REINFORCEMENT LEARNING AND STOCHASTIC OPTIMIZATION

A unified framework for sequential decisions

Warren B. Powell

August 22, 2021

WILEY-INTERSCIENCE

A JOHN WILEY & SONS, INC., PUBLICATION

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

Library of Congress Cataloging-in-Publication Data:

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

10 9 8 7 6 5 4 3 2 1

CHAPTER 6

STEPSIZE POLICIES

There is a wide range of adaptive learning problems that depend on an iteration of the form we first saw in chapter 5 that looks like

$$x^{n+1} = x^n + \alpha_n \nabla_x F(x^n, W^{n+1}). \tag{6.1}$$

The stochastic gradient $\nabla_x F(x^n, W^{n+1})$ tells us what direction to go in, but we need the stepsize α_n to tell us how far we should move.

There are two important settings where this formula is used. The first is where we are maximizing some metric such as contributions, utility or performance. In these settings, the units of $\nabla_x F(x^n, W^{n+1})$ and the decision variable x are different, so the stepsize has to perform the scaling so that the size of $\alpha_n \nabla_x F(x^n, W^{n+1})$ is not too large or too small relative to x^n.

A second and very important setting arises in what is known as *supervised learning*. In this context, we are trying to estimate some function $f(x|\theta)$ using observations $y = f(x|\theta) + \varepsilon$. In this context, $f(x|\theta)$ and y have the same scale. We encounter these problems in three settings:

- Approximating the function $\mathbb{E}F(x, W)$ to create an estimate $\bar{F}(x)$ that can be optimized.

- Approximating the value $V_t(S_t)$ of being in a state S_t and then following some policy (we encounter this problem starting in chapters 16 and 17 when we introduce approximate dynamic programming).

- Creating a parameterized policy $X^\pi(S|\theta)$ to fit observed decisions. Here, we assume we have access to some method of creating a decision x and then we use this to create a parameterized policy $X^\pi(S|\theta)$. One source of decisions x is watching human behavior (for example, the choices made by a physician), but we could use any of our four classes of policies.

In chapter 5, we saw a range of methods for approximating functions. Imagine that we face the simplest problem of estimating the mean of a random variable W, which we can show (see exercise 6.21) solves the following stochastic optimization problem

$$\min_x \mathbb{E}\frac{1}{2}(x-W)^2. \tag{6.2}$$

Let $F(x,W) = \frac{1}{2}(x-W)^2$. The stochastic gradient of $F(x,W)$ with respect to x is

$$\nabla_x F(x,W) = (x-W).$$

We can optimize (6.2) using a stochastic gradient algorithm which we would write (remember that we are minimizing):

$$\begin{aligned} x^{n+1} &= x^n - \alpha_n \nabla F(x^n, W^{n+1}) && (6.3)\\ &= x^n - \alpha_n (x^n - W^{n+1}) && (6.4)\\ &= (1-\alpha_n)x^n + \alpha_n W^{n+1}. && (6.5) \end{aligned}$$

Equation (6.5) will be familiar to many readers as exponential smoothing (also known as a *linear filter* in signal processing). The important observation is that in this setting, the stepsize α_n needs to be between 0 and 1 since x and W are the same scale.

One of the challenges in Monte Carlo methods is finding the stepsize α_n. We refer to a method for choosing a stepsize as a *stepsize policy*, although popular terms include stepsize rule or learning rate schedules. To illustrate, we begin by rewriting the optimization problem (6.2) in terms of finding the estimate $\bar{\mu}$ of μ which is the true mean of the random variable W which we write as

$$\min_{\bar{\mu}} \mathbb{E}\frac{1}{2}(\bar{\mu}-W)^2. \tag{6.6}$$

This switch in notation will allow us to later make decisions about how to estimate $\mu_x = \mathbb{E}_W F(x,W)$ where we observe $\hat{F} = F(x,W)$. For now, we just want to focus on a simple estimation problem.

Our stochastic gradient updating equation (6.4) becomes

$$\bar{\mu}^{n+1} = \bar{\mu}^n - \alpha_n(\bar{\mu}^n - W^{n+1}). \tag{6.7}$$

With a properly designed stepsize rule (such as $\alpha_n = 1/n$), we can guarantee that

$$\lim_{n\to\infty} \bar{\mu}^n \to \mu,$$

but our interest is doing the best we can within a budget of N iterations which means we are trying to solve

$$\max_\pi \mathbb{E}_{S^0}\mathbb{E}_{W^1,\ldots,W^N|S^0}\mathbb{E}_{\widehat{W}|S^0} F(x^{\pi,N}, \widehat{W}), \tag{6.8}$$

where π refers to our stepsize rule, covering both the type of rule and any tunable parameters. We note that in this chapter, we do not care if we are solving the final-reward objective (6.8), or the cumulative-reward objective given by

$$\max_{\pi} \mathbb{E}_{S^0}\mathbb{E}_{W^1,\ldots,W^N|S^0} \sum_{n=0}^{N} F(x^n, W^{n+1}), \tag{6.9}$$

where $x^n = X^\pi(S^n)$. Our goal is to search for the best stepsize formula (and the best within a class) regardless of the objective.

There are two issues when designing a good stepsize rule. The first is the question of whether the stepsize produces some theoretical guarantee, such as asymptotic convergence or a finite time bound. While this is primarily of theoretical interest, these conditions do provide important guidelines to follow to produce good behavior. The second issue is whether the rule produces good empirical performance.

We divide our presentation of stepsize rules into three classes:

Deterministic policies - These are stepsize policies that are deterministic functions of the iteration counter n. This means that we know before we even start running our algorithm what the stepsize α_n will be.

Adaptive policies - Also known as stochastic stepsize policies, these are policies where the stepsize at iteration n depends on the statistics computed from the trajectory of the algorithm. These are also known as *stochastic stepsize rules*.

Optimal policies - Our deterministic and adaptive stepsize policies may have provable guarantees of asymptotic convergence, but were not derived using a formal optimization model. A byproduct of this heritage is that they require tuning one or more parameters. Optimal policies are derived from a formal model which is typically a simplified problem. These policies tend to be more complex, but eliminate or at least minimize the need for parameter tuning.

The deterministic and stochastic rules presented in section 6.1 and section 6.2 are, for the most part, designed to achieve good rates of convergence, but are not supported by any theory that they will produce the best rate of convergence. Some of these stepsize rules are, however, supported by asymptotic proofs of convergence and/or regret bounds.

In section 6.3 we provide a theory for choosing stepsizes that produce the fastest possible rate of convergence when estimating value functions based on policy evaluation. Finally, section 6.4 presents an optimal stepsize rule designed specifically for approximate value iteration.

6.1 DETERMINISTIC STEPSIZE POLICIES

Deterministic stepsize policies are the simplest to implement. Properly tuned, they can provide very good results. We begin by presenting some basic properties that a stepsize rule has to satisfy to ensure asymptotic convergence. While we are going to be exclusively interested in performance in finite time, these rules provide guidelines that are useful regardless of the experimental budget. After this, we present a variety of recipes for deterministic stepsize policies.

6.1.1 Properties for convergence

The theory for proving convergence of stochastic gradient algorithms was first developed in the early 1950's and has matured considerably since then (see section 5.10). However, all the proofs require three basic conditions

$$\alpha_n > 0, \quad n = 0, 1, \ldots, \tag{6.10}$$

$$\sum_{n=0}^{\infty} \alpha_n = \infty, \tag{6.11}$$

$$\sum_{n=0}^{\infty} (\alpha_n)^2 < \infty. \tag{6.12}$$

Equation (6.10) requires that the stepsizes be strictly positive (we cannot allow stepsizes equal to zero). The most important requirement is (6.11), which states that the infinite sum of stepsizes must be infinite. If this condition did not hold, the algorithm might stall prematurely. Finally, condition (6.12) requires that the infinite sum of the squares of the stepsizes be finite. This condition, in effect, requires that the stepsize sequence converge "reasonably quickly."

An intuitive justification for condition (6.12) is that it guarantees that the *variance* of our estimate of the optimal solution goes to zero in the limit. Sections 5.10.2 and 5.10.3 illustrate two proof techniques that both lead to these requirements on the stepsize. However, it is possible under certain conditions to replace equation (6.12) with the weaker requirement that $\lim_{n\to\infty} \alpha_n = 0$.

Condition (6.11) effectively requires that the stepsizes decline according to an arithmetic sequence such as

$$\alpha_{n-1} = \frac{1}{n}. \tag{6.13}$$

This rule has an interesting property. Exercise 6.21 asks you to show that a stepsize of $1/n$ produces an estimate $\bar{\mu}^n$ that is simply an average of all previous observations, which is to say

$$\bar{\mu}^n = \frac{1}{n} \sum_{m=1}^{n} W^m. \tag{6.14}$$

Of course, we have a nice name for equation (6.14): it is called a sample average. And we are all aware that in general (some modest technical conditions are required) as $n \to \infty$, $\bar{\mu}^n$ will converge (in some sense) to the mean of our random variable W.

The issue of the rate at which the stepsizes decrease is of considerable practical importance. Consider, for example, the stepsize sequence

$$\alpha_n = .5\alpha_{n-1},$$

which is a geometrically decreasing progression. This stepsize formula violates condition (6.11). More intuitively, the problem is that the stepsizes would decrease so quickly that the algorithm would stall prematurely. Even if the gradient pointed in the right direction at each iteration, we likely would never reach the optimum.

There are settings where the "$1/n$" stepsize formula is the best that we can do (as in finding the mean of a random variable), while in other situations it can perform extremely

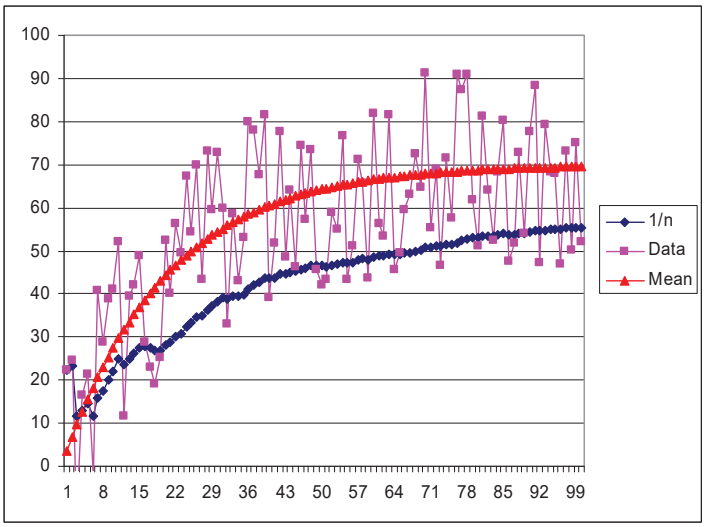


Figure 6.1 Illustration of poor convergence of the $1/n$ stepsize rule in the presence of transient data.

poorly because it can decline to zero too quickly. One situation where it works poorly arises when we are estimating a function that is changing over time (or iterations). For example, the algorithmic strategy called Q-learning (which we first saw in section 2.1.6) involves two steps:

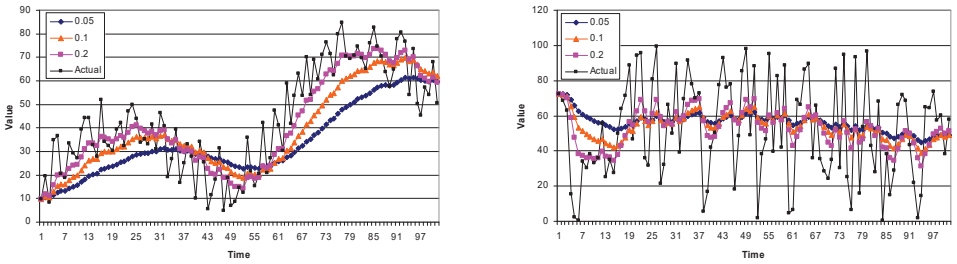
$$\begin{aligned}\hat{q}^n(s^n,a^n) &= r(s^n,a^n)+\gamma\max_{a'}\bar{Q}^{n-1}(s',a'),\\ \bar{Q}^n(s^n,a^n) &= (1-\alpha_{n-1})\bar{Q}^{n-1}(s^n,a^n)+\alpha_{n-1}\hat{q}^n(s^n,a^n).\end{aligned}$$

Here, we create a sampled observation $\hat{q}^n(s^n,a^n)$ of being in a state s^n and taking an action a^n, which we compute using the one period reward $r(s^n,a^n)$ plus an estimate of the downstream value, computed by sampling a downstream state s' given the current state s^n and action a^n, and then choosing the best action a' based on our current estimate of the value of different state-action pairs $\bar{Q}^{n-1}(s',a')$. We then smooth $\hat{q}^n(s^n,a^n)$ using our stepsize α_{n-1} to obtain updated estimates $\bar{Q}^n(s^n,a^n)$ of the value of the state-action pair s^n and a^n.

Figure 6.1 illustrates the behavior of using $1/n$ in this setting, which shows that we are significantly underestimating the values. Below, we fix this by generalizing $1/n$ using a tunable parameter. Later, we are going to present stepsize formulas that help to mitigate this behavior.

6.1.2 A collection of deterministic policies

The remainder of this section presents a series of deterministic stepsize formulas designed to overcome this problem. These rules are the simplest to implement and are typically a good starting point when implementing adaptive learning algorithms.



(a): Low-noise (b): High-noise

Figure 6.2 Illustration of the effects of smoothing using constant stepsizes. Case (a) represents a low-noise dataset, with an underlying nonstationary structure; case (b) is a high-noise dataset from a stationary process.

Constant stepsizes

A constant stepsize rule is simply

$$\alpha_{n-1} = \begin{cases} 1 & \text{if } n = 1, \\ \bar{\alpha} & \text{otherwise,} \end{cases}$$

where $\bar{\alpha}$ is a stepsize that we have chosen. It is common to start with a stepsize of 1 so that we do not need an initial value $\bar{\mu}^0$ for our statistic.

Constant stepsizes are popular when we are estimating not one but many parameters (for large scale applications, these can easily number in the thousands or millions). In these cases, no single rule is going to be right for all of the parameters and there is enough noise that any reasonable stepsize rule will work well.

Constant stepsizes are easy to code (no memory requirements) and, in particular, easy to tune (there is only one parameter). Perhaps the biggest point in their favor is that we simply may not know the rate of convergence, which means that we run the risk with a declining stepsize rule of allowing the stepsize to decline too quickly, producing a behavior we refer to as "apparent convergence."

In dynamic programming, we are typically trying to estimate the value of being in a state using observations that are not only random, but which are also changing systematically as we try to find the best policy. As a general rule, as the noise in the observations of the values increases, the best stepsize decreases. But if the values are increasing rapidly, we want a larger stepsize.

Choosing the best stepsize requires striking a balance between stabilizing the noise and responding to the changing mean. Figure 6.2 illustrates observations that are coming from a process with relatively low noise but where the mean is changing quickly (6.2a), and observations that are very noisy but where the mean is not changing at all (6.2b). For the first, the ideal stepsize is relatively large, while for the second, the best stepsize is quite small.

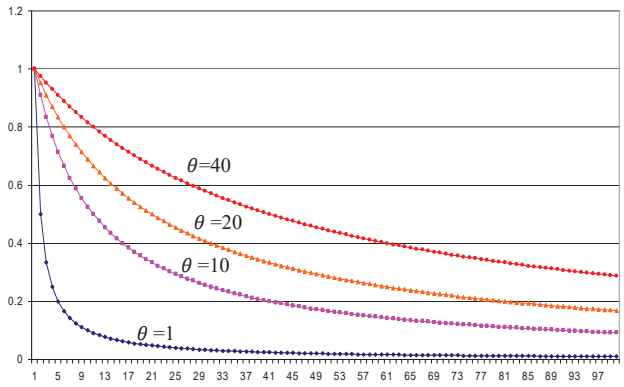


Figure 6.3 Stepsizes for $a/(a+n)$ while varying a.

Generalized harmonic stepsizes

A generalization of the $1/n$ rule is the generalized harmonic sequence given by

$$\alpha_{n-1} = \frac{\theta}{\theta + n - 1}. \tag{6.15}$$

This rule satisfies the conditions for convergence, but produces larger stepsizes for $\theta > 1$ than the $1/n$ rule. Increasing θ slows the rate at which the stepsize drops to zero, as illustrated in figure 6.3. In practice, it seems that despite theoretical convergence proofs to the contrary, the stepsize $1/n$ can decrease to zero far too quickly, resulting in "apparent convergence" when in fact the solution is far from the best that can be obtained.

Polynomial learning rates

An extension of the basic harmonic sequence is the stepsize

$$\alpha_{n-1} = \frac{1}{(n)^\beta}, \tag{6.16}$$

where $\beta \in (\frac{1}{2}, 1]$. Smaller values of β slow the rate at which the stepsizes decline, which improves the responsiveness in the presence of initial transient conditions. The best value of β depends on the degree to which the initial data is transient, and as such is a parameter that needs to be tuned.

McClain's formula

McClain's formula is an elegant way of obtaining $1/n$ behavior initially but approaching a specified constant in the limit. The formula is given by

$$\alpha_n = \frac{\alpha_{n-1}}{1 + \alpha_{n-1} - \bar{\alpha}}. \tag{6.17}$$

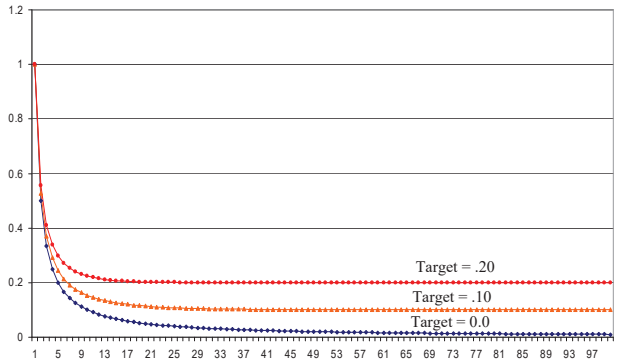


Figure 6.4 The McClain stepsize rule with varying targets.

where $\bar{\alpha}$ is a specified parameter. Note that steps generated by this model satisfy the following properties

$$\alpha_n > \alpha_{n+1} > \bar{\alpha} \quad \text{if} \quad \alpha > \bar{\alpha},$$
$$\alpha_n < \alpha_{n+1} < \bar{\alpha} \quad \text{if} \quad \alpha < \bar{\alpha}.$$

McClain's rule, illustrated in figure 6.4, combines the features of the "$1/n$" rule which is ideal for stationary data, and constant stepsizes for nonstationary data. If we set $\bar{\alpha} = 0$, then it is easy to verify that McClain's rule produces $\alpha_{n-1} = 1/n$. In the limit, $\alpha_n \to \bar{\alpha}$. The value of the rule is that the $1/n$ averaging generally works quite well in the very first iterations (this is a major weakness of constant stepsize rules), but avoids going to zero. The rule can be effective when you are not sure how many iterations are required to start converging, and it can also work well in nonstationary environments.

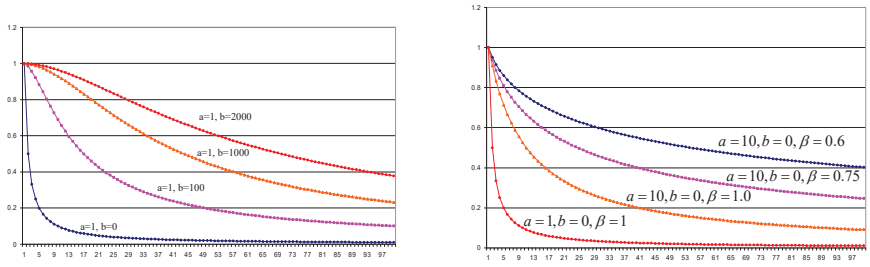
Search-then-converge learning policy

The search-then-converge (STC) stepsize rule is a variation on the harmonic stepsize rule that produces delayed learning. The rule can be written as

$$\alpha_{n-1} = \alpha_0 \frac{\left(\frac{b}{n} + a\right)}{\left(\frac{b}{n} + a + n^{\beta}\right)}. \tag{6.18}$$

If $\beta = 1$, then this formula is similar to the STC policy. In addition, if $b = 0$, then it is the same as the harmonic stepsize policy $\theta/(\theta + n)$. The addition of the term b/n to the numerator and the denominator can be viewed as a kind of harmonic stepsize policy where a is very large but declines with n. The effect of the b/n term, then, is to keep the stepsize larger for a longer period of time, as illustrated in figure 6.5(a). This can help algorithms that have to go through an extended learning phase when the values being estimated are relatively unstable. The relative magnitude of b depends on the number of iterations which are expected to be run, which can range from several dozen to several million.

This class of stepsize rules is termed "search-then-converge" because they provide for a period of high stepsizes (while searching is taking place) after which the stepsize declines (to achieve convergence). The degree of delayed learning is controlled by the parameter b, which can be viewed as playing the same role as the parameter a but which declines



6.5(a) 6.5(b)

Figure 6.5 The search-then-converge rule while (a) varying b, and (b) varying β.

as the algorithm progresses. The rule is designed for approximate dynamic programming methods applied to the setting of playing games with a delayed reward (there is no reward until you win or lose the game).

The exponent β in the denominator has the effect of increasing the stepsize in later iterations (see figure 6.5(b)). With this parameter, it is possible to accelerate the reduction of the stepsize in the early iterations (by using a smaller a) but then slow the descent in later iterations (to sustain the learning process). This may be useful for problems where there is an extended transient phase requiring a larger stepsize for a larger number of iterations.

6.2 ADAPTIVE STEPSIZE POLICIES

There is considerable appeal to the idea that the stepsize should depend on the actual trajectory of the algorithm. For example, if we are consistently observing that our estimate $\bar{\mu}^{n-1}$ is smaller (or larger) than the observations W^n, then it suggests that we are trending upward (or downward). When this happens, we typically would like to use a larger stepsize to increase the speed at which we reach a good estimate. When the stepsizes depend on the observations W^n, then we say that we are using a *adaptive stepsize*. This means, however, that we have to recognize that it is a random variable (some refer to these as stochastic stepsize rules).

In this section, we first review the case for adaptive stepsizes, then present the revised theoretical conditions for convergence, and finally outline a series of heuristic recipes that have been suggested in the literature. After this, we present some stepsize rules that are optimal until special conditions.

6.2.1 The case for adaptive stepsizes

Assume that our estimates are consistently under or consistently over the actual observations. This can easily happen during early iterations due to either a poor initial starting point or the use of biased estimates (which is common in dynamic programming) during the early iterations. For large problems, it is possible that we have to estimate thousands of parameters. It seems unlikely that all the parameters will approach their true value at the same rate. Figure 6.6 shows the change in estimates of the value of being in different states, illustrating the wide variation in learning rates that can occur within the same dynamic program.

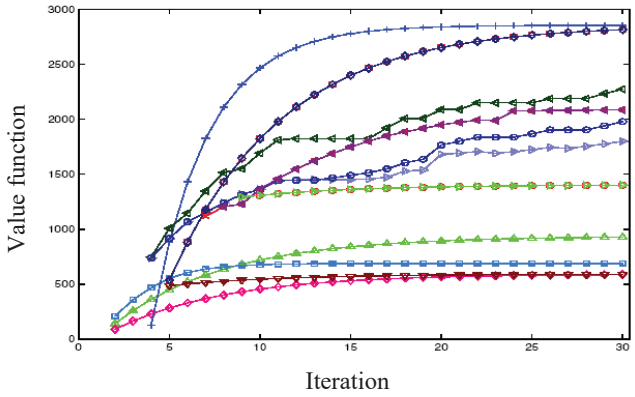


Figure 6.6 Different parameters can undergo significantly different initial rates.

Adaptive stepsizes try to adjust to the data in a way that keeps the stepsize larger while the parameter being estimated is still changing quickly. Balancing noise against the change in the underlying signal, particularly when both of these are unknown, is a difficult challenge.

6.2.2 Convergence conditions

When the stepsize depends on the history of the process, the stepsize itself becomes a random variable, which means we could replace the stepsize α_n with $\alpha_n(\omega)$ to express its dependence on the sample path ω that we are following. This change requires some subtle modifications to our requirements for convergence (equations (6.11) and (6.12)). For technical reasons, our convergence criteria change to

$$\alpha_n > 0, \text{ almost surely,} \tag{6.19}$$

$$\sum_{n=0}^{\infty} \alpha_n = \infty, \text{ almost surely,} \tag{6.20}$$

$$\mathbb{E}\left\{\sum_{n=0}^{\infty} (\alpha_n)^2\right\} < \infty. \tag{6.21}$$

The condition "almost surely" (universally abbreviated "a.s.") means that equations (6.19)-(6.20) holds for every sample path ω, and not just on average. For example we could replace equation (6.20) with

$$\sum_{n=0}^{\infty} \alpha_n(\omega) = \infty, \text{ for all } \omega, p(\omega) > 0. \tag{6.22}$$

More precisely, we mean every sample path ω that might actually happen, which is why we introduced the codition $p(\omega) > 0$. We exclude sample paths where the probability that the sample path would happen is zero, which is something that mathematicians stress over. Note that almost surely is *not* the same as requiring

$$\mathbb{E}\left\{\sum_{n=0}^{\infty} \alpha_n\right\} = \infty, \tag{6.23}$$

which requires that this condition be satisfied on average but would allow it to fail for specific sample paths. This is a much weaker condition, and would *not* guarantee convergence every time we run the algorithm. Note that the condition (6.21) does, in fact, use an expectation, which hints that this is a weaker condition.

For the reasons behind these conditions, go to our "Why does it work" section (5.10). Note that while the theoretical conditions provide broad guidance, there are significant empirical differences between policies that satisfy the conditions for asymptotic optimality.

6.2.3 A collection of stochastic policies

The desire to find stepsize policies that adapt to the data has become a cottage industry which has produced a variety of formulas with varying degrees of sophistication and convergence guarantees. This section provides a brief sample of some popular policies, some (such as AdaGrad) with strong performance guarantees. Later, we present some optimal policies for specialized problems.

To present our adaptive stepsize formulas, we need to define a few quantities. Recall that our basic updating expression is given by

$$\bar{\mu}^n = (1-\alpha_{n-1})\bar{\mu}^{n-1} + \alpha_{n-1}W^n.$$

$\bar{\mu}^{n-1}$ is an estimate of whatever value we are estimating. Note that we may be estimating a function $\mu(x) = EF(x,W)$ (for discrete x), or we may be estimating a continuous function where smoothing is required. We can compute an error by comparing the difference between our current estimate $\bar{\mu}^{n-1}$ and our latest observation W^n which we write as

$$\varepsilon^n = \bar{\mu}^{n-1} - W^n.$$

Some formulas depend on tracking changes in the sign of the error. This can be done using the indicator function

$$\mathbb{1}_{\{X\}} = \begin{cases} 1 & \text{if the logical condition } X \text{ is true,} \\ 0 & \text{otherwise.} \end{cases}$$

Thus, $\mathbb{1}_{\varepsilon^n\varepsilon^{n-1}<0}$ indicates if the sign of the error has changed in the last iteration.

Below, we first summarize three classic rules. Kesten's rule is the oldest and is perhaps the simplest illustration of an adaptive stepsize rule. Trigg's formula is a simple rule widely used in the demand forecasting community. The stochastic gradient adaptive stepsize rule enjoys a theoretical convergence proof, but is controlled by several tunable parameters that complicate its use in practice. Then, we present three more modern rules: ADAM, AdaGrad and RMSProp are rules that were developed by the machine learning community for fitting neural networks to data.

Kesten's rule

Kesten's rule was one of the earliest stepsize rules which took advantage of a simple principle. If we are far from the optimal, the gradients $\nabla_x F(x^n, W^{n+1})$ tend to point in the same direction. As we get closer to the optimum, the gradients start to switch directions. Exploiting this simple observation, Kesten proposed the simple rule

$$\alpha_{n-1} = \frac{\theta}{\theta + K^n - 1}, \tag{6.24}$$

where θ is a parameter to be calibrated. K^n counts the number of times that the sign of the error has changed, where we use

$$K^n = \begin{cases} n & \text{if } n = 1, 2, \\ K^{n-1} + \mathbb{1}_{\{(\nabla_x F(x^{n-1}, W^n))^T \nabla_x F(x^n, W^{n+1}) < 0\}} & \text{if } n > 2. \end{cases} \tag{6.25}$$

Kesten's rule is particularly well suited to initialization problems. It slows the reduction in the stepsize as long as successive gradients generally point in the same direction. They decline when the gradients begin to alternate sign, indicating that we are moving around the optimum.

Trigg's formula

Let $S(\cdot)$ be the smoothed estimate of errors calculated using

$$S(\varepsilon^n) = (1 - \beta)S(\varepsilon^{n-1}) + \beta\varepsilon^n.$$

Trigg's formula is given by

$$\alpha_n = \frac{|S(\varepsilon^n)|}{S(|\varepsilon^n|)}. \tag{6.26}$$

The formula takes advantage of the simple property that smoothing on the absolute value of the errors is greater than or equal to the absolute value of the smoothed errors. If there is a series of errors with the same sign, that can be taken as an indication that there is a significant difference between the true mean and our estimate of the mean, which means we would like larger stepsizes.

Stochastic gradient adaptive stepsize rule

This class of rules uses stochastic gradient logic to update the stepsize. We first compute

$$\psi^n = (1 - \alpha_{n-1})\psi^{n-1} + \varepsilon^n. \tag{6.27}$$

The stepsize is then given by

$$\alpha_n = \left[\alpha_{n-1} + \nu\psi^{n-1}\varepsilon^n\right]_{\alpha_-}^{\alpha_+}, \tag{6.28}$$

where α_+ and α_- are, respectively, upper and lower limits on the stepsize. $[\cdot]_{\alpha_-}^{\alpha_+}$ represents a projection back into the interval $[\alpha_-, \alpha_+]$, and ν is a scaling factor. $\psi^{n-1}\varepsilon^n$ is a stochastic gradient that indicates how we should change the stepsize to improve the error. Since the stochastic gradient has units that are the square of the units of the error, while the stepsize is unitless, ν has to perform an important scaling function. The equation $\alpha_{n-1} + \nu\psi^{n-1}\varepsilon^n$ can easily produce stepsizes that are larger than 1 or smaller than 0, so it is customary to specify an allowable interval (which is generally smaller than (0,1)). This rule has provable convergence, but in practice, ν, α_+ and α_- all have to be tuned.

ADAM

ADAM (Adaptive Moment Estimation) is another stepsize policy that has attracted attention in recent years. As above, let $g^n = \nabla_x F(x^{n-1}, W^n)$ be our gradient, and let g_i^n be the i^{th} element. ADAM proceeds by adaptively computing means and variances according to

$$m_i^n = \beta_1 m_i^{n-1} + (1 - \beta_1)g_i^n, \tag{6.29}$$

$$v_i^n = \beta_2 v_i^{n-1} + (1 - \beta_2)(g_i^n)^2. \tag{6.30}$$

These updating equations introduce biases when the data is nonstationary, which is typically the case in stochastic optimization. ADAM compensates for these biases using

$$\begin{aligned}\bar{m}_i^n &= \frac{m_i^n}{1-\beta_1},\\ \bar{v}_i^n &= \frac{v_i^n}{1-\beta_2}.\end{aligned}$$

The stochastic gradient equation for ADAM is then given by

$$x_i^{n+1} = x_i^n + \frac{\eta}{\sqrt{\bar{v}_i^n + \epsilon}}\bar{m}_i^n. \tag{6.31}$$

AdaGrad

AdaGrad ("adaptive gradient") is a relatively recent stepsize policy that has attracted considerable attention in the machine learning literature which not only enjoys nice theoretical performance guarantees, but has also become quite popular because it seems to work quite well in practice.

Assume that we are trying to solve our standard problem

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

where we make the assumption that not only is x a vector, but also that the scaling for each dimension might be different (an issue we have ignored so far). To simplify the notation a bit, let the stochastic gradient with respect to $x_i,\ i = 1, \ldots, I$ be given by

$$g_i^n = \nabla_{x_i} F(x^{n-1}, W^n).$$

Now create a $I \times I$ diagonal matrix G^n where the $(i,i)^{th}$ element G_{ii}^n is given by

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

We then set a stepsize for the i^{th} dimension using

$$\alpha_{ni} = \frac{\eta}{(G_{ii}^n)^2 + \epsilon}, \tag{6.32}$$

where ϵ is a small number (e.g. 10^{-8}) to avoid the possibility of dividing by zero. This can be written in matrix form using

$$\alpha_n = \frac{\eta}{\sqrt{G^n + \epsilon}} \otimes g_t, \tag{6.33}$$

where α_n is an I-dimensional matrix.

AdaGrad does an unusually good job of adapting to the behavior of a function. It also adapts to potentially different behaviors of each dimension. For example, we might be solving a machine learning problem to learn a parameter vector θ (this would be the decision variable instead of x) for a linear model of the form

$$y = \theta_0 + \theta_1 X_1 + \theta_2 X_2 + \ldots.$$

The explanatory variables $X_1, X_2, \ldots$ can take on values in completely different ranges. In a medical setting, X_1 might be blood sugar with values between 5 and 8, while X_2 might be the weight of a patient that could range between 100 and 300 pounds. The coefficients θ_1 and θ_2 would be scaled according to the inverse of the scales of the explanatory variables.

RMSProp

RMSProp (Root Mean Squared Propagation) was designed to address the empirical observation that AdaGrad declines too quickly. We continue to let $g^n = \nabla_x F(x^n, W^{n+1})$ be our stochastic gradient. Let $\bar{g}^n$ be a smoothed version of the inner product $(g^n)^T g^n$ given by

$$\bar{g}^n = (1-\beta)\bar{g}^n + \beta\|g^n\|^2. \tag{6.34}$$

We then compute our stepsize using

$$\alpha_n = \frac{\eta}{\sqrt{\bar{g}^n}}. \tag{6.35}$$

Suggested parameter values are $\beta = 0.1$ and $\eta = 0.001$, but we always suggest performing some exploration with tunable parameters.

6.2.4 Experimental notes

A word of caution is offered when testing out stepsize rules. It is quite easy to test out these ideas in a controlled way in a simple spreadsheet on randomly generated data, but there is a big gap between showing a stepsize that works well in a spreadsheet and one that works well in specific applications. Adaptive stepsize rules work best in the presence of transient data where the degree of noise is not too large compared to the change in the signal (the mean). As the variance of the data increases, adaptive stepsize rules begin to suffer and simpler deterministic rules tend to work better.

6.3 OPTIMAL STEPSIZE POLICIES*

Given the variety of stepsize formulas we can choose from, it seems natural to ask whether there is an optimal stepsize rule. Before we can answer such a question, we have to define exactly what we mean by it. Assume that we are trying to estimate a parameter that we denote by μ that may be static, or evolving over time (perhaps as a result of learning behavior), in which case we will write it as μ^n.

At iteration n, assume we are trying to track a time-varying process μ^n. For example, when we are estimating approximate value functions $\overline{V}^n(s)$, we will use algorithms where the estimate $\overline{V}^n(s)$ tends to rise (or perhaps fall) with the iteration n. We will use a learning policy π, so we are going to designate our estimate $\bar{\mu}^{\pi,n}$ to make the dependence on the learning policy explicit. At time n, we would like to choose a stepsize policy to minimize

$$\min_{\pi} \mathbb{E}(\bar{\mu}^{\pi,n} - \mu^n)^2. \tag{6.36}$$

Here, the expectation is over the entire history of the algorithm (note that it is not conditioned on anything, although the conditioning on S^0 is implicit) and requires (in principle) knowing the true value of the parameter being estimated.

The best way to think of this is to first imagine that we have a stepsize policy such as the harmonic stepsize rule

$$\alpha_n(\theta) = \frac{\theta}{\theta + n - 1},$$

which means that optimizing over π is the same (for this stepsize policy) as optimizing over θ. Assume that we observe our process with error ε, which is to say

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

Our estimate of $\bar{\mu}^{\pi,n}$ is given by

$$\bar{\mu}^{\pi,n+1} = (1 - \alpha_n(\theta))\bar{\mu}^{\pi,n} + \alpha_n(\theta)W^{n+1}.$$

Now imagine that we create a series of sample paths ω of observations of $(\varepsilon^n)_{n=1}^N$. If we follow a particular sample realization of observation errors $(\varepsilon^n(\omega))_{n=1}^N$, then this gives us a sequence of observations $(W^n(\omega))_{n=1}^N$, which will then produce, for a given stepsize policy π, a sequence of estimates $(\bar{\mu}^{\pi,n}(\omega))_{n=1}^N$. We can now write our optimization problem as

$$\min_{\theta} \frac{1}{N} \sum_{n=1}^{N} (\bar{\mu}^{\pi,n}(\omega^n) - \mu^n)^2. \tag{6.37}$$

The optimization problem in (6.37) illustrates how we might go through the steps of optimizing stepsize policies. Of course, we will want to do more than just tune the parameter of a particular policy. We are going to want to compare different stepsize policies, such as those listed in section 6.2.

We begin our discussion of optimal stepsizes in section 6.3.1 by addressing the case of estimating a constant parameter which we observe with noise. Section 6.3.2 considers the case where we are estimating a parameter that is changing over time, but where the changes have mean zero. Finally, section 6.3.3 addresses the case where the mean may be drifting up or down with nonzero mean, a situation that we typically face when approximating a value function.

6.3.1 Optimal stepsizes for stationary data

Assume that we observe W^n at iteration n and that the observations W^n can be described by

$$W^n = \mu + \varepsilon^n$$

where μ is an unknown constant and ε^n is a stationary sequence of independent and identically distributed random deviations with mean 0 and variance σ_ε^2. We can approach the problem of estimating μ from two perspectives: choosing the best stepsize and choosing the best linear combination of the estimates. That is, we may choose to write our estimate $\bar{\mu}^n$ after n observations in the form

$$\bar{\mu}^n = \sum_{m=1}^{n} a_m^n W^m.$$

For our discussion, we will fix n and work to determine the coefficients of the vector $a_1, \ldots, a_n$ (where we suppress the iteration counter n to simplify notation). We would like our statistic to have two properties: It should be unbiased, and it should have minimum

variance (that is, it should solve (6.36)). To be unbiased, it should satisfy

$$\begin{aligned}\mathbb{E}\left[\sum_{m=1}^{n} a_m W^m\right] &= \sum_{m=1}^{n} a_m \mathbb{E}W^m \\ &= \sum_{m=1}^{n} a_m \mu \\ &= \mu,\end{aligned}$$

which implies that we must satisfy

$$\sum_{m=1}^{n} a_m = 1.$$

The variance of our estimator is given by:

$$Var(\bar{\mu}^n) = Var\left[\sum_{m=1}^{n} a_m W^m\right].$$

We use our assumption that the random deviations are independent, which allows us to write

$$\begin{aligned}Var(\bar{\mu}^n) &= \sum_{m=1}^{n} Var[a_m W^m] \\ &= \sum_{m=1}^{n} a_m^2 Var[W^m] \\ &= \sigma_\varepsilon^2 \sum_{m=1}^{n} a_m^2. \qquad (6.38)\end{aligned}$$

Now we face the problem of finding $a_1, \ldots, a_n$ to minimize (6.38) subject to the requirement that $\sum_m a_m = 1$. This problem is easily solved using the Lagrange multiplier method. We start with the nonlinear programming problem

$$\min_{\{a_1,\ldots,a_n\}} \sum_{m=1}^{n} a_m^2,$$

subject to

$$\begin{aligned}\sum_{m=1}^{n} a_m &= 1, \qquad (6.39)\\ a_m &\geq 0. \qquad (6.40)\end{aligned}$$

We relax constraint (6.39) and add it to the objective function

$$\min_{\{a_m\}} L(a, \lambda) = \sum_{m=1}^{n} a_m^2 - \lambda\left(\sum_{m=1}^{n} a_m - 1\right),$$

subject to (6.40). We are now going to try to solve $L(a, \lambda)$ (known as the "Lagrangian") and hope that the coefficients a are all nonnegative. If this is true, we can take derivatives and set them equal to zero

$$\frac{\partial L(a,\lambda)}{\partial a_m} = 2a_m - \lambda. \tag{6.41}$$

The optimal solution (a^*, λ^*) would then satisfy

$$\frac{\partial L(a,\lambda)}{\partial a_m} = 0.$$

This means that at optimality

$$a_m = \lambda/2,$$

which tells us that the coefficients a_m are all equal. Combining this result with the requirement that they sum to one gives the expected result:

$$a_m = \frac{1}{n}.$$

In other words, our best estimate is a sample average. From this (somewhat obvious) result, we can obtain the optimal stepsize, since we already know that $\alpha_{n-1} = 1/n$ is the same as using a sample average.

This result tells us that if the underlying data is stationary, and we have no prior information about the sample mean, then the best stepsize rule is the basic $1/n$ rule. Using any other rule requires that there be some violation in our basic assumptions. In practice, the most common violation is that the observations are not stationary because they are derived from a process where we are searching for the best solution.

6.3.2 Optimal stepsizes for nonstationary data - I

Assume now that our parameter evolves over time (iterations) according to the process

$$\mu^n = \mu^{n-1} + \xi^n, \tag{6.42}$$

where $\mathbb{E}\xi^n = 0$ is a zero mean drift term with variance σ_ξ^2. As before, we measure μ^n with an error according to

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

We want to choose a stepsize so that we minimize the mean squared error. This problem can be solved using a method known as the *Kalman filter*. The Kalman filter is a powerful recursive regression technique, but we adapt it here for the problem of estimating a single parameter. Typical applications of the Kalman filter assume that the variance of ξ^n, given by σ_ξ^2, and the variance of the measurement error, ε^n, given by σ_ε^2, are known. In this case, the Kalman filter would compute a stepsize (generally referred to as the gain) using

$$\alpha_n = \frac{\sigma_\xi^2}{\nu^n + \sigma_\varepsilon^2}, \tag{6.43}$$

where ν^n is computed recursively using

$$\nu^n = (1-\alpha_{n-1})\nu^{n-1} + \sigma_\xi^2. \tag{6.44}$$

Remember that $\alpha_0 = 1$, so we do not need a value of ν^0. For our application, we do not know the variances so these have to be estimated from data. We first estimate the bias using

$$\bar{\beta}^n = (1-\eta_{n-1})\,\bar{\beta}^{n-1} + \eta_{n-1}\left(\bar{\mu}^{n-1} - W^n\right), \tag{6.45}$$

where η_{n-1} is a simple stepsize rule such as the harmonic stepsize rule or McClain's formula. We then estimate the total error sum of squares using

$$\bar{\nu}^n = (1-\eta_{n-1})\,\bar{\nu}^{n-1} + \eta_{n-1}\left(\bar{\mu}^{n-1} - W^n\right)^2. \tag{6.46}$$

Finally, we estimate the variance of the error using

$$(\bar{\sigma}_\varepsilon^{2,n}) = \frac{\bar{\nu}^n - (\bar{\beta}^n)^2}{1+\bar{\lambda}^{n-1}}, \tag{6.47}$$

where $\bar{\lambda}^{n-1}$ is computed using

$$\lambda^n = \begin{cases} (\alpha_{n-1})^2, & n=1, \\ (1-\alpha_{n-1})^2\lambda^{n-1} + (\alpha_{n-1})^2, & n>1. \end{cases}$$

We use $(\bar{\sigma}_\varepsilon^{2,n})$ as our estimate of σ_ε^2. We then propose to use $\left(\bar{\beta}^n\right)^2$ as our estimate of σ_ξ^2. This is purely an approximation, but experimental work suggests that it performs quite well, and it is relatively easy to implement.

6.3.3 Optimal stepsizes for nonstationary data - II

In dynamic programming, we are trying to estimate the value of being in a state (call it v) by $\bar{v}$ which is estimated from a sequence of random observations $\hat{v}$. The problem we encounter is that $\hat{v}$ might depend on a value function approximation which is steadily increasing (or decreasing), which means that the observations $\hat{v}$ are nonstationary. Furthermore, unlike the assumption made by the Kalman filter that the mean of $\hat{v}$ is varying in a zero-mean way, our observations of $\hat{v}$ might be steadily increasing. This would be the same as assuming that $\mathbb{E}\xi = \mu > 0$ in the section above. In this section, we derive the Kalman filter learning rate for biased estimates.

Our challenge is to devise a stepsize that strikes a balance between minimizing error (which prefers a smaller stepsize) and responding to the nonstationary data (which works better with a large stepsize). We return to our basic model

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

where μ^n varies over time, but it might be steadily increasing or decreasing. This would be similar to the model in the previous section (equation (6.42)) but where ξ^n has a nonzero mean. As before we assume that $\{\varepsilon^n\}_{n=1,2,\ldots}$ are independent and identically distributed with mean value of zero and variance, σ^2. We perform the usual stochastic gradient update to obtain our estimates of the mean

$$\bar{\mu}^n(\alpha_{n-1}) = (1-\alpha_{n-1})\,\bar{\mu}^{n-1}(\alpha_{n-1}) + \alpha_{n-1}W^n. \tag{6.48}$$

We wish to find α_{n-1} that solves,

$$\min_{\alpha_{n-1}} F(\alpha_{n-1}) = \mathbb{E}\left[(\bar{\mu}^n(\alpha_{n-1}) - \mu^n)^2\right]. \tag{6.49}$$

It is important to realize that we are trying to choose α_{n-1} to minimize the *unconditional* expectation of the error between $\bar{\mu}^n$ and the true value μ^n. For this reason, our stepsize rule will be deterministic, since we are not allowing it to depend on the information obtained up through iteration n.

We assume that the observation at iteration n is unbiased, which is to say

$$\mathbb{E}\left[W^{n+1}\right] = \mu^n. \tag{6.50}$$

But the smoothed estimate is biased because we are using simple smoothing on nonstationary data. We denote this bias as

$$\begin{aligned} \beta^{n-1} &= \mathbb{E}\left[\bar{\mu}^{n-1} - \mu^n\right] \\ &= \mathbb{E}\left[\bar{\mu}^{n-1}\right] - \mu^n. \end{aligned} \tag{6.51}$$

We note that β^{n-1} is the bias computed after iteration $n-1$ (that is, after we have computed $\bar{\mu}^{n-1}$). β^{n-1} is the bias when we use $\bar{\mu}^{n-1}$ as an estimate of μ^n.

The variance of the observation W^n is computed as follows:

$$\begin{aligned} Var\left[W^n\right] &= \mathbb{E}\left[(W^n - \mu^n)^2\right] \\ &= \mathbb{E}\left[(\varepsilon^n)^2\right] \\ &= \sigma_\varepsilon^2. \end{aligned} \tag{6.52}$$

It can be shown (see section 6.7.1) that the optimal stepsize is given by

$$\alpha_{n-1} = 1 - \frac{\sigma_\varepsilon^2}{(1+\lambda^{n-1})\,\sigma_\varepsilon^2 + (\beta^{n-1})^2}, \tag{6.53}$$

where λ is computed recursively using

$$\lambda^n = \begin{cases} (\alpha_{n-1})^2, & n = 1, \\ (1-\alpha_{n-1})^2\lambda^{n-1} + (\alpha_{n-1})^2, & n > 1. \end{cases} \tag{6.54}$$

We refer to the stepsize rule in equation (6.53) as the *bias adjusted Kalman filter*, or BAKF. The BAKF stepsize formula enjoys several nice properties:

Stationary data For a sequence with a static mean, the optimal stepsizes are given by

$$\alpha_{n-1} = \frac{1}{n} \quad \forall\, n = 1, 2, \ldots\,. \tag{6.55}$$

This is the optimal stepsize for stationary data.

No noise For the case where there is no noise ($\sigma^2 = 0$), we have the following:

$$\alpha_{n-1} = 1 \quad \forall\, n = 1, 2, \ldots. \tag{6.56}$$

This is ideal for nonstationary data with no noise.

Bounded by $1/n$ At all times, the stepsize obeys

$$\alpha_{n-1} \geq \frac{1}{n} \quad \forall\, n = 1, 2, \ldots.$$

This is important since it guarantees asymptotic convergence.

These are particularly nice properties since we typically have to do parameter tuning to get this behavior. The properties are particularly when estimating value functions, since sampled estimates of the value of being in a state tends to be transient.

The problem with using the stepsize formula in equation (6.53) is that it assumes that the variance σ^2 and the bias $(\beta^n)^2$ are known. This can be problematic in real instances, especially the assumption of knowing the bias, since computing this basically requires knowing the real function. If we have this information, we do not need this algorithm.

As an alternative, we can try to estimate these quantities from data. Let

$$\begin{aligned}
\bar{\sigma}_{\varepsilon}^{2,n} &= \text{estimate of the variance of the error after iteration } n,\\
\bar{\beta}^n &= \text{estimate of the bias after iteration } n,\\
\bar{\nu}^n &= \text{estimate of the variance of the bias after iteration } n.
\end{aligned}$$

To make these estimates, we need to smooth new observations with our current best estimate, something that requires the use of a stepsize formula. We could attempt to find an optimal stepsize for this purpose, but it is likely that a reasonably chosen deterministic formula will work fine. One possibility is McClain's formula (equation (6.17)):

$$\eta_n = \frac{\eta_{n-1}}{1 + \eta_{n-1} - \bar{\eta}}.$$

A limit point such as $\bar{\eta} \in (0.05, 0.10)$ appears to work well across a broad range of functional behaviors. The property of this stepsize that $\eta_n \to \bar{\eta}$ can be a strength, but it does mean that the algorithm will not tend to converge in the limit, which requires a stepsize that goes to zero. If this is needed, we suggest a harmonic stepsize rule:

$$\eta_{n-1} = \frac{a}{a + n - 1},$$

where a in the range between 5 and 10 seems to work quite well for many dynamic programming applications.

Care needs to be used in the early iterations. For example, if we let $\alpha_0 = 1$, then we do not need an initial estimate for $\bar{\mu}^0$ (a trick we have used throughout). However, since the formulas depend on an estimate of the variance, we still have problems in the second iteration. For this reason, we recommend forcing η_1 to equal 1 (in addition to using $\eta_0 = 1$). We also recommend using $\alpha_n = 1/(n+1)$ for the first few iterations, since the estimates of $(\bar{\sigma}^2)^n$, $\bar{\beta}^n$ and $\bar{\nu}^n$ are likely to be very unreliable in the very beginning.

Figure 6.7 summarizes the entire algorithm. Note that the estimates have been constructed so that α_n is a function of information available up through iteration n.

Figure 6.8 illustrates the behavior of the bias-adjusted Kalman filter stepsize rule for two signals: very low noise (figure 6.8a) and with higher noise (figure 6.8b). For both cases, the signal starts small and rises toward an upper limit of 1.0 (on average). In both figures, we also show the stepsize $1/n$. For the low-noise case, the stepsize stays quite large. For the high noise case, the stepsize roughly tracks $1/n$ (note that it never goes below $1/n$).

Step 0. Initialization:

Step 0a. Set the baseline to its initial value, $\bar{\mu}_0$.

Step 0b. Initialize the parameters - $\bar{\beta}_0, \bar{\nu}_0$ and $\bar{\lambda}_0$.

Step 0c. Set initial stepsizes $\alpha_0 = \eta_0 = 1$, and specify the stepsize rule for η.

Step 0d. Set the iteration counter, $n = 1$.

Step 1. Obtain the new observation, W^n.

Step 2. Smooth the baseline estimate.

$$\bar{\mu}^n = (1-\alpha_{n-1})\bar{\mu}^{n-1} + \alpha_{n-1}W^n$$

Step 3. Update the following parameters:

$$\begin{aligned}
\varepsilon^n &= \bar{\mu}^{n-1} - W^n, \\
\bar{\beta}^n &= (1-\eta_{n-1})\,\bar{\beta}^{n-1} + \eta_{n-1}\varepsilon^n, \\
\bar{\nu}^n &= (1-\eta_{n-1})\bar{\nu}^{n-1} + \eta_{n-1}(\varepsilon^n)^2, \\
(\bar{\sigma}^2)^n &= \frac{\bar{\nu}^n - (\bar{\beta}^n)^2}{1+\lambda^{n-1}}.
\end{aligned}$$

Step 4. Evaluate the stepsizes for the next iteration.

$$\begin{aligned}
\alpha_n &= \begin{cases} 1/(n+1) & n = 1, 2, \\ 1 - \frac{(\bar{\sigma}^2)^n}{\bar{\nu}^n}, & n > 2, \end{cases} \\
\eta_n &= \frac{a}{a+n-1}. \quad \text{Note that this gives us } \eta_1 = 1.
\end{aligned}$$

Step 5. Compute the coefficient for the variance of the smoothed estimate of the baseline.

$$\bar{\lambda}^n = (1-\alpha_{n-1})^2\bar{\lambda}^{n-1} + (\alpha_{n-1})^2.$$

Step 6. If $n < N$, then $n = n + 1$ and go to Step 1, else stop.

Figure 6.7 The bias-adjusted Kalman filter stepsize rule.

6.4 OPTIMAL STEPSIZES FOR APPROXIMATE VALUE ITERATION*

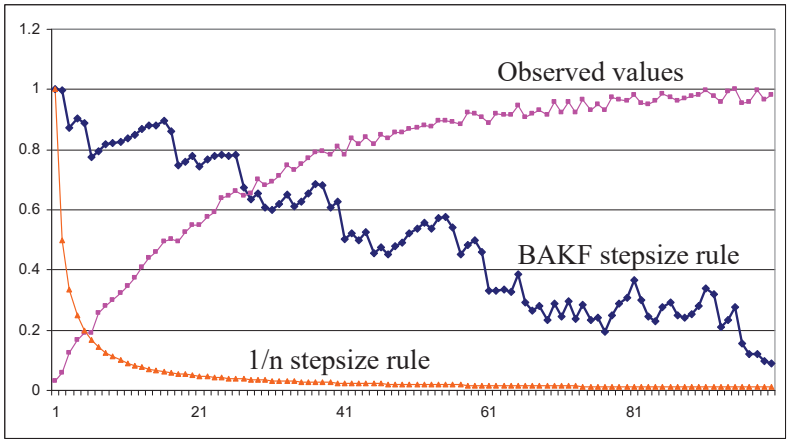
All the stepsize rules that we have presented so far are designed to estimate the mean of a nonstationary series. In this section, we develop a stepsize rule that is specifically designed for approximate value iteration, which is an algorithm we are going to see in chapters 16 and 17. Another application is Q-learning, which we first saw in section 2.1.6.

We use as our foundation a dynamic program with a single state and single action. We use the same theoretical foundation that we used in section 6.3. However, given the complexity of the derivation, we simply provide the expression for the optimal stepsize, which generalizes the BAKF stepsize rule given in equation (6.53).

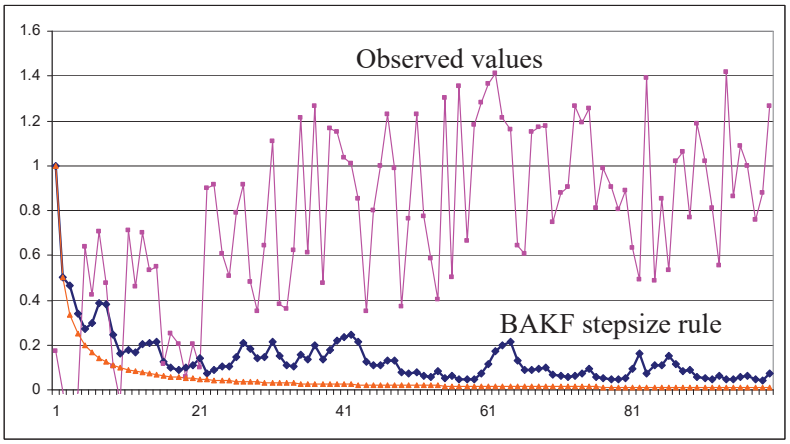
We start with the basic relationship for our single state problem

$$v^n(\alpha_{n-1}) = (1-(1-\gamma)\alpha_{n-1})v^{n-1} + \alpha_{n-1}\hat{C}^n. \tag{6.57}$$

Let $c = \hat{C}$ be the expected one-period contribution for our problem, and let $Var(\hat{C}) = \sigma^2$. For the moment, we assume c and σ^2 are known. We next define the iterative formulas for



6.8a Bias-adjusted Kalman filter for a signal with low noise.



6.8b Bias-adjusted Kalman filter for a signal with higher noise.

Figure 6.8 The BAKF stepsize rule for low-noise (a) and high-noise (b). Each figure shows the signal, the BAKF stepsizes and the stepsizes produced by the $1/n$ stepsize rule.

two series, λ^n and δ^n, as follows:

$$\lambda^n = \begin{cases} \alpha_0^2 & n = 1 \\ \alpha_{n-1}^2 + (1-(1-\gamma)\alpha_{n-1})^2\lambda^{n-1} & n > 1. \end{cases}$$

$$\delta^n = \begin{cases} \alpha_0 & n = 1 \\ \alpha_{n-1} + (1-(1-\gamma)\alpha_{n-1})\delta^{n-1} & n > 1. \end{cases}$$

It is possible to then show that

$$\begin{aligned} \mathbb{E}(v^n) &= \delta^n c, \\ Var(v^n) &= \lambda^n \sigma^2. \end{aligned}$$

Let $v^n(\alpha_{n-1})$ be defined as in equation (6.57). Our goal is to solve the optimization problem

$$\min_{\alpha_{n-1}} \mathbb{E}\left[\left(v^n(\alpha_{n-1}) - \mathbb{E}\hat{v}^n\right)^2\right] \tag{6.58}$$

The optimal solution can be shown to be given by

$$\alpha_{n-1} = \frac{(1-\gamma)\lambda^{n-1}\sigma^2 + (1-(1-\gamma)\delta^{n-1})^2 c^2}{(1-\gamma)^2\lambda^{n-1}\sigma^2 + (1-(1-\gamma)\delta^{n-1})^2 c^2 + \sigma^2}. \tag{6.59}$$

We refer to equation (6.59) as the *optimal stepsize for approximate value iteration* (OSAVI). Of course, it is only optimal for our single state problem, and it assumes that we know the expected contribution per time period c, and the variance in the contribution $\hat{C}$, σ^2.

OSAVI has some desirable properties. If $\sigma^2 = 0$, then $\alpha_{n-1} = 1$. Also, if $\gamma = 0$, then $\alpha_{n-1} = 1/n$. It is also possible to show that $\alpha_{n-1} \geq (1-\gamma)/n$ for any sample path.

All that remains is adapting the formula to more general dynamic programs with multiple states and where we are searching for optimal policies. We suggest the following adaptation. We propose to estimate a single constant $\bar{c}$ representing the average contribution per period, averaged over all states. If $\hat{C}^n$ is the contribution earned in period n, let

$$\begin{aligned}
\bar{c}^n &= (1-\nu_{n-1})\bar{c}^{n-1} + \nu_{n-1}\hat{C}^n, \\
(\bar{\sigma}^n)^2 &= (1-\nu_{n-1})(\bar{\sigma}^{n-1})^2 + \nu_{n-1}(\bar{c}^n - \hat{C}^n)^2.
\end{aligned}$$

Here, ν_{n-1} is a separate stepsize rule. Our experimental work suggests that a constant stepsize works well, and that the results are quite robust with respect to the value of ν_{n-1}. We suggest a value of $\nu_{n-1} = 0.2$. Now let $\bar{c}^n$ be our estimate of c, and let $(\bar{\sigma}^n)^2$ be our estimate of σ^2.

We could also consider estimating $\bar{c}^n(s)$ and $(\bar{\sigma}^n)^2(s)$ for each state, so that we can estimate a state-dependent stepsize $\alpha_{n-1}(s)$. There is not enough experimental work to support the value of this strategy, and lacking this we favor simplicity over complexity.

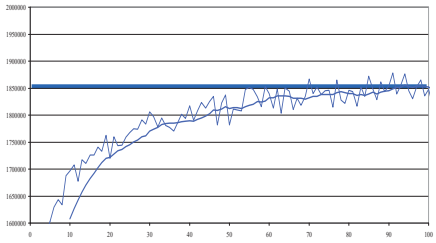
6.5 CONVERGENCE

A practical issue that arises with all stochastic approximation algorithms is that we simply do not have reliable, implementable stopping rules. Proofs of convergence in the limit are an important theoretical property, but they provide no guidelines or guarantees in practice.

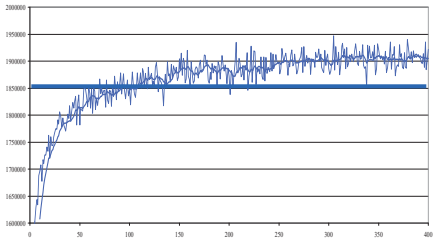
A good illustration of the issue is given in figure 6.9. Figure 6.9a shows the objective function for a dynamic program over 100 iterations (in this application, a single iteration required approximately 20 minutes of CPU time). The figure shows the objective function for an ADP algorithm which was run 100 iterations, at which point it appeared to be flattening out (evidence of convergence). Figure 6.9b is the objective function for the same algorithm run for 400 iterations, which shows that there remained considerable room for improvement after 100 iterations.

We refer to this behavior as "apparent convergence," and it is particularly problematic on large-scale problems where run times are long. Typically, the number of iterations needed before the algorithm "converges" requires a level of subjective judgment. When the run times are long, wishful thinking can interfere with this process.

Complicating the analysis of convergence in stochastic search is the behavior in some problems to go through periods of stability which are simply a precursor to breaking



6.9a: Objective function over 100 iterations.



6.9b: Objective function over 400 iterations.

Figure 6.9 The objective function, plotted over 100 iterations (a), displays "apparent convergence." The same algorithm, continued over 400 iterations (b), shows significant improvement.

through to new plateaus. During periods of exploration, a stochastic gradient algorithm might discover a strategy that opens up new opportunities, moving the performance of the algorithm to an entirely new level.

Special care has to be made in the choice of stepsize rule. In any algorithm using a declining stepsize, it is possible to show a stabilizing objective function simply because the stepsize is decreasing. This problem is exacerbated when using algorithms based on value iteration, where updates to the value of being in a state depend on estimates of the values of future states, which can be biased. We recommend that initial testing of a stochastic gradient algorithm start with inflated stepsizes. After getting a sense for the number of iterations needed for the algorithm to stabilize, decrease the stepsize (keeping in mind that the number of iterations required to convergence may increase) to find the right tradeoff between noise and rate of convergence.

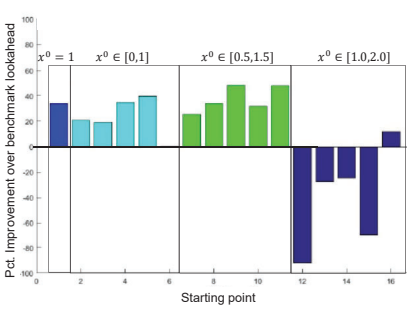
6.6 GUIDELINES FOR CHOOSING STEPSIZE FORMULAS

Given the plethora of strategies for computing stepsizes, it is perhaps not surprising that there is a need for general guidance when choosing a stepsize formula. Strategies for stepsizes are problem-dependent, and as a result any advice reflects the experience of the individual giving the advice.

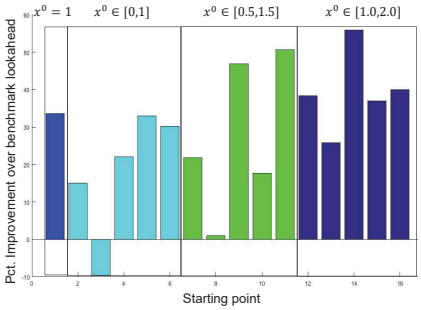
An issue that is often overlooked is the role of tuning of the stepsize policy. If a stepsize is not performing well, is it because you are not using an effective stepsize policy? Or is it because you have not properly tuned the one that you are using? Even more problematic is when you feel that you have tuned your stepsize policy as well as it can be tuned, but then you change something in your problem. For example, the distance from starting point to optimal solution matters. Changing your starting point, or modifying problem parameters so that the optimal solution moves, can change the optimal tuning of your stepsize policy.

This helps to emphasize the importance of our formulation which poses stochastic search algorithms as optimization problems *searching for the best algorithm*. Since parameter tuning for stepsizes is a manual process, people tend to overlook it, or minimize it. Figure 6.10 illustrates the risk of failing to recognize the point of tuning.

Figure 6.10(a) shows the performance of a stochastic gradient algorithm using a "tuned" stepsize, for four sets of starting points for x^0: $x^0 = 1$, $x^0 \in [0, 1.0]$, $x^0 \in [0.5, 1.5]$, and $x^0 \in [1.0, 2.0]$. Note the poor performance when the starting point was chosen in the range $x^0 \in [1.0, 2.0]$. Figure 6.10(b) shows the same algorithm after the stepsize was re-tuned for the range $x^0 \in [1.0, 2.0]$ (the same stepsize was used for all four ranges).



6.10(a)



6.10(b)

Figure 6.10 Performance of stochastic gradient algorithm using starting point $x^0 = 1$, $x^0 \in [0, 1]$, $x^0 \in [0.5, 1.5]$, and $x^0 \in [1.0, 2.0]$ using two different tuned values of the stepsize parameter θ.

With this in mind, we offer the following general strategies for choosing stepsizes:

Step 1 Start with a constant stepsize α and test out different values. Problems with a relatively high amount of noise will require smaller stepsizes. Periodically stop the search and test the quality of your solution (this will require running multiple simulations of $F(x, \widehat{W})$ and averaging). Plot the results to see roughly how many iterations are needed before your results stop improving.

Step 2 Now try the harmonic stepsize $\theta/(\theta + n - 1)$. $\theta = 1$ produces the $1/n$ stepsize rule that is provably convergent, but is likely to decline too quickly. To choose θ, look at how many iterations seemed to be needed when using a constant stepsize. If 100 iterations appears to be enough for a stepsize of 0.1, then try $\theta \approx 10$, as it produces a stepsize of roughly .1 after 100 iterations. If you need 10,000 iterations, choose $\theta \approx 1000$. But you will need to tune θ. An alternative rule is the polynomial stepsize rule $\alpha_n = 1/n^\beta$ with $\beta \in (0.5, 1]$ (we suggest 0.7 as a good starting point).

Step 3 Now start experimenting with the adaptive stepsize policies. RMSProp has become popular as of this writing for stationary stochastic search. For nonstationary settings, we suggest the BAKF stepsize rule (section 6.3.3). We will encounter an important class of nonstationary applications when we are estimating value function approximations in chapters 16 and 17.

There is always the temptation to do something simple. A constant stepsize, or a harmonic rule, are both extremely simple to implement. Keep in mind that both have a tunable parameter, and that the constant stepsize rule will not converge to anything (although the final solution may be quite acceptable). A major issue is that the best tuning of a stepsize not only depends on a problem, but also on the parameters of a problem such as the discount factor.

BAKF and OSAVI are more difficult to implement, but are more robust to the setting of the single, tunable parameter. Tunable parameters can be a major headache in the design of algorithms, and it is good strategy to absolutely minimize the number of tunable parameters your algorithm needs. Stepsize rules should be something you code once and forget about, but keep in mind the lesson of figure 6.10.

6.7 WHY DOES IT WORK*

6.7.1 Proof of BAKF stepsize

We now have what we need to derive an optimal stepsize for nonstationary data with a mean that is steadily increasing (or decreasing). We refer to this as the *bias-adjusted Kalman filter* stepsize rule (or BAKF), in recognition of its close relationship to the Kalman filter learning rate. We state the formula in the following theorem:

Theorem 6.7.1. *The optimal stepsizes $(\alpha_m)_{m=0}^n$ that minimize the objective function in equation (6.49) can be computed using the expression*

$$\alpha_{n-1} = 1 - \frac{\sigma^2}{(1+\lambda^{n-1})\,\sigma^2 + (\beta^{n-1})^2}, \tag{6.60}$$

where λ is computed recursively using

$$\lambda^n = \begin{cases} (\alpha_{n-1})^2, & n = 1 \\ (1-\alpha_{n-1})^2\lambda^{n-1} + (\alpha_{n-1})^2, & n > 1. \end{cases} \tag{6.61}$$

Proof: We present the proof of this result because it brings out some properties of the solution that we exploit later when we handle the case where the variance and bias are unknown. Let $F(\alpha_{n-1})$ denote the objective function from the problem stated in (6.49).

$$\begin{aligned} F(\alpha_{n-1}) &= \mathbb{E}\left[(\bar{\mu}^n(\alpha_{n-1}) - \mu^n)^2\right] && (6.62)\\ &= \mathbb{E}\left[\left((1-\alpha_{n-1})\,\bar{\mu}^{n-1} + \alpha_{n-1}W^n - \mu^n\right)^2\right] && (6.63)\\ &= \mathbb{E}\left[\left((1-\alpha_{n-1})\left(\bar{\mu}^{n-1} - \mu^n\right) + \alpha_{n-1}\left(W^n - \mu^n\right)\right)^2\right] && (6.64)\\ &= (1-\alpha_{n-1})^2\,\mathbb{E}\left[\left(\bar{\mu}^{n-1} - \mu^n\right)^2\right] + (\alpha_{n-1})^2\,\mathbb{E}\left[(W^n - \mu^n)^2\right] \\ &\quad +2\alpha_{n-1}\,(1-\alpha_{n-1})\,\underbrace{\mathbb{E}\left[\left(\bar{\mu}^{n-1} - \mu^n\right)(W^n - \mu^n)\right]}_{I}. && (6.65) \end{aligned}$$

Equation (6.62) is true by definition, while (6.63) is true by definition of the updating equation for $\bar{\mu}^n$. We obtain (6.64) by adding and subtracting $\alpha_{n-1}\mu^n$. To obtain (6.65), we expand the quadratic term and then use the fact that the stepsize rule, α_{n-1}, is deterministic, which allows us to pull it outside the expectations. Then, the expected value of the cross-product term, I, vanishes under the assumption of independence of the observations and the objective function reduces to the following form

$$F(\alpha_{n-1}) = (1-\alpha_{n-1})^2\,\mathbb{E}\left[\left(\bar{\mu}^{n-1} - \mu^n\right)^2\right] + (\alpha_{n-1})^2\,\mathbb{E}\left[(W^n - \mu^n)^2\right]. \tag{6.66}$$

In order to find the optimal stepsize, α_{n-1}^*, that minimizes this function, we obtain the first-order optimality condition by setting $\frac{\partial F(\alpha_{n-1})}{\partial \alpha_{n-1}} = 0$, which gives us

$$-2\left(1-\alpha_{n-1}^*\right)\mathbb{E}\left[\left(\bar{\mu}^{n-1} - \mu^n\right)^2\right] + 2\alpha_{n-1}^*\mathbb{E}\left[(W^n - \mu^n)^2\right] = 0. \tag{6.67}$$

Solving this for α_{n-1}^* gives us the following result

$$\alpha_{n-1}^* = \frac{\mathbb{E}\left[\left(\bar{\mu}^{n-1} - \mu^n\right)^2\right]}{\mathbb{E}\left[\left(\bar{\mu}^{n-1} - \mu^n\right)^2\right] + \mathbb{E}\left[(W^n - \mu^n)^2\right]}. \tag{6.68}$$

Recall that we can write $(\bar{\mu}^{n-1} - \mu^n)^2$ as the sum of the variance plus the bias squared using

$$\mathbb{E}\left[\left(\bar{\mu}^{n-1} - \mu^n\right)^2\right] = \lambda^{n-1}\sigma^2 + \left(\beta^{n-1}\right)^2. \tag{6.69}$$

Using (6.69) and $\mathbb{E}\left[(W^n - \mu^n)^2\right] = \sigma^2$ in (6.68) gives us

$$\begin{aligned}\alpha_{n-1} &= \frac{\lambda^{n-1}\sigma^2 + (\beta^{n-1})^2}{\lambda^{n-1}\sigma^2 + (\beta^{n-1})^2 + \sigma^2} \\ &= 1 - \frac{\sigma^2}{(1+\lambda^{n-1})\,\sigma^2 + (\beta^{n-1})^2},\end{aligned}$$

which is our desired result (equation (6.60)). □

From this result, we can next establish several properties through the following corollaries.

Corollary 6.7.1. *For a sequence with a static mean, the optimal stepsizes are given by*

$$\alpha_{n-1} = \frac{1}{n} \quad \forall\, n = 1, 2, \ldots\,. \tag{6.70}$$

Proof: In this case, the mean $\mu^n = \mu$ is a constant. Therefore, the estimates of the mean are unbiased, which means $\beta^n = 0 \quad \forall t = 2, \ldots,$. This allows us to write the optimal stepsize as

$$\alpha_{n-1} = \frac{\lambda^{n-1}}{1+\lambda^{n-1}}. \tag{6.71}$$

Substituting (6.71) into (6.54) gives us

$$\alpha_n = \frac{\alpha_{n-1}}{1+\alpha_{n-1}}. \tag{6.72}$$

If $\alpha_0 = 1$, it is easy to verify (6.70). □

For the case where there is no noise ($\sigma^2 = 0$), we have the following:

Corollary 6.7.2. *For a sequence with zero noise, the optimal stepsizes are given by*

$$\alpha_{n-1} = 1 \quad \forall\, n = 1, 2, \ldots. \tag{6.73}$$

The corollary is proved by simply setting $\sigma^2 = 0$ in equation (6.53).

As a final result, we obtain

Corollary 6.7.3. *In general,*

$$\alpha_{n-1} \geq \frac{1}{n} \quad \forall\, n = 1, 2, \ldots.$$

Proof: We leave this more interesting proof as an exercise to the reader (see exercise 6.18).

Corollary 6.7.3 is significant since it establishes one of the conditions needed for convergence of a stochastic approximation method, namely that $\sum_{n=1}^{\infty} \alpha_n = \infty$. An open theoretical question, as of this writing, is whether the BAKF stepsize rule also satisfies the requirement that $\sum_{n=1}^{\infty} (\alpha_n)^2 < \infty$.

6.8 BIBLIOGRAPHIC NOTES

Sections 6.1 - 6.2 A number of different communities have studied the problem of "stepsizes," including the business forecasting community (Brown (1959), Holt et al. (1960), Brown (1963), Giffin (1971), Trigg (1964), Gardner (1983)), artificial intelligence (Jaakkola et al. (1994), Darken & Moody (1991), Darken et al. (1992), Sutton & Singh (1994)), stochastic programming (Kesten (1958) , Mirozahmedov & Uryasev (1983) , Pflug (1988), Ruszczyński & Syski (1986)) and signal processing (Goodwin & Sin (1984), Douglas & Mathews (1995)). The neural network community refers to "learning rate schedules"; see Haykin (1999). Even-dar & Mansour (2003) provides a thorough analysis of convergence rates for certain types of stepsize formulas, including $1/n$ and the polynomial learning rate $1/n^{\beta}$, for Q-learning problems. These sections are based on the presentation in Powell & George (2006)

Broadie et al. (2011) revisits the stepsize conditions (6.19)-(6.19).

Section 6.3.1 - The optimality of averaging for stationary data is well known. Our presentation was based on Kushner & Yin (2003)[pp. 1892-185].

Section 6.3.2 - This result for nonstationary data is a classic result from Kalman filter theory (see, for example, Meinhold & Singpurwalla (2007)).

Section 6.3.3 - The BAKF stepsize formula was developed by Powell & George (2006), where it was initially called the "optimal stepsize algorithm" (or OSA).

Section 6.4 - The OSAVI stepsize formula for approximate value iteration was developed in Ryzhov et al. (2015).

Section 6.6 - Figure 6.10 was prepared by Saeed Ghadimi.

EXERCISES

Review questions

6.1 What is a harmonic stepsize policy? Show that a stepsize $\alpha_n = 1/n$ is the same as simple averaging.

6.2 What three conditions have to be satisfied for convergence of a deterministic stepsize policy.

6.3 Describe Kesten's rule and provide an intuitive explanation for the design of this policy.

6.4 Assume that the stepsize α_n is an adaptive (that is, stochastic) stepsize policy. What do we mean when we require

$$\sum_{n=0}^{\infty} \alpha_n = \infty$$

to be true *almost surely*. Why is this not equivalent to requiring

$$\mathbb{E}\left\{\sum_{n=0}^{\infty} \alpha_n\right\} = \infty?$$

What is the practical implication of requiring the condition to be true "almost surely."

6.5 Explain why $1/n$ is the *optimal* stepsize policy when estimating the mean of a random variable from observations that are stationary over the iterations.

6.6 Give the underlying stochastic model assumed by the Kalman filter. What is the optimal policy for this model?

Computational exercises

6.7 Let U be a uniform $[0, 1]$ random variable, and let

$$\mu^n = 1 - \exp(-\theta_1 n).$$

Now let $\hat{R}^n = \mu^n + \theta_2(U^n - .5)$. We wish to try to estimate μ^n using

$$\bar{R}^n = (1 - \alpha_{n-1})\bar{R}^{n-1} + \alpha_{n-1}\hat{R}^n.$$

In the exercises below, estimate the mean (using $\bar{R}^n$) and compute the standard deviation of $\bar{R}^n$ for $n = 1, 2, \ldots, 100$, for each of the following stepsize rules:

- $\alpha_{n-1} = 0.10$.
- $\alpha_{n-1} = a/(a + n - 1)$ for $a = 1, 10$.
- Kesten's rule.
- The bias-adjusted Kalman filter stepsize rule.

For each of the parameter settings below, compare the rules based on the average error (1) over all 100 iterations and (2) in terms of the standard deviation of $\bar{R}^{100}$.

(a) $\theta_1 = 0, \theta_2 = 10$.

(b) $\theta_1 = 0.05, \theta_2 = 0$.

(c) $\theta_1 = 0.05, \theta_2 = 0.2$.

(d) $\theta_1 = 0.05, \theta_2 = 0.5$.

(e) Now pick the single stepsize that works the best on all four of the above exercises.

6.8 Consider a random variable given by $R = 10U$ (which would be uniformly distributed between 0 and 10). We wish to use a stochastic gradient algorithm to estimate the mean of R using the iteration $\bar{\theta}^n = \bar{\theta}^{n-1} - \alpha_{n-1}(R^n - \bar{\theta}^{n-1})$, where R^n is a Monte Carlo sample of R in the n^{th} iteration. For each of the stepsize rules below, use the mean squared error

$$MSE = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(R^n - \bar{\theta}^{n-1})^2}. \tag{6.74}$$

to measure the performance of the stepsize rule to determine which works best, and compute an estimate of the bias and variance at each iteration. If the stepsize rule requires choosing a parameter, justify the choice you make (you may have to perform some test runs).

(a) $\alpha_{n-1} = 1/n$.

(b) Fixed stepsizes of $\alpha_n = .05, .10$ and .20.

(c) The stochastic gradient adaptive stepsize rule (equations 6.27)-(6.28)).

(d) The Kalman filter (equations (6.43)-(6.47)).

(e) The optimal stepsize rule (algorithm 6.7).

6.9 Repeat exercise 6.8 using

$$R^n = 10(1 - e^{-0.1n}) + 6(U - 0.5).$$

6.10 Repeat exercise 6.8 using

$$R^n = \Big(10/(1 + e^{-0.1(50-n)})\Big) + 6(U - 0.5).$$

6.11 Use a stochastic gradient algorithm to solve the problem

$$\min_x \frac{1}{2}(X - x)^2,$$

where X is a random variable. Use a harmonic stepsize rule (equation (6.15)) with parameter $\theta = 5$. Perform 1000 iterations assuming that you observe $X^1 = 6, X^2 = 2, X^3 = 5$ (this can be done in a spreadsheet). Use a starting initial value of $x^0 = 10$. What is the best possible formula for θ for this problem?

6.12 Consider a random variable given by $R = 10U$ (which would be uniformly distributed between 0 and 10). We wish to use a stochastic gradient algorithm to estimate the mean of R using the iteration $\bar{\mu}^n = \bar{\mu}^{n-1} - \alpha_{n-1}(R^n - \bar{\mu}^{n-1})$, where R^n is a Monte Carlo sample of R in the n^{th} iteration. For each of the stepsize rules below, use equation (6.74) (see exercise 6.8) to measure the performance of the stepsize rule to determine which works best, and compute an estimate of the bias and variance at each iteration. If the stepsize rule requires choosing a parameter, justify the choice you make (you may have to perform some test runs).

(a) $\alpha_{n-1} = 1/n$.

(b) Fixed stepsizes of $\alpha_n = .05, .10$ and .20.

(c) The stochastic gradient adaptive stepsize rule (equations (6.27)-(6.28)).

(d) The Kalman filter (equations (6.43)-(6.47)).

(e) The optimal stepsize rule (algorithm 6.7).

6.13 Repeat exercise 6.8 using

$$R^n = 10(1 - e^{-0.1n}) + 6(U - 0.5).$$

6.14 Repeat exercise 6.8 using

$$R^n = \Big(10/(1 + e^{-0.1(50-n)})\Big) + 6(U - 0.5).$$

6.15 Let U be a uniform $[0,1]$ random variable, and let

$$\mu^n = 1 - \exp(-\theta_1 n).$$

Now let $\hat{R}^n = \mu^n + \theta_2(U^n - .5)$. We wish to try to estimate μ^n using

$$\bar{R}^n = (1-\alpha_{n-1})\bar{R}^{n-1} + \alpha_{n-1}\hat{R}^n.$$

In the exercises below, estimate the mean (using $\bar{R}^n$) and compute the standard deviation of $\bar{R}^n$ for $n = 1, 2, \ldots, 100$, for each of the following stepsize rules:

- $\alpha_{n-1} = 0.10$.
- $\alpha_{n-1} = \theta/(\theta + n - 1)$ for $a = 1, 10$.
- Kesten's rule.
- The bias-adjusted Kalman filter stepsize rule.

For each of the parameter settings below, compare the rules based on the average error (1) over all 100 iterations and (2) in terms of the standard deviation of $\bar{R}^{100}$.

(a) $\theta_1 = 0, \theta_2 = 10$.

(b) $\theta_1 = 0.05, \theta_2 = 0$.

(c) $\theta_1 = 0.05, \theta_2 = 0.2$.

(d) $\theta_1 = 0.05, \theta_2 = 0.5$.

(e) Now pick the single stepsize that works the best on all four of the above exercises.

Theory questions

6.16 Show that if we use a stepsize rule $\alpha_{n-1} = 1/n$, then $\bar{\mu}^n$ is a simple average of $W^1, W^2, \ldots, W^n$ (thus proving equation 6.14). Use this result to argue that any solution of equation (6.7) produces the mean of W.

6.17 The proof in section 5.10.3 was performed assuming that μ is a scalar. Repeat the proof assuming that μ 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 $\|a+b\| \leq \|a\| + \|b\|$.

6.18 Prove corollary 6.7.3.

6.19 The bias adjusted Kalman filter (BAKF) stepsize rule (equation (6.53)), is given by

$$\alpha_{n-1} = 1 - \frac{\sigma_\varepsilon^2}{(1+\lambda^{n-1})\,\sigma_\varepsilon^2 + (\beta^{n-1})^2},$$

where λ is computed recursively using

$$\lambda^n = \begin{cases} (\alpha_{n-1})^2, & n = 1 \\ (1-\alpha_{n-1})^2\lambda^{n-1} + (\alpha_{n-1})^2, & n > 1. \end{cases}$$

Show that for a stationary data series, where the bias $\beta^n = 0$, produces stepsizes that satisfy

$$\alpha_{n-1} = \frac{1}{n} \quad \forall\, n = 1, 2, \ldots\ .$$

Problem solving questions

6.20 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 $1 \leq n \leq 1000$.

6.21 Show that if we use a stepsize rule $\alpha_{n-1} = 1/n$, then $\bar{\mu}^n$ is a simple average of $W^1, W^2, \ldots, W^n$ (thus proving equation 6.14).

6.22 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.

6.23 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

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

(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 $\bar{\mu}^f_t$. 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:

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

where U^f_t and U^s_t 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 $\bar{\mu}^f_0 = 30$ 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?

Sequential decision analytics and modeling

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

6.24 Read sections 5.1-5.6 on the static shortest path problem. We are going to focus on the extension in section 5.6, where the traveler gets to see the actual link cost $\hat{c}_{ij}$ before traversing the link.

a) Write out the five elements of this dynamic model. Use our style of representing the policy as $X^\pi(S_t)$ without specifying the policy.

b) We are going to use a VFA-based policy which requires estimating the function:

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

We cover value function approximations in much greater depth later, but at the moment, we are interested in the stepsize α_n, which has a major impact on the performance of the system. The ADP algorithm has been implemented in Python, which can be downloaded from `http://tinyurl.com/sdagithub` using the module "StochasticShortestPath_Static." The code currently uses the harmonic stepsize rule

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

where θ^α is a tunable parameter. Run the code for 50 iterations using $\theta^\alpha = 1, 2, 5, 10, 20, 50$ and report on the performance.

c) Implement the stepsize rule RMSProp (described in section 6.2.3) (which has its own tunable parameter), and compare your best implementation of RMSProp with your best version of the harmonic stepsize.

Diary problem

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

6.25 Try to identify at least one, but more if possible, parameters (or functions) that you would have to adaptively estimate in an online fashion, either from a flow of real data, or from an iterative search algorithm. For each case, answer the following:

a) Describe the characteristics of the observations in terms of the degree of stationary or nonstationary behavior, the amount of noise, and whether the series might undergo sudden shifts (this would only be the case for data coming from live observations).

b) Suggest one deterministic stepsize policy, and one adaptive stepsize policy, for each data series, and explain your choice. Then compare these to the BAKF policy and discuss strengths and weaknesses.

Bibliography

Broadie, M., Cicek, D. & Zeevi, a. (2011), 'General Bounds and Finite-Time Improvement for the Kiefer-Wolfowitz Stochastic Approximation Algorithm', *Operations Research* **59**(5), 1211–1224.

Brown, R. G. (1959), *Statistical Forecasting for Inventory Control*, McGraw-Hill, New York.

Brown, R. G. (1963), *Smoothing, Forecasting and Prediction of Discrete Time Series*, Prentice-Hall, Englewood Cliffs, N.J.

Darken, C. & Moody, J. (1991), Note on Learning Rate Schedules for Stochastic Optimization, *in* R. P. Lippmann, J. Moody & D. S. Touretzky, eds, 'Advances in Neural Information Processing Systems 3', pp. 1009–1016.

Darken, C., Chang, J. & Moody, J. (1992), 'Learning Rate Schedules for Faster Stochastic Gradient Search', *Neural Networks for Signal Processing 2 - Proceedings of the 1992 IEEE Workshop*.

Douglas, S. C. & Mathews, V. J. (1995), 'Stochastic Gradient Adaptive Step Size Algorithms for Adaptive Filtering', *Proc. International Conference on Digital Signal Processing, Limassol, Cyprus* **1**, 142–147.

Even-dar, E. & Mansour, Y. (2003), 'Learning rates for Q-learning', *Journal of Machine Learning Research* **vol**, 5pp1–25.

Gardner, E. S. (1983), 'Automatic Monitoring of Forecast Errors', *Journal of Forecasting* **2**, 1–21.

Giffin, W. C. (1971), *Introduction to Operations Engineering*, R. D. Irwin, Inc., Homewood, IL.

Goodwin, G. C. & Sin, K. S. (1984), *Adaptive Filtering and Control*, Prentice-Hall, Englewood Cliffs, NJ.

Haykin, S. (1999), *Neural Networks: A comprehensive foundation*, Prentice Hall, Englewood Cliffs, N.J.

Holt, C. C., Modigliani, F., Muth, J. & Simon, H. (1960), *Planning, Production, Inventories and Work Force*, Prentice-Hall, Englewood Cliffs, NJ.

Jaakkola, T., Singh, S. P. & Jordan, M. I. (1994), 'Reinforcement learning algorithm for partially observable Markov decision problems', *Advances in Neural Information Processing Systems* **7**, 345.

Kesten, H. (1958), 'Accelerated Stochastic Approximation', *The Annals of Mathematical Statistics* **29**, 41–59.

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

Meinhold, R. J. & Singpurwalla, N. D. (2007), 'Understanding the Kalman Filter', **37**(2), 123–127.

Mirozahmedov, F. & Uryasev, S. (1983), 'Adaptive Stepsize Regulation for Stochastic Optimization Algorithm', *Zurnal vicisl. mat. i. mat. fiz. 23* **6**, 1314–1325.

Pflug, G. (1988), Stepsize rules, stopping times and their implementation in stochastic quasigradient algorithms, *in* 'Numerical Techniques for Stochastic Optimization', Springer-Verlag, New York, pp. 353–372.

Powell, W. B. & George, A. P. (2006), 'Adaptive stepsizes for recursive estimation with applications in approximate dynamic programming', *Journal of Machine Learning* **65**(1), 167–198.

Ruszczyński, A. & Syski, W. (1986), 'A method of aggregate stochastic subgradients with on-line stepsize rules for convex stochastic programming problems', *Mathematical Programming Study* **28**, 113–131.

Ryzhov, I. O., Frazier, P. I. & Powell, W. B. (2015), 'A new optimal stepsize for approximate dynamic programming', *IEEE Transactions on Automatic Control* **60**(3), 743–758.

Sutton, R. S. & Singh, S. P. (1994), On Step-Size and Bias in Temporal-Difference Learning, *in* C. for System Science, ed., 'Eight Yale Workshop on Adaptive and Learning Systems', Yale University, pp. 91–96.

Trigg, D. W. (1964), 'Monitoring a forecasting system', *Operations Research Quarterly* **15**, 271–274.