \alpha_{n-1} g(\bar{x}^{n-1}, \omega). \tag{5.55}$$

We first prove our result using the proof technique of Blum (1954*b*) that generalized the original stochastic approximation procedure proposed by Robbins & Monro (1951) to multidimensional problems. This approach does not depend on more advanced concepts such as martingales and, as a result, is accessible to a broader audience. This proof helps the reader understand the basis for the conditions $\sum_{n=0}^{\infty} \alpha_n = \infty$ and $\sum_{n=0}^{\infty} (\alpha_n)^2 < \infty$ that are required of all stochastic approximation algorithms.

We make the following (standard) assumptions on stepsizes

$$\alpha_n > 0 \text{ for all } n \geq 0, \tag{5.56}$$

$$\sum_{n=0}^{\infty} \alpha_n = \infty, \tag{5.57}$$

$$\sum_{n=0}^{\infty} (\alpha_n)^2 < \infty. \tag{5.58}$$

We want to show that under suitable assumptions, the sequence generated by (5.55) converges to an optimal solution. That is, we want to show that

$$\lim_{n\to\infty} x^n = x^* \ a.s. \tag{5.59}$$

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

$$F(x) = F(x^0) + \nabla F(x^0 + \eta(x - x^0))(x - x^0). \tag{5.60}$$

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

$$F(x) = F(x^0) + \nabla F(x^0)(x - x^0) + \frac{1}{2}(x - x^0)^T \nabla^2 F(x^0 + \eta(x - x^0))(x - x^0) \tag{5.61}$$

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

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

$$g^n = g(x^{n-1}, \omega). \tag{5.62}$$

This means that, by definition of our algorithm,

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

From our stochastic gradient algorithm (5.55), we may write

$$\begin{aligned} F(x^n, \omega) &= F(x^{n-1} + \alpha_{n-1} g^n, \omega) \\ &= F(x^{n-1}, \omega) + \nabla F(x^{n-1}, \omega)(\alpha_{n-1} g^n) \\ &\quad + \frac{1}{2}(\alpha_{n-1} g^n)^T \nabla^2 F(x^{n-1} + \eta \alpha_{n-1} g^n, \omega)(\alpha_{n-1} g^n). \end{aligned} \tag{5.63}$$

It is now time to use a *standard mathematician's trick*. We sum both sides of (5.63) to get

$$\begin{aligned} \sum_{n=1}^{N} F(x^n, \omega) &= \sum_{n=1}^{N} F(x^{n-1}, \omega) + \sum_{n=1}^{N} \nabla F(x^{n-1}, \omega)(\alpha_{n-1} g^n) + \\ &\quad \frac{1}{2} \sum_{n=1}^{N} (\alpha_{n-1} g^n)^T \nabla^2 F\left(x^{n-1} + \eta \alpha_{n-1} g^n, \omega\right)(\alpha_{n-1} g^n). \end{aligned} \tag{5.64}$$

Note that the terms $F(x^n), n = 2, 3, \ldots, N$ appear on both sides of (5.64). We can cancel these. We then use our lower bound on the quadratic term (5.54) to write

$$F(x^N, \omega) \geq F(x^0, \omega) + \sum_{n=1}^{N} \nabla F(x^{n-1}, \omega)(\alpha_{n-1} g^n) + \frac{1}{2} \sum_{n=1}^{N} (\alpha_{n-1})^2 (-M). \tag{5.65}$$

We now want to take the limit of both sides of (5.65) as $N \to \infty$. In doing so, we want to show that everything must be bounded. We know that $F(x^N)$ is bounded (*almost surely*) because we assumed that the original function was bounded. We next use the assumption (5.58) that the infinite sum of the squares of the stepsizes is also bounded to conclude that

the rightmost term in (5.65) is bounded. Finally, we use (5.52) to claim that all the terms in the remaining summation ($\sum_{n=1}^{N} \nabla F(x^{n-1})(\alpha_{n-1}g^n)$) are positive. That means that this term is also bounded (from both above and below).

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

$$\sum_{n=1}^{\infty} \alpha_{n-1} \nabla F(x^n, \omega) g^n < \infty \text{ for all } \omega \tag{5.66}$$

and (from (5.57))

$$\sum_{n=1}^{\infty} \alpha_{n-1} = \infty. \tag{5.67}$$

We can conclude that

$$\sum_{n=1}^{\infty} \nabla F(x^{n-1}, \omega) g^n < \infty. \tag{5.68}$$

Since all the terms in (5.68) are positive, they must go to zero. (Remember, everything here is true *almost surely*; after a while, it gets a little boring to keep saying *almost surely* every time. It is a little like reading Chinese fortune cookies and adding the automatic phrase "under the sheets" at the end of every fortune.)

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

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

Our presentation does, however, help to present several key ideas that are present in most proofs of this type. First, concepts of almost sure convergence are virtually standard. Second, it is common to set up equations such as (5.63) and then take a finite sum as in (5.64) using the alternating terms in the sum to cancel all but the first and last elements of the sequence of some function (in our case, $F(x^{n-1}, \omega)$). We then establish the boundedness of this expression as $N \to \infty$, which will require the assumption that $\sum_{n=1}^{\infty} (\alpha_{n-1})^2 < \infty$. Then, the assumption $\sum_{n=1}^{\infty} \alpha_{n-1} = \infty$ is used to show that if the remaining sum is bounded, then its terms must go to zero.

More modern proofs will use functions other than $F(x)$. Popular is the introduction of so-called Lyapunov functions, which are artificial functions that provide a measure of optimality. These functions are constructed for the purpose of the proof and play no role in the algorithm itself. For example, we might let $T^n = ||x^n - x^*||$ be the distance between our current solution x^n and the optimal solution. We will then try to show that T^n is suitably reduced to prove convergence. Since we do not know x^*, this is not a function we can actually measure, but it can be a useful device for proving that the algorithm actually converges.

It is important to realize that stochastic gradient algorithms of all forms do not guarantee an improvement in the objective function from one iteration to the next. First, a sample gradient g^n may represent an appropriate ascent vector for a sample of the function $F(x^n, \omega)$ but not for its expectation. In other words, randomness means that we may go in the wrong direction at any point in time. Second, our use of a nonoptimizing stepsize, such as $\alpha_{n-1} = 1/n$, means that even with a good ascent vector, we may step too far and actually end up with a lower value.

5.10.3 A more modern proof**

Since the original work by Robbins and Monro, more powerful proof techniques have evolved. Below we illustrate a basic martingale proof of convergence. The concepts are somewhat more advanced, but the proof is more elegant and requires milder conditions. A significant generalization is that we no longer require that our function be differentiable (which our first proof required). For large classes of resource allocation problems, this is a significant improvement.

First, just what is a martingale? Let $\omega_1, \omega_2, \ldots, \omega_t$ be a set of exogenous random outcomes, and let $h_t = H_t(\omega) = (\omega_1, \omega_2, \ldots, \omega_t)$ represent the history of the process up to time t. We also let $\mathfrak{F}_t$ be the σ-algebra on Ω generated by H_t. Further, let U_t be a function that depends on h_t (we would say that U_t is a $\mathfrak{F}_t$-measurable function), and bounded ($\mathbb{E}|U_t| < \infty$, $\forall t \geq 0$). This means that if we know h_t, then we know U_t deterministically (needless to say, if we only know h_t, then U_{t+1} is still a random variable). We further assume that our function satisfies

$$\mathbb{E}[U_{t+1}|\mathfrak{F}_t] \quad = \quad U_t.$$

If this is the case, then we say that U_t is a *martingale*. Alternatively, if

$$\mathbb{E}[U_{t+1}|\mathfrak{F}_t] \quad \leq \quad U_t \tag{5.69}$$

then we say that U_t is a *supermartingale*. If U_t is a supermartingale, then it has the property that it drifts downward, usually to some limit point U^*. What is important is that it only drifts downward in expectation. That is, it could easily be the case that $U_{t+1} > U_t$ for specific outcomes. This captures the behavior of stochastic approximation algorithms. Properly designed, they provide solutions that improve on average, but where from one iteration to another the results can actually get worse.

Finally, assume that $U_t \geq 0$. If this is the case, we have a sequence U_t that drifts downward but which cannot go below zero. Not surprisingly, we obtain the following key result:

Theorem 5.10.1. *Let U_t be a positive supermartingale. Then, U_t converges to a finite random variable U^* almost surely.*

Note that "almost surely" (which is typically abbreviated "a.s.") means "for all (or every) ω." Mathematicians like to recognize every possibility, so they will add "every ω that might happen with some probability," which means that we are allowing for the possibility that U_t might not converge for some sample realization ω that would never actually happen (that is, where $p(\omega) > 0$). This also means that it converges with probability one.

So what does this mean for us? We assume that we are still solving a problem of the form

$$\max_x \mathbb{E}F(x, \omega), \tag{5.70}$$

where we assume that $F(x,\omega)$ is continuous and concave (but we do not require differentiability). Let $\bar{x}^n$ be our estimate of x at iteration n (remember that $\bar{x}^n$ is a random variable). Instead of watching the evolution of a process of time, we are studying the behavior of an algorithm over iterations. Let $F^n = \mathbb{E}F(\bar{x}^n)$ be our objective function at iteration n and let F^* be the optimal value of the objective function. If we are maximizing, we know that $F^n \leq F^*$. If we let $U^n = F^* - F^n$, then we know that $U^n \geq 0$ (this assumes that we can find the true expectation, rather than some approximation of it). A stochastic algorithm will not guarantee that $F^n \geq F^{n-1}$, but if we have a good algorithm, then we may be able to show that U^n is a supermartingale, which at least tells us that in the limit, U^n will approach some limit $\bar{U}$. With additional work, we might be able to show that $\bar{U} = 0$, which means that we have found the optimal solution.

A common strategy is to define U^n as the distance between $\bar{x}^n$ and the optimal solution, which is to say

$$U^n = (\bar{x}^n - x^*)^2. \tag{5.71}$$

Of course, we do not know x^*, so we cannot actually compute U^n, but that is not really a problem for us (we are just trying to prove convergence). Note that we immediately get $U^n \geq 0$ (without an expectation). If we can show that U^n is a supermartingale, then we get the result that U^n converges to a random variable U^* (which means the algorithm converges). Showing that $U^* = 0$ means that our algorithm will (eventually) produce the optimal solution. We are going to study the convergence of our algorithm for maximizing $\mathbb{E}F(x,W)$ by studying the behavior of U^n.

We are solving this problem using a stochastic gradient algorithm

$$\bar{x}^n = \bar{x}^{n-1} + \alpha_{n-1}g^n, \tag{5.72}$$

where g^n is our stochastic gradient. If F is differentiable, we would write

$$g^n = \nabla_x F(\bar{x}^{n-1}, W^n).$$

But in general, F may be nondifferentiable, in which case we may have multiple gradients at a point $\bar{x}^{n-1}$ (for a single sample realization). In this case, we write

$$g^n \in \partial_x F(\bar{x}^{n-1}, W^n),$$

where $\partial_x F(\bar{x}^{n-1}, W^n)$ refers to the set of subgradients at $\bar{x}^{n-1}$. We assume our problem is unconstrained, so $\nabla_x F(\bar{x}^*, W^n) = 0$ if F is differentiable. If it is nondifferentiable, we would assume that $0 \in \partial_x F(\bar{x}^*, W^n)$.

Throughout our presentation, we assume that x (and hence g^n) is a scalar (exercise 6.17 provides an opportunity to redo this section using vector notation). In contrast with the previous section, we are now going to allow our stepsizes to be stochastic. For this reason, we need to slightly revise our original assumptions about stepsizes (equations (5.56) to (5.58)) by assuming

$$\alpha_n > 0 \ \textit{a.s.}, \tag{5.73}$$

$$\sum_{n=0}^{\infty} \alpha_n = \infty \ \textit{a.s.}, \tag{5.74}$$

$$\mathbb{E}\left[\sum_{n=0}^{\infty} (\alpha_n)^2\right] < \infty. \tag{5.75}$$

The requirement that α_n be nonnegative "almost surely" (a.s.) recognizes that α_n is a random variable. We can write $\alpha_n(\omega)$ as a sample realization of the stepsize (that is, this is the stepsize at iteration n if we are following sample path ω). When we require that $\alpha_n \geq 0$ "almost surely" we mean that $\alpha_n(\omega) \geq 0$ for all ω where the probability (more precisely, probability measure) of ω, $p(\omega)$, is greater than zero (said differently, this means that the probability that $\mathbb{P}[\alpha_n \geq 0] = 1$). The same reasoning applies to the sum of the stepsizes given in equation (5.74). As the proof unfolds, we will see the reason for needing the conditions (and why they are stated as they are).

We next need to assume some properties of the stochastic gradient g^n. Specifically, we need to assume the following:

Assumption 1 - $\mathbb{E}[g^{n+1}(\bar{x}^n - x^*)|\mathfrak{F}^n] \geq 0$,

Assumption 2 - $|g^n| \leq B_g$,

Assumption 3 - For any x where $|x - x^*| > \delta,\ \delta > 0$, there exists $\epsilon > 0$ such that $\mathbb{E}[g^{n+1}|\mathfrak{F}^n] > \epsilon$.

Assumption 1 assumes that on average, the gradient g^n points toward the optimal solution x^*. This is easy to prove for deterministic, differentiable functions. While this may be harder to establish for stochastic problems or problems where $F(x)$ is nondifferentiable, we do not have to assume that $F(x)$ is differentiable. Nor do we assume that a particular gradient g^{n+1} moves toward the optimal solution (for a particular sample realization, it is entirely possible that we are going to move away from the optimal solution). Assumption 2 assumes that the gradient is bounded. Assumption 3 requires that the expected gradient cannot vanish at a nonoptimal value of x. This assumption will be satisfied for any concave function.

To show that U^n is a supermartingale, we start with

$$\begin{aligned} U^{n+1} - U^n &= (\bar{x}^{n+1} - x^*)^2 - (\bar{x}^n - x^*)^2 \\ &= \left((\bar{x}^n - \alpha_n g^{n+1}) - x^*\right)^2 - (\bar{x}^n - x^*)^2 \\ &= \left((\bar{x}^n - x^*)^2 - 2\alpha_n g^{n+1}(\bar{x}^n - x^*) + (\alpha_n g^{n+1})^2\right) - (\bar{x}^n - x^*)^2 \\ &= (\alpha_n g^{n+1})^2 - 2\alpha_n g^{n+1}(\bar{x}^n - x^*). \end{aligned} \tag{5.76}$$

Taking conditional expectations on both sides gives

$$\mathbb{E}[U^{n+1}|\mathfrak{F}^n] - \mathbb{E}[U^n|\mathfrak{F}^n] = \mathbb{E}[(\alpha_n g^{n+1})^2|\mathfrak{F}^n] - 2\mathbb{E}[\alpha_n g^{n+1}(\bar{x}^n - x^*)|\mathfrak{F}^n]. \tag{5.77}$$

We note that

$$\begin{aligned} \mathbb{E}[\alpha_n g^{n+1}(\bar{x}^n - x^*)|\mathfrak{F}^n] &= \alpha_n \mathbb{E}[g^{n+1}(\bar{x}^n - x^*)|\mathfrak{F}^n] && (5.78) \\ &\geq 0. && (5.79) \end{aligned}$$

Equation (5.78) is subtle but important, as it explains a critical piece of notation in this book. Keep in mind that we may be using a stochastic stepsize formula, which means that α_n is a random variable. We assume that α_n is $\mathfrak{F}^n$-measurable, which means that we are not allowed to use information from iteration $n+1$ to compute it. This is why we use α_{n-1} in updating equations such as equation (5.13) instead of α_n. When we condition on $\mathfrak{F}^n$ in equation (5.78), α_n is deterministic, allowing us to take it outside the expectation. This allows us to write the conditional expectation of the product of α_n and g^{n+1} as the product

of the expectations. Equation (5.79) comes from Assumption 1 and the nonnegativity of the stepsizes.

Recognizing that $\mathbb{E}[U^n|\mathfrak{F}^n] = U^n$ (given $\mathfrak{F}^n$), we may rewrite (5.77) as

$$\begin{aligned}\mathbb{E}[U^{n+1}|\mathfrak{F}^n] &= U^n + \mathbb{E}[(\alpha_n g^{n+1})^2|\mathfrak{F}^n] - 2\mathbb{E}[\alpha_n g^{n+1}(\bar{x}^n - x^*)|\mathfrak{F}^n] \\ &\leq U^n + \mathbb{E}[(\alpha_n g^{n+1})^2|\mathfrak{F}^n]. \end{aligned} \tag{5.80}$$

Because of the positive term on the right-hand side of (5.80), we cannot directly get the result that U^n is a supermartingale. But hope is not lost. We appeal to a neat little trick that works as follows. Let

$$W^n = \mathbb{E}[U^n + \sum_{m=n}^{\infty}(\alpha_m g^{m+1})^2|\mathfrak{F}^n]. \tag{5.81}$$

We are going to show that W^n is a supermartingale. From its definition, we obtain

$$\begin{aligned} W^n &= \mathbb{E}[W^{n+1} + U^n - U^{n+1} + (\alpha_n g^{n+1})^2|\mathfrak{F}^n], \\ &= \mathbb{E}\left[W^{n+1}|\mathfrak{F}^n\right] + U^n - \mathbb{E}\left[U^{n+1}|\mathfrak{F}^n\right] + \mathbb{E}\left[(\alpha_n g^{n+1})^2|\mathfrak{F}^n\right] \end{aligned}$$

which is the same as

$$\mathbb{E}[W^{n+1}|\mathfrak{F}^n] = W^n - \underbrace{\left(U^n + \mathbb{E}\left[(\alpha_n g^{n+1})^2|\mathfrak{F}^n\right] - \mathbb{E}[U^{n+1}|\mathfrak{F}^n]\right)}_{I}.$$

We see from equation (5.80) that $I \geq 0$. Removing this term gives us the inequality

$$\mathbb{E}[W^{n+1}|\mathfrak{F}^n] \leq W^n. \tag{5.82}$$

This means that W^n is a supermartingale. It turns out that this is all we really need because $\lim_{n\rightarrow\infty} W^n = \lim_{n\rightarrow\infty} U^n$. This means that

$$\lim_{n\rightarrow\infty} U^n \rightarrow U^* \quad a.s. \tag{5.83}$$

Now that we have the basic convergence of our algorithm, we have to ask: but what is it converging to? For this result, we return to equation (5.76) and sum it over the values $n = 0$ up to some number N, giving us

$$\sum_{n=0}^{N}(U^{n+1} - U^n) = \sum_{n=0}^{N}(\alpha_n g^{n+1})^2 - 2\sum_{n=0}^{N}\alpha_n g^{n+1}(\bar{x}^n - x^*). \tag{5.84}$$

The left-hand side of (5.84) is an alternating sum (sometimes referred to as a telescoping sum), which means that every element cancels out except the first and the last, giving us

$$U^{N+1} - U^0 = \sum_{n=0}^{N}(\alpha_n g^{n+1})^2 - 2\sum_{n=0}^{N}\alpha_n g^{n+1}(\bar{x}^n - x^*).$$

Taking expectations of both sides gives

$$\mathbb{E}[U^{N+1} - U^0] = \mathbb{E}\left[\sum_{n=0}^{N}(\alpha_n g^{n+1})^2\right] - 2\mathbb{E}\left[\sum_{n=0}^{N}\alpha_n g^{n+1}(\bar{x}^n - x^*)\right]. \tag{5.85}$$

We want to take the limit of both sides as N goes to infinity. To do this, we have to appeal to the *Dominated Convergence Theorem* (DCT), which tells us that

$$\lim_{N\rightarrow\infty}\int_x f^n(x)dx = \int_x \left(\lim_{N\rightarrow\infty} f^n(x)\right)dx$$

if $|f^n(x)| \leq g(x)$ for some function $g(x)$ where

$$\int_x g(x)dx < \infty.$$

For our application, the integral represents the expectation (we would use a summation instead of the integral if x were discrete), which means that the DCT gives us the conditions needed to exchange the limit and the expectation. Above, we showed that $\mathbb{E}[U^{n+1}|\mathfrak{F}^n]$ is bounded (from (5.80) and the boundedness of U^0 and the gradient). This means that the right-hand side of (5.85) is also bounded for all n. The DCT then allows us to take the limit as N goes to infinity inside the expectations, giving us

$$U^* - U^0 \quad = \quad \mathbb{E}\left[\sum_{n=0}^{\infty}(\alpha_n g^{n+1})^2\right] - 2\mathbb{E}\left[\sum_{n=0}^{\infty}\alpha_n g^{n+1}(\bar{x}^n - x^*)\right].$$

We can rewrite the first term on the right-hand side as

$$\begin{aligned}
\mathbb{E}\left[\sum_{n=0}^{\infty}(\alpha_n g^{n+1})^2\right] &\leq \mathbb{E}\left[\sum_{n=0}^{\infty}(\alpha_n)^2(B)^2\right] &(5.86)\\
&= B^2\,\mathbb{E}\left[\sum_{n=0}^{\infty}(\alpha_n)^2\right] &(5.87)\\
&< \infty. &(5.88)
\end{aligned}$$

Equation (5.86) comes from Assumption 2 which requires that $|g^n|$ be bounded by B, which immediately gives us Equation (5.87). The requirement that $\mathbb{E}\sum_{n=0}^{\infty}(\alpha_n)^2 < \infty$ (equation (5.58)) gives us (5.88), which means that the first summation on the right-hand side of (5.85) is bounded. Since the left-hand side of (5.85) is bounded, we can conclude that the second term on the right-hand side of (5.85) is also bounded.

Now let

$$\begin{aligned}
\beta^n &= \mathbb{E}\left[g^{n+1}(\bar{x}^n - x^*)\right]\\
&= \mathbb{E}\left[\mathbb{E}\left[g^{n+1}(\bar{x}^n - x^*)|\mathfrak{F}^n\right]\right]\\
&\geq 0,
\end{aligned}$$

since $\mathbb{E}[g^{n+1}(\bar{x}^n - x^*)|\mathfrak{F}^n] \geq 0$ from Assumption 1. This means that

$$\sum_{n=0}^{\infty}\alpha_n\beta^n \quad < \quad \infty \quad \text{a.s.} \tag{5.89}$$

But, we have required that $\sum_{n=0}^{\infty}\alpha_n = \infty$ a.s. (equation (5.74)). Since $\alpha_n > 0$ and $\beta^n \geq 0$ (a.s.), we conclude that

$$\lim_{n\rightarrow\infty}\beta^n \quad \rightarrow \quad 0 \text{ a.s.} \tag{5.90}$$

If $\beta^n \rightarrow 0$, then $\mathbb{E}[g^{n+1}(\bar{x}^n - x^*)] \rightarrow 0$, which allows us to conclude that $\mathbb{E}[g^{n+1}(\bar{x}^n - x^*)|\mathfrak{F}^n] \rightarrow 0$ (the expectation of a nonnegative random variable cannot be zero unless the random variable is always zero). But what does this tell us about the behavior of $\bar{x}^n$? Knowing that $\beta^n \rightarrow 0$ does not necessarily imply that $g^{n+1} \rightarrow 0$ or $\bar{x}^n \rightarrow x^*$. There are three scenarios:

1) $\bar{x}^n \rightarrow x^*$ for all n, and of course all sample paths ω. If this were the case, we are done.

2) $\bar{x}^{n_k} \rightarrow x^*$ for a subsequence $n_1, n_2, \ldots, n_k, \ldots$. For example, it might be that the sequence $\bar{x}^1, \bar{x}^3, \bar{x}^5, \ldots \rightarrow x^*$, while $\mathbb{E}[g^2|\mathfrak{F}^1], \mathbb{E}[g^4|\mathfrak{F}^3], \ldots, \rightarrow 0$. This would mean that for the subsequence n_k, $U^{n_k} \rightarrow 0$. But we already know that $U^n \rightarrow U^*$ where U^* is the unique limit point, which means that $U^* = 0$. But if this is the case, then this is the limit point for every sequence of $\bar{x}^n$.

3) There is no subsequence $\bar{x}^{n_k}$ which has $\bar{x}^*$ as its limit point. This means that $\mathbb{E}[g^{n+1}|\mathfrak{F}^n] \rightarrow 0$. However, assumption 3 tells us that the expected gradient cannot vanish at a nonoptimal value of x. This means that this case cannot happen.

This completes the proof. □

5.11 BIBLIOGRAPHIC NOTES

Section 5.3 - The theoretical foundation for estimating value functions from Monte Carlo estimates has its roots in stochastic approximation theory, originated by Robbins & Monro (1951), with important early contributions made by Kiefer & Wolfowitz (1952), Blum (1954*a*) and Dvoretzky (1956). For thorough theoretical treatments of stochastic approximation theory, see Wasan (1969), Kushner & Clark (1978) and Kushner & Yin (1997). Very readable treatments of stochastic optimization can be found in Pflug (1996) and Spall (2003) (Spall's book is a modern classic on stochastic approximation methods). More modern treatments of stochastic gradient methods are given in Fu (2014) and Shapiro et al. (2014).

Section 5.4 - There are a number of ways to compute gradients, including numerical derivatives (when exact gradients are not available), gradient smoothing, mini-batches (averages of sampled gradients). Excellent modern treatments can be found in Michael Fu's edited volume Fu (2014), including Fu's chapter on stochastic gradient estimation [Chapter 5], and Chau and Fu's chapter on stochastic approximation methods and finite-difference methods [Chapter 6].

Section 5.4.4 - The simultaneous perturbation stochastic approximation (SPSA) method, which provides a practical strategy for estimating numerical gradients for higher-dimensional problems, is due to Spall (see Spall (2003)). Figure 5.3 was prepared by Saeed Ghadimi.

Section 5.6 - The formulation of a stochastic gradient algorithm as a sequential decision problem was first described in Powell (2019). However, mention should be made of the work of Harold Kushner (see Kushner & Yin (2003) for a summary) which viewed algorithms as dynamical systems. Our work viewing algorithms as *controlled* dynamical systems appears to be new, although this is hard to verify.

Section 5.10.2 - This proof is based on Blum (1954*b*), which generalized the original paper by Robbins & Monro (1951).

Section 5.10.3 - The proof in section 5.10.3 uses standard techniques drawn from several sources, notably Wasan (1969), Chong (1991), Kushner & Yin (1997) and, for this author, Powell & Cheung (2000).

EXERCISES

Review questions

5.1 Write out a basic stochastic gradient algorithm for iteration n, and explain why W^{n+1} is indexed by $n+1$ instead of n. Write out the stochastic gradient for the newsvendor problem.

5.2 There are potentially three forms of uncertainty that arise in the use of stochastic gradient algorithms. Give the notation for each and explain with an example (you may use an example from the chapter).

5.3 A gradient for a continuous, deterministic function points in the direction of steepest ascent. Is this true for stochastic gradients? Illustrate this for the problem of estimating the mean of a random variable.

5.4 Consider the newsvendor problem

$$F(x,W) = 10\max\{x,W\} - 8x.$$

For $x = 9$ and $W = 10$ compute the numerical derivative $\nabla_x F(x,W)$ using the increment $\delta = 1$. What if you use $\delta = 4$?

Modeling questions

5.5 Consider a function $F(x,W)$ that depends on a decision $x = x^n$ after which we observe a random outcome W^{n+1}. Assume that we can compute the gradient $\nabla_x F(x^n, W^{n+1})$. We would like to optimize this problem using a standard stochastic gradient algorithm:

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

Our goal is to find the best answer we can after N iterations.

a) Assume that we are using a stepsize policy of

$$\alpha_n = \frac{\theta}{\theta + n - 1}.$$

Model the problem of finding the best stepsize policy as a stochastic optimization problem. Give the state variable(s), the decision variable, the exogenous information, the transition function, and the objective function. Please use precise notation.

b) How does your model change if you switch to Kesten's stepsize rule which uses

$$\alpha_n = \frac{\theta}{\theta + N^n - 1},$$

where N^n is the number of times that the gradient has changed signs, which is computed using

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

5.6 A customer is required by her phone company to pay for a minimum number of minutes per month for her cell phone. She pays 12 cents per minute of guaranteed minutes, and 30 cents per minute that she goes over her minimum. Let x be the number of minutes she commits to each month, and let M be the random variable representing the number of minutes she uses each month, where M is normally distributed with mean 300 minutes and a standard deviation of 60 minutes.

(a) Write down the objective function in the form $\min_x \mathbb{E} f(x, M)$.

(b) Derive the stochastic gradient for this function.

(c) Let $x^0 = 0$ and choose as a stepsize $\alpha_{n-1} = 10/n$. Use 100 iterations to determine the optimum number of minutes the customer should commit to each month.

5.7 An oil company covers the annual demand for oil using a combination of futures and oil purchased on the spot market. Orders are placed at the end of year $t-1$ for futures that can be exercised to cover demands in year t. If too little oil is purchased this way, the company can cover the remaining demand using the spot market. If too much oil is purchased with futures, then the excess is sold at 70 percent of the spot market price (it is not held to the following year – oil is too valuable and too expensive to store).

To write down the problem, model the exogenous information using

$$\begin{aligned} \hat{D}_t &= \text{demand for oil during year } t, \\ \hat{p}^s_t &= \text{spot price paid for oil purchased in year } t, \\ \hat{p}^f_{t,t+1} &= \text{futures price paid in year } t \text{ for oil to be used in year } t+1. \end{aligned}$$

The demand (in millions of barrels) is normally distributed with mean 600 and standard deviation of 50. The decision variables are given by

$$\begin{aligned} \bar{\theta}^f_{t,t+1} &= \text{number of futures to be purchased at the end of year } t \text{ to be used in year } t+1, \\ \bar{\theta}^s_t &= \text{spot purchases made in year } t. \end{aligned}$$

(a) Set up the objective function to minimize the expected total amount paid for oil to cover demand in a year $t+1$ as a function of $\bar{\theta}^f_t$. List the variables in your expression that are not known when you have to make a decision at time t.

(b) Give an expression for the stochastic gradient of your objective function. That is, what is the derivative of your function for a particular sample realization of demands and prices (in year $t+1$)?

(c) Generate 100 years of random spot and futures prices as follows:

$$\begin{aligned} \hat{p}^f_t &= 0.80 + 0.10 U^f_t, \\ \hat{p}^s_{t,t+1} &= \hat{p}^f_t + 0.20 + 0.10 U^s_t, \end{aligned}$$

where U_t^f and U_t^s are random variables uniformly distributed between 0 and 1. Run 100 iterations of a stochastic gradient algorithm to determine the number of futures to be purchased at the end of each year. Use $\bar{\theta}_0^f = 30$ as your initial order quantity, and use as your stepsize $\alpha_t = 20/t$. Compare your solution after 100 years to your solution after 10 years. Do you think you have a good solution after 10 years of iterating?

Computational exercises

5.8 We want to compute a numerical derivative of the newsvendor problem

$$F(x, W) = 10 \min\{x, W\} - 8x.$$

Assume that we have generated a random sample of $W = 12$, and that we want to generate a numerical derivative to estimate the gradient $\nabla_x F(x, W)$ for $x = 8$ and $W = 12$.

a) Compute a right-biased numerical derivative using $\delta = 1.0$. Show how to perform the computation and given the resulting estimate.

b) Compute a balanced numerical derivative centered on $x = 8$, but using estimates perturbed by $+\delta$ and $-\delta$.

c) Write the software using any environment to optimize $F(x, W)$ using numerical derivatives, assuming $W \in Uniform[5, 20]$. Carefully specify any assumptions you make. Run your algorithm for 20 iterations.

5.9 Below is a form of two-dimensional newsvendor problem, where we allocate two types of resource, x_1 and x_2, to meet a common demand W:

$$F(x_1, x_2, W) = 10 \min\{x_1, W\} + 14 \min\{x_2, (\max\{0, W - x_1\})\} - 8x_1 - 10x_2.$$

We are going to pretend that our vector x might have a dozen or more dimensions, but use this two-dimensional version to perform a detailed numerical example of the SPSA method for estimating gradients.

a) Use the SPSA algorithm to compute an estimate of the gradient $\nabla_x F(x_1, x_2, W)$ using two-function evaluations around the point $x_1 = 8$, $x_2 = 10$. Show all the detailed calculations and the resulting gradient. Show how you handled the sampling of W.

b) Write the software using any environment to optimize $F(x_1, x_2, W)$ using the SPSA algorithm, assuming $W \in Uniform[5, 20]$. Carefully specify any assumptions you make. Run your algorithm for 20 iterations.

Theory questions

5.10 The proof in section 5.10.3 was performed assuming that x is a scalar. Repeat the proof assuming that x is a vector. You will need to make adjustments such as replacing Assumption 2 with $\|g^n\| < B$. You will also need to use the triangle inequality which states that $\|a + b\| \leq \|a\| + \|b\|$.

Problem solving questions

5.11 Write out the stochastic gradient for the nested newsvendor problem given in equation (5.9).

5.12 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 revert back to the employer. Assume that you are in a 40 percent tax bracket (sounds nice, and the arithmetic is a bit easier). Let M be the random variable representing total medical expenses in the upcoming year, and let $F(x) = Prob[M \leq x]$ be the cumulative distribution function of the random variable M.

a) Write out the objective function 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 $g(x)$ is the gradient of your objective function if you allocate x to the FSA, use the property that $g(x^*) = 0$ to derive (you must show the derivation) the critical ratio that gives the relationship between x^* and the cumulative distribution function $F(x)$.

c) If you are in a 35 percent tax bracket, what percentage of the time should you have funds left over at the end of the year?

5.13 We are going to solve a classic stochastic optimization problem known as the newsvendor problem. Assume we have to order x assets after which we try to satisfy a random demand D for these assets, where D is randomly distributed between 100 and 200. If $x > D$, we have ordered too much and we pay $5(x - D)$. If $x < D$, we have an underage, and we have to pay $20(D - x)$.

(a) Write down the objective function in the form $\min_x \mathbb{E}f(x, D)$.

(b) Derive the stochastic gradient for this function.

(c) Find the optimal solution analytically [Hint: take the expectation of the stochastic gradient, set it equal to zero and solve for the quantity $\mathbb{P}(D \leq x^*)$. From this, find x^*.]

(d) Since the gradient is in units of dollars while x is in units of the quantity of the asset being ordered, we encounter a scaling problem. Choose as a stepsize $\alpha_{n-1} = \alpha_0/n$ where α_0 is a parameter that has to be chosen. Use $x^0 = 100$ as an initial solution. Plot x^n for 1000 iterations for $\alpha_0 = 1, 5, 10, 20$. Which value of α_0 seems to produce the best behavior?

(e) Repeat the algorithm (1000 iterations) 10 times. Let $\omega = (1, \ldots, 10)$ represent the 10 sample paths for the algorithm, and let $x^n(\omega)$ be the solution at iteration n for sample path ω. Let *Var*(x^n) be the variance of the random variable x^n where

$$\overline{V}(x^n) = \frac{1}{10}\sum_{\omega=1}^{10}(x^n(\omega) - x^*)^2$$

Plot the standard deviation as a function of n for $1 \leq n \leq 1000$.

5.14 Following the methods of section 5.5.1, compute the gradient $\nabla_\theta F(x^n, \hat{f}^{n+1}|\theta)$ for the network depicted in figure 5.7.

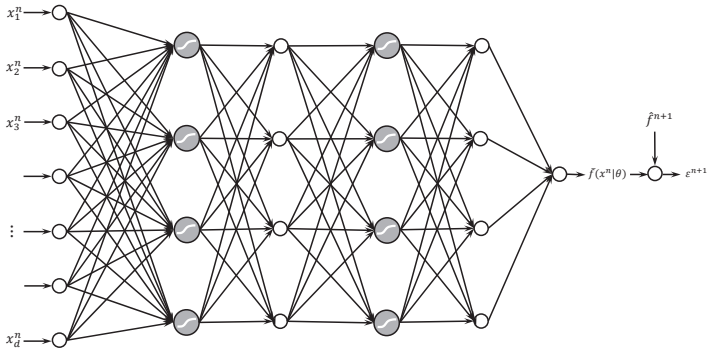


Figure 5.7 A four-layer neural network for exercise 5.14.

Sequential decision analytics and modeling

These exercises are drawn from the online book *Sequential Decision Analytics and Modeling* available at `http://tinyurl.com/sdaexamplesprint`.

5.15 Read chapter 3, sections 3.1-3.4 on the adaptive market planning problem. The presentation provides a classical derivation of the optimal order quantity for a newsvendor problem, but then presents a version of the problem where the objective is the cumulative reward.

a) When we are optimizing a single-period newsvendor problem, we are looking for the best x. What are we searching for when we are optimizing a multiperiod newsvendor problem where we are maximizing cumulative rewards (as is done in the book)?

b) Compute the gradient of the cumulative reward objective function with respect to the stepsize parameter θ^{step} in the stepsize rule.

c) Describe a stochastic gradient algorithm for optimizing θ^{step} when using a cumulative reward objective.

Diary problem

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

5.16 Your diary problem may have a decision x which is continuous (a quantity, a temperature, a price, a dosage). If not, you may wish to use a parameterized procedure $X^\pi(S_t|\theta)$ for determining x_t when you are in state S_t, where you are going to face the need to tune θ. At this point in the book, you have not developed the background to model and solve your problem, but discuss any continuous decisions (or tunable parameters) that you think might arise when modeling and solving your problem. Then, describe any new information that you might learn after you fix your continuous variable (this would play the role of W in the canonical problem for this chapter).

Bibliography

Blum, J. (1954*a*), 'Multidimensional stochastic approximation methods'', *Annals of Mathematical Statistics* **25**, 737–74462.

Blum, J. R. (1954*b*), 'Approximation Methods which Converge with Probability One', *Annals of Mathematical Statistics* **25**, 382–386.

Chong, E. K. P. (1991), On-Line Stochastic Optimization of Queueing Systems.

Dvoretzky, A. (1956), On Stochastic Approximation, *in* J. Neyman, ed., 'Proceedings 3rd Berkeley Symposium on Mathematical Statistics and Probability', University of California Press, pp. 39–55.

Ermoliev, Y. (1988), Stochastic Quasigradient Methods, *in* Y. Ermoliev & R. Wets, eds, 'Numerical Techniques for Stochastic Optimization', Springer-Verlag, Berlin.

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

Gaivoronski, A. (1988), Stochastic quasigradient methods and their implementation, *in* Y. Ermoliev & R. Wets, eds, 'Numerical Techniques for Stochastic Optimization', Springer-Verlag, Berlin.

Kiefer, J. & Wolfowitz, J. (1952), 'Stochastic estimation of the maximum of a regression function', *Annals of Mathematical Statistics* **23**, 462–466.

Kushner, H. J. & Clark, S. (1978), *Stochastic Approximation Methods for Constrained and Unconstrained Systems*, Springer-Verlag, New York.

Kushner, H. J. & Yin, G. G. (1997), *Stochastic Approximation Algorithms and Applications*, Springer-Verlag, New York.

Kushner, H. J. & Yin, G. G. (2003), *Stochastic Approximation and Recursive Algorithms and Applications*, Springer, New York.

Pflug, G. (1996), *Optimization of Stochastic Models: The Interface Between Simulation and Optimization*, Kluwer International Series in Engineering and Computer Science: Discrete Event Dynamic Systems, Kluwer Academic Publishers, Boston.

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

Powell, W. B. & Cheung, R. K.-M. (2000), 'SHAPE: A Stochastic Hybrid Approximation Procedure for Two-Stage Stochastic Programs', *Operations Research* **48**, 73–79.

Robbins, H. & Monro, S. (1951), 'A stochastic approximation method', *The Annals of Mathematical Statistics* **22**(3), 400–407.

Ruszczyński, A. (1980), 'Feasible Direction Methods for Stochastic Programming Problems', *Math. Programming* **19**, 220–229.

Ruszczyński, A. (1987), 'A Linearization Method for Nonsmooth Stochastic Programming Problems', *Mathematics of Operations Research* **12**, 32–49.

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

Spall, J. C. (2003), *Introduction to Stochastic Search and Optimization: Estimation, simulation and control*, John Wiley & Sons, Hoboken, NJ.

Wasan, M. T. (1969), *Stochastic approximation*, Cambridge University Press, Cambridge.