

## 2

# Applications to Learning, Repeated Games, State Dependent Noise, and Queue Optimization

## 2.0 Outline of Chapter

This chapter deals with more specific classes of examples, which are of increasing importance in current applications in many areas of technology and operations research. They are described in somewhat more detail than those of Chapter 1. Each example should be taken as one illustration of a class of problems in a rapidly expanding literature. They demonstrate some of the great variety of ways that recursive stochastic algorithms arise. Section 1 deals with a problem in learning theory; in particular, the learning of an optimal hunting strategy by an animal, based on the history of successes and failures in repeated attempts to feed itself efficiently, and is typical of many “adaptive” models in biology. Section 2 concerns the “learning” or “training” of a neural network. In the training phase, a random series of inputs is presented to the network, and there is a desirable response to each input. The problem is to adjust the weights in the network to minimize the average distance between the actual and desired responses. This is done by a training procedure, where the weights are adjusted after each (input, output) pair is observed. Loosely speaking, the increments in the weights are proportional to stochastic estimates of the derivative of the error with respect to the weights.

Many problems in applications are of interest over a long time period. One observes a process driven by correlated noise and wishes to recursively adjust a parameter of the process so that some quantity is optimized. Suppose that for each parameter value, there is an associated stationary

probability, and one wishes to minimize a criterion that depends on these stationary probabilities. This is to be done recursively. One observes the process over a (perhaps not very long) period of time at the current parameter value, appropriately adjusts the parameter value based on the measurements, and then continues to observe the same system over the next period of time, etc. For such problems, the current measurements are affected by the “state memory in the system,” as well as by the external driving forces appearing during the current observational period. This involves the concept of Markov state-dependent noise. A versatile model for such a process is introduced in Section 3, and it is illustrated by a simple routing example.

Section 4 concerns recursive procedures for optimizing control systems or approximating their value functions. First, we consider the so-called  $Q$ -learning, where we have a controlled Markov chain model whose transition probabilities are not known and one wishes to learn the optimal strategy in the course of the system’s operation. Then we consider the problem of approximating a value function by a linear combination of some basis functions. The process might be only partially observable and its law is unknown. It need not be Markov. In the next example, we are given a Markov control problem whose law is known, and whose control is parameterized. The problem is to find the optimal parameter value by working with a single long sample path. Although there are serious practical issues in the implementation of such algorithms, they are being actively investigated and are of interest since the law of the process is often not completely known or analytical computation is much too hard. All the noise types appear, including martingale difference, correlated, and Markov state-dependent noise.

The theme of optimization of an average cost over a long time interval is continued in Section 5, which concerns the optimization of the stationary performance of a queue with respect to a parameter of the service time distribution. The IPA (infinitesimal perturbation analysis) method is used to get the sample derivatives. Notice that the effective “memory” of the system at the start of each new iteration needs to be taken into account in the convergence analysis. Analogous models and issues arise in many problems in the optimization of queueing type networks and manufacturing systems.

The example in Section 6, called “passive stochastic approximation,” is a type of nonlinear regression algorithm. One is given a sequence of “noisy” observations at parameter values that are determined *exogenously* (not subject to the control of the experimenter), and one seeks the parameter value at which the mean value of the observation is zero. The stochastic approximation algorithm is a useful nonparametric alternative to the classical method of fitting a nonlinear function (say in the mean square sense) and then locating the roots of that function. An interesting application of this idea to the problem of determining the concentration of a product in a

chemical reactor subject to randomly varying inputs is in [274, 275]. Other interesting examples can be found in [16, 225, 250] and in Chapter 3.

Section 7 concerns learning in two-player repeated games. Each player has a choice of actions, knows its own payoffs, but not those of the other player. The player tries to learn its optimal strategy via repeated games and observations of the opponents strategies. Such problems have arisen in the economics literature. The results illustrate new ways in which complex recursive stochastic algorithms arise, as well as some of the phenomena to be expected.

## 2.1 An Animal Learning Model

This example concerns the purported learning behavior of an animal (say, a lizard) as it tries to maximize its reward per unit time in its food gathering activities. There are many variations of the problem that are covered by the basic results of convergence of stochastic approximation algorithms, and they illustrate the value of the general theory for the analysis of recursive algorithms arising in diverse fields. The basic model is taken from [177, 218]. The lizard has a fixed home location and can hunt in an arbitrary finite region. In the references, the region is restricted to a circle, but this is not needed.

Insects appear at random moments. Their weights are random and will be considered surrogates for the food value they contain. If an insect appears when the lizard is prepared to hunt (i.e., the lizard is at its home location), then the lizard makes a decision whether or not to pursue, with the (implicit) goal of maximizing its long-run return per unit time. We suppose that when the lizard is active (either pursuing or returning from a pursuit), no insect will land within the hunting range. It is not claimed that the model reflects conscious processes, but it is of interest to observe whether the learning behavior is consistent with some sort of optimization via a “return for effort” principle.

Next, we define the basic sequence of intervals between decision times for the lizard. At time zero, the lizard is at its home base, and the process starts. Let  $\tau_1$  denote the time elapsed until the appearance of the first insect. If the lizard pursues, let  $r_1$  denote the time required to pursue and return to the home base and  $\tau_2$  the time elapsed after return until the next insect appears. If the lizard decides not to pursue, then  $\tau_2$  denotes the time elapsed until the appearance of the next insect. In general, let  $\tau_n$  denote the time interval until the next appearance of an insect from either the time of appearance of the  $(n - 1)$ st, if it was not pursued, or from the time of return from pursuing the  $(n - 1)$ st, if it was pursued. If the lizard pursues, let  $r_n$  denote the time to pursue and return to the home base ready to hunt again. Let  $w_n$  denote the weight of the  $n$ th insect, and let  $J_n$  denote the indicator function of that insect being caught if pursued. Let  $I_n$  denote the

indicator function of the event that the lizard decides to pursue at the  $n$ th opportunity. Let the time of completion of action on the  $(n-1)$ st decision be  $T_n$ . The pair  $(w_n, r_n)$  is known to the lizard when the insect appears. Define

$$W_n = \sum_{i=1}^{n-1} w_i I_i J_i,$$

and let  $\theta_n = T_n/W_n$ , which is the inverse of the sample consumption rate per unit time.

Suppose that the random variables  $(w_n, \tau_n, r_n, J_n)$  for  $n \geq 1$  are mutually independent in  $n$  and identically distributed, with bounded second moments, and  $w_n > 0, \tau_n > 0$  with probability one. The four components can be correlated with each other. [In fact, the independence (in  $n$ ) assumption can be weakened considerably without affecting the convergence of the learning algorithm.] Let there be a continuous function  $p(\cdot)$  such that  $E[J_n | r_n, w_n] = p(r_n, w_n) > 0$  with probability one, which is known to the lizard. The learning algorithm to be discussed performs as follows. Let  $\bar{\theta} = \liminf_n ET_n/W_n$ , where the infimum is over all (non-anticipative) strategies. The algorithm produces a sequence of estimates  $\theta_n$  such that (Section 5.7)  $\theta_n \rightarrow \bar{\theta}$  with probability one. The threshold  $\bar{\theta}$  gives the optimal long-term decision rule: Pursue only if  $r_n \leq \bar{\theta} w_n p(r_n, w_n)$ . Thus, the optimal policy is to pursue only if the ratio of the required pursuit time to the expected gain is no greater than the threshold, and this policy is approximated asymptotically by the learning procedure.

The model can be generalized further by supposing that the estimates of  $p(w_n, r_n)$  and of  $r_n$  are subject to “error.” One introduces additional random variables to account for the errors.

Let  $n \geq 1$ , and let  $\theta_1$  be an arbitrary real number. If the lizard does not pursue at the  $n$ th opportunity, then

$$\theta_{n+1} = \frac{T_n + \tau_n}{W_{n+1}} = \frac{T_n + \tau_n}{W_n}. \quad (1.1a)$$

If the insect is pursued and captured, then

$$\theta_{n+1} = \frac{T_n + \tau_n + r_n}{W_{n+1}} = \frac{T_n + \tau_n + r_n}{W_n + w_n}. \quad (1.1b)$$

If the insect is pursued but not captured, then

$$\theta_{n+1} = \frac{T_n + \tau_n + r_n}{W_n}. \quad (1.1c)$$

Thus, if the insect is pursued,

$$\theta_{n+1} = \frac{T_n + \tau_n + r_n}{W_n + w_n} J_n + \frac{T_n + \tau_n + r_n}{W_n} (1 - J_n). \quad (1.1d)$$

The lizard compares the conditional expectations, given  $(r_n, w_n, \theta_n)$  of the right sides in (1.1a) and (1.1d) and chooses the action that gives the minimum.

Define  $\epsilon_n = 1/W_n$ . Then  $\epsilon_n$  decreases only after a successful pursuit. Either (1.2a) or (1.2b) will hold, according to the choice of nonpursuit or pursuit:

$$\theta_{n+1} = \theta_n + \epsilon_n \tau_n, \quad (1.2a)$$

$$\theta_{n+1} = \theta_n + \epsilon_n(\tau_n + r_n) - \epsilon_n \theta_n w_n J_n + O(\epsilon_n^2) J_n. \quad (1.2b)$$

Under the conditions to be imposed,  $\epsilon_n \rightarrow 0$  with probability one and the term  $O(\epsilon_n^2)$  in (1.2b) can be shown to be asymptotically unimportant relative to the other terms. It does not contribute to the limit, and for notational simplicity, it will be ignored henceforth. Thus, if the insect is pursued, neglecting the “small”  $O(\epsilon_n^2)$  term, we have

$$\theta_{n+1} = \theta_n + \epsilon_n(\tau_n + r_n) - \epsilon_n \theta_n w_n p(r_n, w_n) + \epsilon_n \theta_n w_n (p(r_n, w_n) - J_n).$$

Finally, with  $K_n = I_{\{r_n - \theta_n w_n p(r_n, w_n) < 0\}}$ ,

$$\begin{aligned} \theta_{n+1} = & \theta_n + \epsilon_n \tau_n + \epsilon_n \min\{r_n - \theta_n w_n p(r_n, w_n), 0\} \\ & + \epsilon_n \theta_n w_n (p(r_n, w_n) - J_n) K_n. \end{aligned} \quad (1.3)$$

The problem is interesting partly because the step sizes decrease randomly.

For fixed nonrandom  $\theta$ , define the mean value

$$\bar{g}(\theta) = E\tau_n + E \min\{r_n - \theta w_n p(r_n, w_n), 0\},$$

and define the noise term

$$\xi_n = \tau_n + \min\{r_n - \theta_n w_n p(r_n, w_n), 0\} - \bar{g}(\theta_n) + \theta_n w_n (p(r_n, w_n) - J_n) K_n.$$

For nondegenerate cases,  $\bar{g}(\cdot)$  is Lipschitz continuous, positive for  $\theta = 0$ , approximately proportional to  $-\theta$  for large  $\theta$ , and there is a unique value  $\theta = \bar{\theta}$  at which  $\bar{g}(\theta) = 0$ . Rewrite the algorithm as

$$\theta_{n+1} = \theta_n + \epsilon_n \bar{g}(\theta_n) + \epsilon_n \xi_n. \quad (1.4)$$

The mean ODE that determines the asymptotic behavior is  $\dot{\theta} = \bar{g}(\theta)$ .

The value of  $\epsilon_n$  and the final form of the algorithm are simply consequences of the representation used for the iteration; in particular, the definition  $\epsilon_n = 1/W_n$ . This representation allows us to put the iteration into the familiar form of stochastic approximation, so that a well-known theory can be applied.

## 2.2 A Neural Network

In the decision problem of Example 3 in Subsection 1.1.3, a sequence of patterns appeared, with the  $n$ th pattern denoted by  $y_n$ . The patterns themselves were not observable, but at each time  $n$ , one could observe random  $\phi_n$  that is correlated with the  $y_n$ . We sought an affine decision rule (an affine function of the observables) that is best in the mean square sense. The statistics of the pairs  $(y_n, \phi_n)$  were not known. However, during a training period many samples of the pairs  $(y_n, \phi_n)$  were available, and a recursive linear least squares algorithm was used to sequentially get the optimal weights for the affine decision function. Thus, during the training period, we used a sequence of inputs  $\{\phi_n\}$  and chose  $\theta$  so that the outputs  $v_n = \theta' \phi_n$  matches the sequence of correct decisions  $y_n$  as closely as possible in the mean square sense. Neural networks serve a similar purpose, but the output  $v_n$  can be a fairly general nonlinear function of the input [8, 97, 193, 205, 253]. The issue of large dimensionality is discussed in [171] via the random directions Kiefer–Wolfowitz procedure (see Chapters 5 and 10). A related “Kohonen” algorithm is treated in [73].

In this model, there are two layers of “neurons,” the first layer having  $K$  neurons and the second (output layer) having only one neuron. The input at time  $n$  to each of the  $K$  neurons in the first layer is a linear combination of the observable random variables at that time. The output of each of these neurons is some nonlinear function of the input. The input to the neuron in the output layer is a linear combination of the outputs of the first layer, and the network output is some nonlinear function of the input to that output neuron. For a suitable choice of the nonlinear function (say the sigmoid), any continuous vector-valued map between input and output can be approximated by appropriate choices of the number of neurons and the weights in the linear combinations [97, 253]. In practice, the number of neurons needed for a good approximation can be very large, and much insight into the actual problem of concern might be required to get an effective network of reasonable size. Some more critical discussion from a statistical perspective is in [51, 205].

The training of the network can be described in terms of its input, output, and the desired and actual relationships between them. The definitions are illustrated in Figure 2.1.

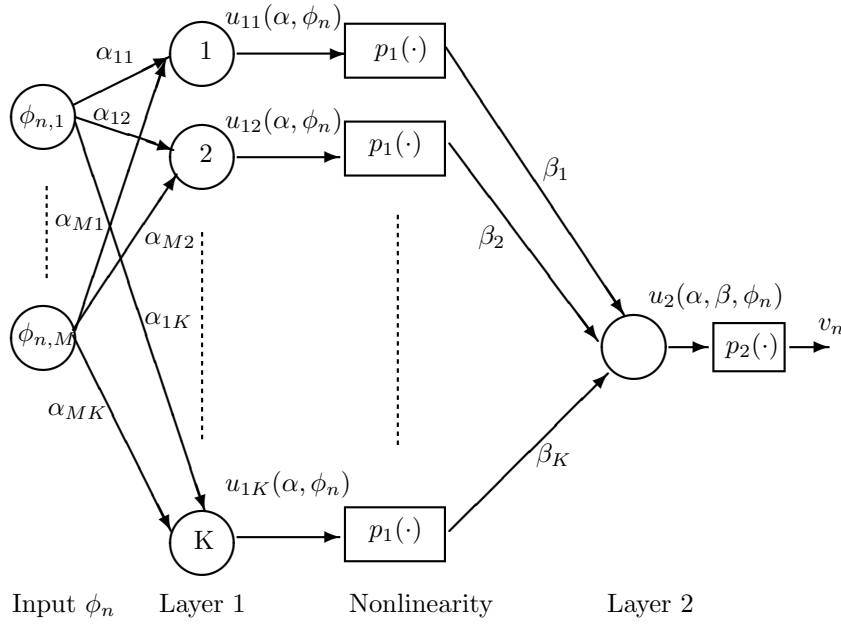


Figure 2.1. A two-layer neural network.

First we describe the network with fixed weights. There is a sequence of  $M$ -dimensional input vectors, the one at time  $n$  being  $\phi_n = (\phi_{n,1}, \dots, \phi_{n,M})$ , with associated actual network outputs  $v_n$  (that can be observed) and desired outputs  $y_n$  (that are not generally observable in applications, at least after the training period). If the network is well designed, then  $v_n$  is a “good approximation” to  $y_n$ . There are weights  $\alpha_{ij}$  such that the input at time  $n$  to “neuron”  $j$  in the first layer is

$$u_{1j}(\alpha, \phi_n) = \sum_{i=1}^M \alpha_{ij} \phi_{n,i}, \quad j = 1, \dots, K.$$

The output of this neuron is  $p_1(u_{1j}(\alpha, \phi_n))$ , where  $p_1(\cdot)$  is a real-valued antisymmetric nondecreasing and continuously differentiable function of a real variable, which is positive for positive arguments and is often chosen to be the sigmoid function. Denote the derivative by  $\dot{p}_1(\cdot)$ . There are weights  $\beta_i$  such that the input to the single second layer or “output” neuron is the linear combination of the outputs of the first layer

$$u_2(\alpha, \beta, \phi_n) = \sum_{j=1}^K \beta_j p_1(u_{1j}(\alpha, \phi_n)).$$

The output of the second layer neuron is  $u(\alpha, \beta, \phi_n) = p_2(u_2(\alpha, \beta, \phi_n))$ , where  $p_2(\cdot)$  has the properties of  $p_1(\cdot)$ . Denote the derivative of  $p_2(\cdot)$  by

$\dot{p}_2(\cdot)$ . One wishes to “train” the network to minimize the error between the desired and the actual outputs.

Suppose that there is a “training” period in which the set of pairs (input, desired output)  $\{(\phi_n, y_n), n = 1, \dots\}$ , as well as the actual outputs  $\{v_n\}$  are available. For simplicity, let us suppose that the input-output pairs  $(y_n, \phi_n)$  are mutually independent and that its distribution does not depend on  $n$ . The actual situations that can be handled are far more general and can be seen from the convergence proofs in the following chapters. Let  $\theta = (\alpha, \beta)$  denote the  $r$ -vector of weights that is to be chosen, and let  $\theta_n$  denote the value used for the  $n$ th training session. Define  $v_n = u(\theta_n, \phi_n)$ . The weights are to be recursively adjusted to minimize the mean square error  $E[y - u(\theta, \phi)]^2$ . Of course, the results are not confined to this error function. Define the sample mean square error  $e(\theta, \phi, y) = (1/2)[y - u(\theta, \phi)]^2$  and the sample error  $e_n = [y_n - v_n]$ .

The basic adaptive algorithm has the form

$$\theta_{n+1,i} = \theta_{n,i} - \epsilon_n \frac{\partial e(\theta_n, \phi_n, y_n)}{\partial \theta^i} = \theta_{n,i} + \epsilon_n e_n \frac{\partial u(\theta_n, \phi_n)}{\partial \theta^i}, \quad i = 1, \dots, r.$$

Using the formula

$$v = u(\alpha, \beta, \phi) = p_2(u_2(\alpha, \beta, \phi)) = p_2 \left( \sum_{j=1}^K \beta_j p_1(u_{1j}(\alpha, \phi)) \right),$$

the derivatives are evaluated from the formulas for repeated differentiation

$$\begin{aligned} \frac{\partial u(\theta, \phi)}{\partial \beta_j} &= \dot{p}_2(u_2(\alpha, \beta, \phi)) p_1(u_{1j}(\alpha, \phi)), \\ \frac{\partial u(\theta, \phi)}{\partial \alpha_{ij}} &= \dot{p}_2(u_2(\alpha, \beta, \phi)) \beta_j \dot{p}_1(u_{1j}(\alpha, \phi)) \phi^i, \end{aligned}$$

where  $\phi^i$  denotes the  $i$ th component of the canonical input vector  $\phi$ .

Under appropriate conditions, the ODE that characterizes the asymptotic behavior is

$$\dot{\theta}^i = \bar{g}^i(\theta) = -\frac{E \partial e(\theta, \phi, y)}{\partial \theta^i}, \quad i = 1, \dots, r.$$

Being a gradient procedure,  $\theta_n$  converges to a stationary point of this ODE. In applications there are usually many local minima. If it appears that the iteration is “hung up” at a local minimum, then several training runs might be needed, each with a different initial condition. The time required for training can be very long if there are many weights to be adjusted.

The neurons in the first layer can be connected to one another, and feedback can be incorporated from the output back to the first layer. An interesting application of this “interconnected” form to a problem in nonlinear filtering for processes defined by stochastic difference equations is in [170]. Interesting stochastic approximation algorithms motivated by the training of neural networks are in [7, 13].



## 2.3 State-Dependent Noise: A Motivational Example

Most of the noise processes that have appeared in the stochastic approximation algorithms up to this point in the book have been martingale differences, where  $E[Y_n|\theta_0, Y_i, i < n]$  depended on  $\theta_n$  but not otherwise random. More complicated noise processes are common occurrences. For example, suppose that the sequence of patterns in Example 3 of Section 1.1 is correlated or that the training data in the neural network example of Section 2 is correlated, but that the distributions do not depend on the stochastic approximation algorithm. Such noise processes are called *exogenous*. Example 4 of Section 1.1 provides a similar example, if the  $\{\chi^m\}$  in (1.1.20) are correlated, but determined purely by effects “external” to the algorithm. In many important applications, the evolution of the effective noise process depends more intimately on the iterate (also to be called the *state*), and there is a reasonably long-term “memory” in this dependence. The following adaptive “routing” example taken from [136] illustrates the point in a simple way. The adaptive queueing problem discussed in Section 5 and the other references listed there show that such models are of considerable interest. Indeed, they are current canonical forms used in the optimization of queueing and manufacturing systems where the cost criterion is an average performance over the infinite time interval. The time-varying parameter tracking problem in [150] (see also Chapter 3) is another example of a complicated model in stochastic approximation that can be treated with this type of noise model. The example and the specific assumptions given in this section are for motivational purposes only.

To proceed, let us consider the following example. Suppose that calls arrive at a switch randomly, but at discrete instants  $n = 1, 2, \dots$ . No more than a single call can arrive at a time and

$$P\{\text{arrival at time } n \mid \text{data to but not including time } n\} = \mu > 0.$$

The assumptions concerning single calls and discrete time make the formulation simpler, but the analogous models in continuous time are treated in essentially the same manner. To have a clear sequencing of the events, suppose that the calls are completed “just before” the discrete instants, so that if a call is completed at time  $n^-$ , then that circuit is available for use by a new call that arrives at time  $n$ . There are two possible routes for each call. The  $i$ th route has  $N_i$  lines and can handle  $N_i$  calls simultaneously. The sets of call lengths and interarrival times are mutually independent, and  $\lambda_i > 0$  is the probability that a call is completed at the  $(n + 1)$ st instant, given that it is in the system at time  $n$  and handled by route  $i$ , and the rest of the past data. The system is illustrated in Figure 3.1.

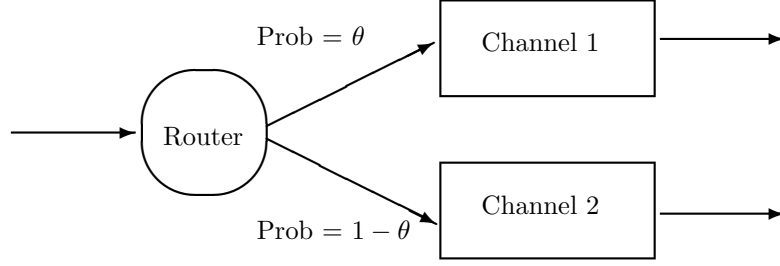


Figure 3.1. The routing system.

The routing law is “random,” and it is updated by a stochastic approximation procedure with constant step size  $\epsilon$ . Let  $\xi_n^\epsilon = (\xi_{n,1}^\epsilon, \xi_{n,2}^\epsilon)$  denote the occupancies of the two routes at time  $n$ . If a call arrives at time  $n + 1$  then it is sent first to route 1 with probability  $\theta_n^\epsilon$  and to route 2 with probability  $1 - \theta_n^\epsilon$ . If all lines of the selected route are occupied at that time, the call is redirected to the other route. If the alternative route is also full, the call is lost from the system. Let  $J_{n,i}^\epsilon$  be the indicator function of the event that a call arrives at time  $n + 1$ , is sent to route  $i$ , and is accepted there. Our updating rule for  $\theta_n^\epsilon$  is

$$\theta_{n+1}^\epsilon = \Pi_{[a,b]} [\theta_n^\epsilon + \epsilon_n Y_n^\epsilon] = \Pi_{[a,b]} [\theta_n^\epsilon + \epsilon ((1 - \theta_n^\epsilon) J_{n,1}^\epsilon - \theta_n^\epsilon J_{n,2}^\epsilon)], \quad (3.1)$$

where  $0 < a < b < 1$  are truncation levels and  $\Pi_{[a,b]}$  denotes the truncation. One could also use decreasing step sizes  $\epsilon_n \rightarrow 0$ , but in such problems one normally wishes to allow tracking of the optimal value of  $\theta$  when the statistics change, in which case we cannot have  $\epsilon_n \rightarrow 0$ .

An examination of the right-hand side  $\bar{g}(\theta)$  of the mean ODE in (3.4) shows that  $\bar{g}(\theta) = 0$  is equivalent to (under stationarity) the statement that there is equal probability that each route is full at the time of a call’s arrival. Thus, the adaptive algorithm (3.3) given below serves to equate the probabilities of being full in the long run. This might be called a “fairness to the user” criterion. Many other design goals can be realized with appropriate forms of the algorithm.

The occupancies  $\xi_n^\epsilon$  (and the random acceptances and routing choices) determine the effective noise in the system, and the evolution of  $\{\xi_n^\epsilon\}$  depends on  $\{\theta_n^\epsilon\}$  in a complicated way and with significant memory. The dependence is of Markovian type in that

$$P \left\{ \xi_{n+1}^\epsilon = \tilde{\xi} \mid \xi_i^\epsilon, \theta_i^\epsilon, i \leq n \right\} = P \left\{ \xi_{n+1}^\epsilon = \tilde{\xi} \mid \theta_n^\epsilon, \xi_n^\epsilon \right\}. \quad (3.2)$$

Define  $v_i = (1 - \lambda_i)^{N_i}$ . If a call is assigned to route 1 at time  $n + 1$  and is not accepted there then (a) the call arrives at  $(n + 1)$ ; (b) it is assigned to route 1; (c) route 1 is full at time  $n$  and there are no departures on the

interval  $[n, n+1)$ . Thus, we have the conditional expectations

$$\begin{aligned} E[J_{n,1}^\epsilon | \theta_i^\epsilon, \xi_i^\epsilon, i \leq n] &= \mu \theta_n^\epsilon [1 - v_1 I_{\{\xi_{n,1}^\epsilon = N_1\}}] \equiv \gamma_1(\theta_n^\epsilon, \xi_n^\epsilon), \\ E[J_{n,2}^\epsilon | \theta_i^\epsilon, \xi_i^\epsilon, i \leq n] &= \mu (1 - \theta_n^\epsilon) [1 - v_2 I_{\{\xi_{n,2}^\epsilon = N_2\}}] \equiv \gamma_2(\theta_n^\epsilon, \xi_n^\epsilon). \end{aligned}$$

Define

$$\begin{aligned} g(\theta, \xi) &= (1 - \theta)\gamma_1(\theta, \xi) - \theta\gamma_2(\theta, \xi), \\ \delta M_n &= [(1 - \theta_n^\epsilon)J_{n,1}^\epsilon - \theta_n^\epsilon J_{n,2}^\epsilon] - g(\theta_n^\epsilon, \xi_n^\epsilon). \end{aligned}$$

Thus, for the  $Y_n$  defined in (3.1),

$$E[Y_n^\epsilon | \theta_i^\epsilon, \xi_i^\epsilon, i \leq n] = g(\theta_n^\epsilon, \xi_n^\epsilon).$$

Rewrite the algorithm as

$$\theta_{n+1}^\epsilon = \Pi_{[a,b]} [\theta_n^\epsilon + \epsilon (g(\theta_n^\epsilon, \xi_n^\epsilon) + \delta M_n)]. \quad (3.3)$$

The  $\delta M_n$  are martingale differences, but the conditional mean  $g(\theta_n^\epsilon, \xi_n^\epsilon)$  is not simple, since the time evolution of the sequence  $\{\xi_n^\epsilon\}$  depends heavily on the time evolution of the iterate sequence  $\{\theta_n^\epsilon\}$ . Nevertheless, as will be seen in Chapter 8, effective methods are available to show that the effects of the noise disappear in the limit as  $\epsilon \rightarrow 0$  and  $n \rightarrow \infty$ .

To study the asymptotic properties of  $\theta_n^\epsilon$ , it is useful to introduce the “fixed- $\theta$ ” process as follows. Let  $\{\xi_n(\theta)\}$  denote the Markov chain for the occupancy levels that would result if the parameter  $\theta_n^\epsilon$  were held constant at the value  $\theta$ . The  $\{\xi_n(\theta)\}$  is an ergodic Markov chain; we use  $E^\theta$  to denote the expectations of functionals with respect to its unique stationary measure.

If  $\epsilon$  is small, then  $\theta_n^\epsilon$  varies slowly and, loosely speaking, the “local” evolution of the  $g(\theta_n^\epsilon, \xi_n^\epsilon)$  can be treated as if the  $\theta_n^\epsilon$  were essentially constant. That is, for small  $\epsilon$  and  $\Delta$  and  $\theta_n^\epsilon = \theta$ , a law of large numbers suggests that we have approximately

$$\epsilon \sum_{i=n}^{n+\Delta/\epsilon} g(\theta_i^\epsilon, \xi_i^\epsilon) \approx \Delta E^\theta g(\theta, \xi_n(\theta)).$$

Then the mean ODE that characterizes the limit behavior for small  $\epsilon$  and large time is

$$\dot{\theta} = \bar{g}(\theta), \text{ for } \theta \in [a, b], \quad (3.4)$$

where

$$\bar{g}(\theta) = E^\theta g(\theta, \xi_n(\theta))$$

is a continuous function of  $\theta$ . This will be established in Chapter 8. If the solution of (3.4) tries to exit the interval  $[a, b]$ , then it is stopped on the boundary.

**Comment.** In this example, the ultimate goal of the algorithm is equating the stationary probabilities that the routes will be full on arrival of a call. Yet the iteration proceeds in real time using the history of the rejections for the updating, and each update uses a different value of the parameter  $\theta$ . So, the observations are not unbiased estimates of the “mean error function”  $\bar{g}(\theta)$  at any value of  $\theta$ . The process  $\{\xi_n\}$  is analogous to a sufficient statistic in that it encapsulates the effects of the past on the future performance of the algorithm. The heuristic discussion shows that because  $\theta_n$  varies slowly for small  $\epsilon$ , the stationary averages at the current parameter values are actually being estimated increasingly well as  $\epsilon$  decreases and time increases. This idea will be formalized in Chapters 8 and 9, in which the Markov property (3.2) will play an important role. An increasing number of applications of stochastic approximation concern long-term averages, and use observations on a single sample path taken at instants of time going to infinity, each observation being taken at a new parameter value. In these cases, the “effective memory” process  $\xi_n$  and their fixed- $\theta$  versions  $\xi_n(\theta)$  will play a crucial role. The examples in the next two sections illustrate various applications of the state-dependent noise model to the optimization of discrete event systems.

## 2.4 Learning Optimal Controls and Value Functions

In this section, three classes of learning algorithms for controlled processes will be discussed. They concern problems of estimating a cost function or an optimal control when analytical methods cannot be used effectively. They are all based on simulation. Such methods are called for if the transition probabilities are not known, but the processes can be simulated or physical samples observed under chosen controls. Even if the transition probabilities are known, the problem of analytic computation might be too hard, and one might seek an approximation method that exploits the possibilities of simulation. The intention in each case is to introduce the subject by focusing on one basic approach. The entire area is under active study, and much remains to be done to get broadly useful algorithms. There is a growing literature on these topics and a recent survey and additional references are in [208].

The problem of Subsection 4.1 concerns finding an optimal feedback control and value function for a Markov chain by the method of  $Q$ -learning. It involves martingale difference noise. If the number of (state, control) pairs is large, as it usually is, then this procedure will converge slowly.

The problem of Subsection 4.2 concerns the approximation of a value function for a fixed control, and involves correlated “exogenous” noise. Such approximation methods for the value function can be used as the basis of

recursive algorithms which approximate an optimal control by following the logic of the approximation in policy space procedure [22] or some other gradient-type procedure to get a sequence of controls which (hopefully) approximate the optimal one. Consider the following sequential approach. For a given control, get a good approximation to the value function via the method of Subsection 4.2. Then, assuming that this approximation is the true value function, apply the approximation in policy space method to get an improved control, then approximate the value function under the new control by a method such as that in Subsection 4.2, etc. Such methods go by the generic name of *actor-critic* [208]. Obviously, one cannot actually compute the approximation of Subsection 4.2 of the value function for each control. References [112, 113] concern a two-time-scale stochastic approximation approach to this problem. Here, the control is parameterized and the iteration on the control occurs relatively infrequently, which allows the algorithm for computing the approximation to the value function to nearly converge between control updates. The method essentially approximates the cost function for the current parameter value via a procedure like that in Subsection 4.2, and then uses that approximation to get the next value by a scheme such as approximation in policy space. One expects such two-time-scale methods to be slow, unless a lot of attention is given to the details concerning the time between control updates and the nature of the update itself. The paper [221] is concerned with various efficient adaptations of the actor-critic method and successfully illustrates the ideas on several “balancing” type problems.

The problem of Subsection 4.3 concerns the optimization of a stationary cost for a controlled Markov chain, and involves Markov state-dependent noise of the type introduced in Section 3. The paper [174] concerns getting the optimal parameter value for a parameterized control for the average cost per unit time problem. It uses the behavior of the process between returns to some fixed state to estimate the derivatives of the stationary cost with respect to the parameter. The control parameter is updated at the start of each such cycle. If the state space is large, then the return times and the variances of the estimators will generally be large as well. The paper [172] applies the approximation methods to a problem in communications and presents data showing the convergence. The book [232] contains an interesting historical perspective on the origin of the various methods.

### 2.4.1 *Q-Learning*

This example concerns the adaptive optimization of a control system governed by a Markov chain, a problem that is of increasing interest in robotic learning and artificial intelligence. Let  $p(i, j|d)$  be transition probabilities for a controlled finite-state Markov chain, where  $d$  denotes the control variable and it takes values in a finite set  $U(i)$  for each  $i$ . Let  $u_n$  denote the control action taken at time  $n$  and  $\psi_n$  the state of the chain at time  $n$ . Let

$\mathcal{F}_n$  denote the minimal  $\sigma$ -algebra generated by  $\{\psi_j, u_j, j < n, \psi_n\}$ . The control action taken at each time  $n$  might simply be a function of the state  $\psi_n$  at that time. More generally, it can be a random function of the past history. In the latter case, it is assumed to be *admissible* in the sense that it is chosen according to a probability law depending on the history up to the present. That is, it is selected by specifying the conditional probabilities

$$P\{u_n = d | \mathcal{F}_n\}.$$

We use  $\pi$  to denote an admissible control policy.

Given initial state  $i$ , admissible control policy  $\pi$ , and discount factor  $0 < \beta < 1$ , the cost criterion is

$$W(i, \pi) = E_i^\pi \sum_{n=0}^{\infty} \beta^n c_n,$$

where  $c_n$  is the real-valued cost realized at time  $n$  and  $E_i^\pi$  denotes the expectation under initial state  $i$  and policy  $\pi$ . It is supposed that the current cost depends only on the current state and control action in that for all  $c$

$$P\{c_n = c | \psi_j, u_j, j < n, \psi_n = i, u_n = d\} = P\{c_n = c | \psi_n = i, u_n = d\},$$

where the right side does not depend on  $n$ . Henceforth, for notational convenience,  $c_{n,id}$  is used to denote the random cost realized at time  $n$  when (state, control) =  $(i, d)$ . Conditioned on the current state being  $i$  and the current control action being  $d$ , we suppose that the current cost  $c_{n,id}$  has a finite mean  $\bar{c}_{id}$  and a uniformly bounded variance.

Then there is an optimal control that depends only on the current state. Define  $V(i) = \inf_\pi W(i, \pi)$ . Then the Bellman (dynamic programming) equation for the minimal cost is

$$V(i) = \min_{d \in U(i)} \left[ \bar{c}_{id} + \beta \sum_j p(i, j | d) V(j) \right]. \quad (4.1)$$

The minimizing controls in (4.1) are an optimal policy. In other words, when in state  $i$ , the optimal control is the (or any, if nonunique) minimizer in (4.1) (see [18, 198]).

The so-called  $Q$ -learning algorithm, to be described next, was motivated by the problem of adaptive optimization of the Markov chain control problem, when the transition probabilities are not known, but the system can be simulated or observed under any choice of control actions. The idea originated in [251], and proofs of convergence are in [234, 252]; the former is quite general; see also [5]. The analogous algorithm for the average cost per unit time problem is discussed in [1, 173].

The algorithm involves recursively estimating the so-called  $Q$ -functions  $\bar{Q}_{id}$ , where  $\bar{Q}_{id}$  is the cost given that we start at state  $i$ , action  $d \in U(i)$

is taken, and then the optimal policy is used henceforth. This is to be distinguished from the definition of  $V(i)$ , which is the value given that we start at state  $i$  and use the optimal policy at all times. One must estimate the quantity  $\bar{Q}_{id}$  for each state and control action, but this is easier than estimating the transition probabilities for each  $i, j, d$ . Since  $\min_{d \in U(i)} \bar{Q}_{id} = V(i)$ , an application of the principle of optimality of dynamic programming yields the relationship

$$\bar{Q}_{id} = \bar{c}_{id} + \beta \sum_j p(i, j|d) \min_{v \in U(j)} \bar{Q}_{jv}. \quad (4.2)$$

A stochastic approximation algorithm is to be used to recursively estimate the  $Q$ -values. The dimension of the problem is thus the total number of (state, action) pairs. The algorithm can use either a single simulated process, an actual physical control process, or multiple simultaneous processes. For simplicity, suppose that only a single control process is simulated. At each step, one observes the value  $i$  of the current state of the Markov chain and then selects a control action  $d$ . Then the value of the state at the next time is observed. At that time, the  $Q$ -value for only that particular (state, action) pair  $(i, d)$  is updated. Thus the algorithm is asynchronous.

Suppose that the pair  $(i, d)$  occurred at time  $n$ , and the next state  $\psi_{n+1}$  is observed. Let  $Q_n = \{Q_{n,xd}; x, d : d \in U(x)\}$  be the current estimator of  $\bar{Q}$ . Then we update as

$$Q_{n+1,id} = Q_{n,id} + \tilde{\epsilon}_{n,id} \left[ c_{n,id} + \beta \min_{v \in U(\psi_{n+1})} Q_{n,\psi_{n+1}v} - Q_{n,id} \right], \quad (4.3)$$

where  $\tilde{\epsilon}_{n,id}$  is the step size and is chosen as follows. Let  $\epsilon_n$  be a sequence of positive real numbers such that  $\sum \epsilon_n = \infty$ , and let  $w_{id}$  be positive real numbers. Define  $\epsilon_{n,id} = w_{id}\epsilon_n$ . The step size used at the current iteration depends on the number of times that the particular current (state, action) pair has been seen before. If the current (state, action) pair is  $(i, d)$ , and this pair has been seen  $(k-1)$  times previously in the simulation, then the step size used in the current updating of the estimate of  $\bar{Q}_{id}$  is  $\tilde{\epsilon}_{n,id} = \epsilon_{k,id}$ . More formally, let  $k(i, d, n)$  denote the number of times the pair  $(i, d)$  has been seen before or at time  $n$ , and define  $\tilde{\epsilon}_{n,id} = \epsilon_{k(i,d,n),id}$ . If  $(i, d)$  is the (state, action) pair seen at time  $n$ , then for  $(j, v) \neq (i, d)$ , there is no update at  $n$  and we have  $Q_{n+1,jv} = Q_{n,jv}$ . In Chapter 12, where the proof of convergence will be given, the values of the estimators of the  $\bar{Q}_{id}$  will be truncated at  $\pm B$  for some suitably large  $B$ . As will be seen, this simplifies the proofs and causes no loss of generality. In fact  $|\bar{Q}_{id}| \leq \max_{i,d} |\bar{c}_{i,d}|/(1-\beta)$ . Constant step sizes can also be used, and this is done in the convergence proof in Chapter 12. The proof for the decreasing step size case is virtually the same.

For a vector  $Q = \{Q_{id}; i, d : d \in U(i)\}$ , define

$$T_{id}(Q) = E \left[ c_{n,id} + \beta \min_{v \in U(\psi_{n+1})} Q_{\psi_{n+1}v} \middle| \mathcal{F}_n, u_n = d, \psi_n = i \right].$$

By the Markov property,

$$T_{id}(Q) = \bar{c}_{i,d} + \beta \sum_j p(i, j|d) \min_{v \in U(j)} Q_{jv}.$$

With the definition of the “noise”

$$\delta M_n = \left[ c_{n,id} + \beta \min_{v \in U(\psi_{n+1})} Q_{n,\psi_{n+1}v} \right] - T_{id}(Q_n),$$

we can write

$$Q_{n+1,id} = Q_{n,id} + \tilde{\epsilon}_{n,id} [T_{id}(Q_n) - Q_{n,id} + \delta M_n]. \quad (4.4)$$

Note that  $E[\delta M_n | \mathcal{F}_n, u_n = d] = 0$ , and that the conditional variance  $E[\delta M_n^2 | \mathcal{F}_n, u_n = d]$  is bounded uniformly in  $n, \omega$ .

The map  $T(\cdot)$  is a Lipschitz continuous contraction with respect to the sup norm and  $\bar{Q}$  is the unique fixed point [18, 190]. Suppose that each (state, action) pair is visited infinitely often with probability one. Then one can show that  $Q_n$  converges to  $\bar{Q}$  with probability one. Let  $N_{id}(n, n+m)$  be the number of times that the pair  $(i, d)$  is returned to in the time interval  $[n, n+m]$ . Suppose that for any pairs  $(i, d)$  and  $(j, v)$ , the ratios

$$N_{id}(n, n+m)/N_{jv}(n, n+m)$$

are bounded away from both zero and infinity as  $n$  and  $m$  go to infinity. Then, it is shown in Chapter 12 that for the constrained algorithm, there is a diagonal matrix  $D(t)$  whose diagonal components  $D_{id}(t)$  are bounded and strictly positive such that the mean ODE that characterizes the limit points is

$$\dot{Q}_{id} = D_{id}(t) (T_{id}(Q) - Q_{id}) + z_{id}, \quad (4.5)$$

where the  $z_{id}(t)$  term is zero if  $-B < Q_{id}(t) < B$ ; otherwise it serves only to keep  $Q_{id}(\cdot)$  from leaving the interval  $[-B, B]$  if the dynamics try to force it out. If  $B$  is large enough, then the limit point is  $\bar{Q}$ .

### 2.4.2 Approximating a Value Function

In this subsection, we will illustrate a class of algorithms that is of current interest for recursively approximating a discounted cost function for a stochastic process, as a function of the initial condition. If the process is controlled, then the control is fixed. The class of approximating algorithms is known as  $TD(\lambda)$ ,  $0 < \lambda \leq 1$  [235] ( $TD$  standing for temporal difference).



We will concentrate on the class  $TD(1)$ , and discuss some of the basic ideas. The classes  $\lambda < 1$  will only be commented on briefly.

We will start with a Markov model. This Markov assumption can be weakened considerably, which is an advantage of the approach. Let  $\{X_n\}$  denote a Markov chain with values in some compact topological space, a time-invariant transition function, and a unique invariant measure  $\mu(\cdot)$ . Let  $E_\mu$  denote the expectation with respect to the measure of the stationary process. For a bounded and continuous real-valued function  $k(\cdot)$ , the cost is

$$W(x) = E_x \sum_{n=0}^{\infty} \beta^n k(X_n, X_{n+1}), \quad 0 < \beta < 1.$$

We wish to approximate  $W(x)$  by a finite sum  $\widetilde{W}(x, w) = \sum_{i=1}^p w_i \phi_i(x) = w' \phi(x)$ , where the real-valued functions  $\phi_i(\cdot)$  are bounded and continuous and the optimal weight  $w$  is to be determined by an adaptive algorithm. The approximation is in the sense of

$$\min_w e(w), \quad \text{where } e(w) = E_\mu [W(X_0) - w' \phi(X_0)]^2 / 2, \quad (4.6)$$

where  $X_0$  denotes the canonical “stationary random variable.” It is always supposed that  $E_\mu \phi(X_0) \phi'(X_0)$  is positive definite. A simulation based recursive algorithm of the gradient descent type will be used to get the optimal value of  $w$ . Whenever possible, one constructs such adaptive algorithms so that they are of the gradient descent type. Such simulation-based adaptive methods are motivated by the difficulty of computing cost functions for large or continuous state spaces.

Following the usual method [235], define the sequence

$$C_{n+1} = \beta \lambda C_n + \phi(X_{n+1}), \quad 0 \leq \lambda \leq 1, \quad (4.7)$$

and the difference

$$d_n = k(X_n, X_{n+1}) + \beta \widetilde{W}(X_{n+1}, w_n) - \widetilde{W}(X_n, w_n). \quad (4.8)$$

Until further notice, set  $\lambda = 1$ . The difference  $d_n$  is a “noisy” estimate of an approximation error in that

$$W(X_n) - E_{X_n} \beta W(X_{n+1}) = E_{X_n} k(X_n, X_{n+1}).$$

The adaptive algorithm is (which defines  $Y_n$ )

$$w_{n+1} = w_n + \epsilon_n d_n C_n = w_n + \epsilon_n Y_n, \quad (4.9)$$

where  $\epsilon_n > 0, \sum_n \epsilon_n = \infty$ . We will now heuristically justify the gradient descent assertion and motivate the convergence by deriving the mean ODE.

For large  $n$ , (4.9) can be well approximated by

$$\begin{aligned} w_{n+1} = w_n + \epsilon_n \left[ k(X_n, X_{n+1}) + \beta \widetilde{W}(X_{n+1}, w_n) - \widetilde{W}(X_n, w_n) \right] \\ \times \sum_{l=-\infty}^n [\lambda \beta]^{n-l} \phi(X_l). \end{aligned} \quad (4.10)$$

The mean ODE is determined by the stationary mean value of the terms in (4.10), with  $w_n$  held fixed. Suppose that the iterates are bounded. Then, by Theorem 8.2.1 or 8.2.2 and the uniqueness of the stationary measure, the mean ODE is

$$\begin{aligned} \dot{w} &= \bar{g}(w) \\ &= E_\mu \left[ k(X_n, X_{n+1}) + \beta \widetilde{W}(X_{n+1}, w) - \widetilde{W}(X_n, w) \right] \sum_{l=-\infty}^n [\lambda \beta]^{n-l} \phi(X_l). \end{aligned} \quad (4.11)$$

For the probability one convergence case, use Theorem 6.1.1.

To see why this limit form is correct, first note that  $w_n$  varies slowly for large  $n$ . Supposing that  $w_n$  is fixed at  $w$  for many iterates, the law of large numbers for Markov chains implies that the sample average of a large number of the terms  $Y_n$  in (4.10) is approximated by its stationary mean. Now, fixing  $w_n = w$  and computing the stationary expectation of the terms in the coefficient of  $\epsilon_n$  on right side of (4.10) yields

$$\begin{aligned} E_\mu \left[ \beta \widetilde{W}(X_{n+1}, w) - \widetilde{W}(X_n, w) \right] \sum_{l=-\infty}^n [\lambda \beta]^{n-l} \phi(X_l) \\ = -[w' E \phi(X_n)] \phi(X_n). \end{aligned} \quad (4.12)$$

Since  $k(\cdot)$  is real-valued,

$$E_\mu k(X_n, X_{n+1}) \sum_{l=-\infty}^n [\lambda \beta]^{n-l} \phi(X_l) = \sum_{l=0}^{\infty} [\lambda \beta]^l E_\mu \phi(X_0) E_{X_0} k(X_l, X_{l+1}). \quad (4.13)$$

Thus, for  $\lambda = 1$ , we see that the stationary mean of  $Y_n$ , with  $w_n$  fixed at  $w$ , is

$$\text{gradient}_w e(w) = -E_\mu \left[ E_{X_0} \sum_{n=0}^{\infty} \beta^n k(X_n, X_{n+1}) - w' \phi(X_0) \right] \phi(X_0). \quad (4.14)$$

Hence the mean ODE is

$$\dot{w} = -\text{gradient}_w e(w) = \bar{g}(w), \quad (4.15)$$

and the algorithm is of the gradient descent type. If there is a concern that the iterates might not be bounded, then use the constrained algorithm

$$w_{n+1} = \Pi_H [w_n + \epsilon_n d_n C_n],$$

where  $H$  is a constraint set, perhaps a hyperrectangle. The mean ODE is just the projected form of (4.11). If  $H$  is a hyperrectangle, and the optimal point is inside  $H$ , then the solution to the ODE converges to it; otherwise, it converges to the closest point in  $H$ .

If the algorithm is not constrained, then the gradient descent characterization can be used to prove stability of the algorithm by using a perturbed Liapunov function, based on the Liapunov function  $e(w)$  for (4.15). This is covered by Theorem 6.7.3.

**An alternative formulation: less memory.** In lieu of (4.6), consider the problem

$$\min e_0(w), \quad \text{where } e_0(w) = E_\mu [E_{X_0} k(X_0, X_1) - w' \phi(X_0)]^2 / 2. \quad (4.16)$$

Here the cost to be approximated is simply  $E_{X_0} k(X_0, X_1)$ , which takes only one step of the future into account. A convergent recursive algorithm for computing the minimizer is

$$w_{n+1} = w_n + \epsilon_n [k(X_n, X_{n+1}) - w'_n \phi(X_n)] \phi(X_n), \quad (4.17)$$

which is also of the gradient descent form, since (with  $w_n$  fixed at  $w$ ) the stationary mean of the right side of (4.17) is just the gradient of  $e_0(w)$ .

One can interpolate between the criteria (4.6) and (4.16), by constructing minimization problems where the cost takes more and more of the future into account in a discounted way. This is a motivation for the algorithm  $TD(\lambda)$ ,  $\lambda < 1$ ; see [232, 235].

**Comments and extensions.** The evaluations of the mean values in (4.12) and (4.13) involve a lot of cancellations, using the fact that the stationary means of various different random variables that appeared are of opposite sign. Although many of the stationary means cancel, the associated random variables do not. This suggests a high noise level and slow convergence. The algorithm (4.9) could be constrained: For example, we could restrict  $w_{n,i}$  to a finite interval  $[a_i, b_i]$  or subject to other constraints. The time invariance of the transition function could be replaced by periodicity.

For a particularly useful extension, the chain needs to be only partly observable. For example, let  $X_n = (\hat{X}_n, \tilde{X}_n)$ , where only  $\hat{X}_n$  together with the costs  $k(X_n, X_{n+1})$  are observable. Then, with  $\phi(\hat{X}_n)$  used in (4.7) in lieu of  $\phi(X_n)$ , the algorithm (4.9) finds the minimizer of  $E_\mu [W(X_0) - w' \phi(\hat{X})]^2$ . The process need not be Markov, provided that some stationarity and mixing conditions hold. Then  $\mu$  is replaced by the steady state probability and the  $E_x$  in the definition of  $W(x)$  is replaced by conditioning on the data to the initial time. Such robustness is very useful, since “practical” processes often have “hidden states” or are not Markov. This is also covered by either Theorem 8.2.1 or 8.2.2 for the weak convergence case and Theorem 6.1.1 for the probability one convergence case.

### 2.4.3 Parametric Optimization of a Markov Chain Control Problem

**Problem formulation.** In this subsection, we are concerned with a controlled Markov chain and average cost per unit time problem, where the control is parameterized. We seek the optimal parameter value and do not use value function approximations of the type of Subsection 4.2. The approach is based on an estimate of the derivative of the invariant measure with respect to the parameter. This is an example of an optimization problem over an infinite time interval. The procedure attempts to approximate the gradient of the stationary cost with respect to the parameter. It is not a priori obvious how to estimate this gradient when the times between updates of the parameter must be bounded. We will exploit a useful representation of this derivative. The problem of minimizing a stationary cost is continued in the next section, for a queueing model, where an analog of the pathwise derivative of Example 4 of Subsection 1.1.4 will be used.

The process is a finite-state Markov chain  $\{X_n\}$  with time-invariant and known transition probabilities  $p(x, y|\theta)$  that depend continuously and differentially on a parameter  $\theta$  that takes values in some compact set. We will suppose that  $\theta$  is real-valued and confined to some interval  $[a, b]$ . The general vector-valued case is treated simply by using the same type of estimate for each component of  $\theta$ . Write  $p_\theta(x, y|\theta)$  for the  $\theta$ -derivative. Let the chain be ergodic for each  $\theta$ , and denote the unique invariant measure by  $\mu(\theta)$ . Let  $E_{\mu(\theta)}$  denote expectation under the stationary probability and  $E_x^\theta$  the expectation, given parameter value  $\theta$  and initial condition  $x$ . The objective is to obtain

$$\min_{\theta} e(\theta), \quad e(\theta) = E_{\mu(\theta)} k(X_0, X_1, \theta) = \sum_{x, y} \mu(x, \theta) p(x, y|\theta) k(x, y, \theta), \quad (4.18)$$

where  $k(x, y, \cdot)$  is continuously differentiable in  $\theta$  for each value of  $x$  and  $y$ , and  $\mu(x, \theta)$  is the stationary probability of the point  $x$ . Our aim is the illustration of a useful general approach to optimization of an average cost per unit time criterion. The best approach is not clear at this time. But any algorithm must have some way of estimating the derivative of the stationary cost.

The invariant measures are hard to compute, so the minimization is to be done via simulation. Suppose that we try to minimize (4.18) by formally differentiating with respect to  $\theta$ , and use the formal derivative as the basis of a gradient procedure. The gradient of  $e(\cdot)$  at  $\theta$  is  $T_1(\theta) + T_2(\theta) + T_3(\theta)$ , where (the subscript  $\theta$  denotes derivative with respect to  $\theta$ )

$$T_1(\theta) = \sum_{x, y} \mu(x, \theta) p(x, y|\theta) k_\theta(x, y, \theta),$$

$$T_2(\theta) = \sum_{x,y} \mu(x, \theta) p_\theta(x, y | \theta) k(x, y, \theta),$$

$$T_3(\theta) = \sum_{x,y} \mu_\theta(x, \theta) p(x, y | \theta) k(x, y, \theta).$$

For a gradient descent procedure the mean ODE would be

$$\dot{\theta} = -[T_1(\theta) + T_2(\theta) + T_3(\theta)]. \quad (4.19)$$

Write the stochastic algorithm as  $\theta_{n+1} = \theta_n + \epsilon_n Y_n$ , where  $Y_n$  is to be determined. Then, if  $\theta_n$  is fixed at an arbitrary value  $\theta$ , we need at least that the long term average of the  $Y_n$  is the right side of (4.19). The actual stochastic algorithm will be constructed with this in mind. If  $\theta_n$  is held fixed at  $\theta$ , then  $T_1(\theta)$  is easy to estimate since, by the law of large numbers for a Markov chain, it is just the asymptotic average of the samples  $k_\theta(X_n, X_{n+1}, \theta)$ .

If  $k(x, y, \theta)$  does not depend on the second variable  $y$ , then  $T_2(\theta)$  does not appear. A naive approach to approximating  $T_2(\theta)$  for the algorithm would be to simulate  $p_\theta(X_n, X_{n+1} | \theta_n) k(X_n, X_{n+1}, \theta_n)$ . If  $\theta_n \approx \theta$ , then by the law of large numbers for Markov chains, the sample average of many such terms will be close to the stationary expectation

$$\sum_{x,y} \mu(x, \theta) p(x, y | \theta) p_\theta(x, y | \theta) k(x, y, \theta).$$

But this is not  $T_2(\theta)$  owing to the presence of the term  $p(x, y | \theta)$ . Thus, the correct form to simulate is  $L(X_n, X_{n+1}, \theta_n) k(X_n, X_{n+1}, \theta_n)$ , where  $L(x, y, \theta) = p_\theta(x, y | \theta) / p(x, y | \theta)$  (which are assumed to be bounded). The stationary average of  $L(X_n, X_{n+1}, \theta) k(X_n, X_{n+1}, \theta)$  is  $T_2(\theta)$ , as desired.

The term  $T_3(\theta)$  cannot normally be computed explicitly and even its existence needs to be established for more general chains. A useful method of approximation by simulation is not at all obvious. The existence, characterization, and approximation, of this derivative were the subjects of [136, 148, 242, 262]. The method to be employed will be based on the results of [136], which covers the particular case of interest here. For a real-valued function  $q(\cdot)$  of the chain and  $\theta$ , define the stationary mean  $\lambda_q(\theta) = \sum_x \mu(x, \theta) q(x, \theta)$ . Then a useful representation of the derivative of  $\lambda_q(\cdot)$  with respect to  $\theta$  is: (see [136, Theorem 3])

$$\frac{d}{d\theta} \lambda_q(\theta) = \sum_{n=0}^{\infty} \mu'(\theta) P_\theta(\theta) P^n(\theta) [Q(\theta) - \lambda_q(\theta) \mathbf{e}], \quad (4.20)$$

where  $\mathbf{e}$  is the vector whose components are all unity and  $Q(\theta)$  denotes the vector  $\{q(x, \theta)\}$ . Note that the component of (4.20) that involves  $\lambda_q(\theta)$  is zero. The sum (4.20) converges since  $P^n(\theta) \rightarrow 0$  at a geometric rate.

In order to use the representation (4.20) as a basis for estimating  $T_3(\theta)$ , we need to simulate a sequence whose asymptotic expectation is (4.20).

Define  $q(x, \theta) = E_x^\theta k(x, X_1, \theta)$ . Before stating the algorithm, let us note that for  $n \geq l$ ,

$$\begin{aligned} & E_x^\theta q(X_{n+1}, \theta) L(X_l, X_{l+1}, \theta) \\ &= \sum_{x_l} p^{(l)}(x, x_l | \theta) \sum_{x_{l+1}} p(x_l, x_{l+1} | \theta) L(x_l, x_{l+1}, \theta) \\ & \quad \times \sum_w p^{(n-l)}(x_{l+1}, w | \theta) q(w, \theta), \end{aligned} \quad (4.21)$$

where  $p^{(n)}(\cdot)$  denotes the  $n$ -step transition probability. Averaging (4.21) with respect to the invariant measure yields

$$\sum_{x, y, w} \mu(x, \theta) p_\theta(x, y | \theta) p^{(n-l)}(y, w | \theta) q(w, \theta), \quad (4.22)$$

which is the  $(n-l)$ th summand of (4.20).

The above comments suggest one way of approximating (4.20) and a possible algorithm. Let  $0 < \beta < 1$ , but be close to unity. Consider the algorithm for approximating the optimal value of  $\theta$ :

$$\begin{aligned} \theta_{n+1} &= \theta_n - \epsilon_n k_\theta(X_n, X_{n+1}, \theta_n) \\ & \quad - \epsilon_n L(X_n, X_{n+1}, \theta_n) k(X_n, X_{n+1}, \theta_n) \\ & \quad - \epsilon_n (k(X_n, X_{n+1}, \theta_n) - \lambda_n) C_n, \end{aligned} \quad (4.23)$$

$$\begin{aligned} \lambda_{n+1} &= \lambda_n + \epsilon'_n [k(X_n, X_{n+1}, \theta_n) - \lambda_n], \\ C_{n+1} &= \beta C_n + L(X_n, X_{n+1}, \theta_n), \end{aligned}$$

where we use  $\epsilon'_n = q\epsilon_n$  for some  $q > 0$ . The update is done when  $X_{n+1}$  is observed.

**Comments on the algorithm (4.23).** The variable  $\lambda_n$  is intended to be an estimate of the optimal stationary cost. For large  $n$ ,  $\theta_n$  varies slowly. Suppose that it is fixed at a value  $\theta$ . Then  $\lambda_n$  will converge to the stationary value  $E_{\mu(\theta)} k(X_0, X_1, \theta)$ . This suggests that the  $\lambda_n$  in the first line of (4.23) would not affect the mean values, asymptotically, since  $E_{\mu(\theta)} L(X_0, X_1, \theta) = 0$ . However, the “centering” role of  $\lambda_n$  in (4.23) is important in that it serves to reduce the variance of the estimates. The first coefficient of  $\epsilon_n$  in the first equation in (4.23) is used to estimate  $T_1(\theta)$ , the second to estimate  $T_2(\theta)$ , and the third to estimate  $T_3(\theta)$ . Using (4.21) and the “discounted” definition of  $C_n$ , assuming that  $\theta_n \approx \theta$  and  $\lambda_n \approx \lambda_q(\theta)$ , the stationary average of the third term is a discounted form of (4.20), namely,

$$\sum_{n=0}^{\infty} \beta^n \mu'(\theta) P_\theta(\theta) P^n(\theta) [Q(\theta) - \lambda_q(\theta) \mathbf{e}].$$

Thus for  $\beta$  close to unity, we have an approximation to (4.20) and  $T_3(\theta)$ . The noise in the algorithm (4.23) is state-dependent in the sense introduced in Section 3. For the constrained form of the algorithm any of the methods in the state-dependent-noise Section 8.4 can be used to justify the mean ODE and the convergence to a point close to the optimum for  $\beta$  close to unity.

**Notes: Extension to diffusions or discretized diffusions.** Representations analogous to (4.20) exist for Markov processes on more general state spaces [242], and this can be used as the basis of a recursive algorithm, just as (4.20) was. An alternative approach, based on Girsanov transformation techniques, is in [262]. It uses estimates computed over finite intervals  $[n\Delta, n\Delta + \Delta)$ . Reference [262] contains an extensive discussion of derivative estimators for jump-diffusions, where the drift function is subject to control. The method also works for reflected jump diffusions if the reflection direction and jumps are not controlled. In such “continuous state space” problems, one must usually approximate in the simulations, since the actual continuous-time trajectory cannot usually be simulated. The reference and [148] discusses discrete-time and Markov chain approximations and show that the derivative estimators will converge as the discretization parameter goes to zero.

## 2.5 Optimization of a GI/G/1 Queue

We now consider another approach to the optimization of a stationary cost function. The example concerns the optimization of the performance of a single server queue. The times of customer arrivals is a renewal process with bounded mean. For fixed  $\theta$ , let  $X_i(\theta)$  denote the time that the  $i$ th customer spends in the system, and let  $K(\theta)$  be a known bounded real-valued function with a continuous and bounded gradient. The service time distribution is controlled by a real-valued parameter  $\theta$ , which is chosen to minimize the sum of the average waiting time per customer and a cost associated with the use of  $\theta$ . This is the cost function

$$L(\theta) = \lim_N \frac{1}{N} \sum_{i=1}^N EX_i(\theta) + K(\theta) \equiv \widehat{L}(\theta) + K(\theta). \quad (5.1)$$

The aim is to get the minimizing  $\theta$  in the finite interval  $[a, b]$ . Generally, the values of  $\widehat{L}(\theta)$  are very hard to compute. A viable alternative to the direct optimization of the unavailable (5.1) uses stochastic approximation. One would observe the queue over a long time period, and use the observational data (which are the times of arrival, departure, and service for each customer, up to the present). The data would be used to estimate the derivative of the cost function with respect to  $\theta$  at the current value of

$\theta$ , yielding a stochastic approximation analog of the deterministic gradient descent procedure. We use “estimates of the derivative” in a loose sense, since each so-called estimate will be (perhaps strongly) biased. However, their cumulative effect yields the correct result.

This problem has attracted much attention (see [52, 78, 145, 160, 161], among others) and is typical of many current efforts to apply stochastic approximation to queueing and manufacturing systems and networks.

### 2.5.1 *Derivative Estimation and Infinitesimal Perturbation Analysis: A Brief Review*

The parameter  $\theta$  is assumed to be real-valued throughout this section. A basic issue is that we can only update the estimators at finite times, although the cost function is of the “ergodic” type. Before proceeding further, let us review Example 4 of Section 1.1, where we were able to compute the “pathwise derivative”  $\bar{U}^N(\theta)$ . The derivative is said to be *pathwise*, because for each  $\omega$ , the sample value  $\bar{U}^N(\theta, \omega)$  is the derivative of the sample value  $\bar{X}^N(\theta, \omega)$  with respect to  $\theta$ . Let  $\delta\theta$  be a “small” perturbation of  $\theta$ . To get the formula (1.1.21) for the derivative at  $\theta_n = \theta$ , one *implicitly* solves for  $\bar{X}^N(\theta)$  and  $\bar{X}^N(\theta + \delta\theta)$ , using the *same* vector of driving noise  $\bar{\chi}$ , computes the ratio  $[\bar{X}^N(\theta + \delta\theta) - \bar{X}^N(\theta)]/\delta\theta$ , and then lets  $\delta\theta$  go to zero. By the rules of calculus, this procedure is what yields the final explicit formula (1.1.21). For the method to work, it is essential that the driving forces  $\bar{\chi}$  be the same in the computations of  $\bar{X}^N(\theta)$  and the  $\bar{X}^N(\theta + \delta\theta)$  for all small  $\delta\theta$ . If explicit pathwise differentiation were not possible, then one might have to resort to a finite difference estimator, which suffers from the fact that its variance is inversely proportional to the square of the finite difference interval. As an aside, recall that under appropriate smoothness assumptions, finite differences with “common” random variables used for the simulations for the parameters  $\theta_n$  and  $\theta_n + \delta\theta$  are often almost as good as the pathwise derivative, as shown in the discussion connected with (1.2.6).

There is an analogy to this pathwise derivative approach that yields a very useful method for getting good estimates of derivatives of the desired functional of the queue. The approach for the queueing problem is not obvious, because the optimization is to be done via observations on one sample path (i.e., done “on line” using data from the physical process), and there are no “natural driving forces” analogous to the  $\bar{\chi}$  that would allow the computation of the path for a perturbed value of  $\theta$ . It turns out that one can construct suitable “driving forces” from the observations and then get a direct analog of what was done in Example 4 of Section 1.1 by a clever choice of the probability space. This is the subject of the important field *infinitesimal perturbation analysis*, initiated by Y.-C. Ho. The subject of IPA has rapidly grown, and there is a large amount of literature; see, for example, [84, 99, 101] and the references therein.

A few introductory comments on IPA will be made in the next paragraph,



and then the results needed for the queueing problem at hand will be cited. The reader is referred to the literature for further reading. See also [145, 262, 148] for other results concerning pathwise derivatives that are useful in stochastic approximation.

**Introduction to IPA: Computation of a pathwise derivative.** Let  $\theta$  be fixed until further notice. In this application, one observes a sample path of the physical process over a long time interval and seeks to construct an estimate of the derivative  $L_\theta(\theta)$  from the sample data. Suppose, as a pure thought experiment, that one could observe simultaneously paths of the process corresponding to parameter values  $\theta$  and  $\theta + \delta\theta$ , for all arbitrarily small  $\delta\theta$ , where the sequences  $\{X_j(\theta), X_j(\theta + \delta\theta), j < \infty\}$  are defined on the same sample space. Suppose also that the path derivative exists for almost all realizations. In other words, the limit

$$\lim_{\delta\theta \rightarrow 0} \frac{X_j(\theta + \delta\theta) - X_j(\theta)}{\delta\theta} = D_j$$

exists with probability one for each  $j$ . Suppose also that

$$\frac{\partial EX_j(\theta)}{\partial\theta} = E \frac{\partial X_j(\theta)}{\partial\theta} = ED_j. \quad (5.2)$$

Then  $D_j$  is an unbiased estimator of  $\partial EX_j(\theta)/\partial\theta$ .

The idea is to estimate  $\partial X_j(\theta)/\partial\theta$  at the given value of  $\theta$  from a *single* sample path, with the parameter value  $\theta$  used, analogously to what was done in Example 4 of Section 1.1. There are two substantial parts of the work. The first is to get the derivative of the sample path, and the second is to verify the validity of the interchange of differentiation and expectation in (5.2). The reward of success is that one has a useful estimator, a pathwise derivative, which is based only on the observed data and does not involve finite differences. In applications, each of these steps might involve a great deal of work. When these pathwise derivatives can be calculated, it is frequently found that their variance is smaller (often much smaller) than what one would obtain under alternative approaches. From the point of view of stochastic approximation, the smaller the variance of the estimator, the better the procedure. In view of these remarks, it is clear that IPA and related methods are important for applications to stochastic approximation.

Since we are only concerned with expectations, the precise nature of the probability space that is used to compute the pathwise derivative is unimportant to the final result. Any convenient space can be used, provided that the limit formulas depend only on the observations. We will now give a simple example to illustrate this, where we compute the sample  $\theta$ -derivative of a random variable  $T$  with parameterized distribution function  $F(\cdot|\theta)$ .

For illustrative purposes, suppose that the distribution functions corresponding to parameter values  $\theta$  and (for small  $\delta\theta$ )  $\theta + \delta\theta$  are strictly

monotone increasing. To compute a pathwise derivative, the same probability space must be used for all parameter values and this can be done in the following way. By the definition of a distribution function, the random variable  $\chi$  defined by  $\chi = F(T|\theta)$  has a uniform distribution on the interval  $[0, 1]$ , and under our assumption on strict monotonicity, the values of  $\chi$  and  $T$  can be computed uniquely one from the other and we can write  $T = F^{-1}(\chi|\theta)$ .

Now, with  $\chi$  defined by  $\chi = F(T|\theta)$ , we can define the random variable  $T(\theta + \delta\theta) = F^{-1}(\chi|\theta + \delta\theta)$ , which has the distribution  $F(\cdot|\theta + \delta\theta)$ . The random variable  $\chi$  serves as the “driving force,” common to all values of  $\delta\theta$ , and it is constructed directly from the observed value of  $T$ . Thus, if the inverse function  $F^{-1}(\cdot|\theta)$  is smooth enough in  $\theta$ , we can get the representation of the pathwise derivative

$$\lim_{\delta\theta \rightarrow 0} \frac{F^{-1}(\chi|\theta + \delta\theta) - F^{-1}(\chi|\theta)}{\delta\theta} \equiv D.$$

Consider the simplest example, where  $F(T|\theta) = 1 - e^{-T/\theta}$ , the exponential distribution with mean  $\theta$ . Then

$$F^{-1}(\chi|\theta) = -\theta \log(1 - \chi),$$

and the derivative is simply  $-\log(1 - \chi)$ .

In the problem under consideration, the derivative of the distribution of the sojourn time is not easily obtainable, but the values of the sojourn time of the  $i$ th customer depend on the service times of the customers that were in the queue and on the residual service time for the customer in service (if any) at the moment when that  $i$ th customer arrived. Let  $\{T_n\}$  denote the sequence of (assumed mutually independent) service times, each of which has the distribution  $F(\cdot|\theta)$ . Define  $\chi_n = F(T_n|\theta)$ . Then  $\chi_n$  are mutually independent and uniformly distributed on the unit interval, and they are independent of the interarrival intervals and the initial condition. The pathwise derivative  $\partial T_n(\theta)/\partial\theta$  can be used to get suitable estimators of the  $\theta$ -derivative of the mean customer sojourn times. The derivation is involved; we simply copy the estimators for our problem from the references, because we are more concerned with their use than their derivation.

### 2.5.2 The Derivative Estimate for the Queueing Problem

Some of the conditions that we will state will be used when the convergence is proved in Chapter 9. First, the sequence of service times will be defined in terms of “driving noises” that do not depend on  $\theta$ . Then the formula for the pathwise derivative at parameter value  $\theta$  will be stated and adapted for use in the stochastic approximation procedure. Let the parameterized service time distribution  $F(\cdot|\theta)$  be weakly continuous in  $\theta$ : That is for

each bounded and continuous real-valued function  $f(\cdot)$ ,  $\int f(x)F(dx|\theta)$  is continuous in  $\theta$ . Define the inverse function  $F^{-1}(\cdot|\theta)$  by

$$F^{-1}(\chi|\theta) = \inf\{\zeta : F(\zeta|\theta) \geq \chi\}, \chi \in [0, 1],$$

and, for each value of  $\chi$ , let the inverse have a  $\theta$ -derivative that is continuous in  $\theta$  (uniformly in  $\chi$ ). We denote the  $\theta$ -derivative by  $F_\theta^{-1}(\chi|\theta)$ . Let  $\{\zeta_i(\theta), i < \infty\}$  denote the sequence of service times, and define  $\chi_n(\theta) = F(\zeta_n(\theta)|\theta)$  and the derivative  $Z_n(\theta) = F_\theta^{-1}(\chi_n(\theta)|\theta)$ . Define the cost for the first  $m$  customers, initialized at an arbitrary initial queue occupancy  $x_0$  as

$$L_m(\theta, x_0) = \hat{L}_m(\theta, x_0) + K(\theta) = \frac{1}{m} \sum_{i=1}^m EX_i(\theta) + K(\theta). \quad (5.3)$$

Suppose that the supremum over  $\theta \in [a, b]$  of the mean service times is less than the mean interarrival time. Then the busy periods have finite mean length for each  $\theta \in [a, b]$ . Suppose that:

the distribution function of the interarrival times is continuous. (5.4)

Consider the estimator

$$\hat{Z}_m(\theta) = \frac{1}{m} \sum_{i=1}^m \sum_{j=v_i(\theta)}^i Z_j(\theta), \quad (5.5)$$

where  $v_i(\theta)$  is the index of the first arrival in the busy period in which customer  $i$  arrives. The *index* of a customer is defined to be  $n$  if that customer is the  $n$ th to arrive from time zero on. If the initial occupancy of the queue is zero, then (5.5) is the pathwise  $\theta$ -derivative of

$$\frac{1}{m} \sum_{i=1}^m X_i(\theta).$$

Under broad conditions, it is an unbiased estimator of the  $\theta$ -derivative of the mean value  $\hat{L}_m(\theta, x_0)$ , and for each initial condition

$$\hat{Z}_m(\theta) \rightarrow \hat{L}_\theta(\theta) \quad (5.6)$$

with probability one and in mean as  $m \rightarrow \infty$ . The proof of this fact is one of the major issues in IPA. Proofs under various conditions and further references are in [84, 161, 231].

Henceforth, in this section, to simplify notation and not to worry about the possibly separate indices for arrivals and departures, we suppose that the queue starts empty. The conditions and results are the same in general.

The reader should keep in mind that we assume that the service time distribution is known for each  $\theta$  of interest, and this will rarely be the case.

One needs to know that the derivative estimators are robust enough so that they either approximate the derivatives of the true distributions without serious error or their use will still lead to an improved system. Limited simulations have shown that one can often use simple approximations and still improve the system, but more work needs to be done on this issue. The references [148, 262] use a system model that is a parameterized stochastic differential equation and show that the pathwise derivatives of a “numerical” approximation converge to the pathwise derivatives of the original process as the approximation parameter goes to its limit.

**The stochastic approximation algorithm.** When we use stochastic approximation to minimize the cost (5.1) via the use of the IPA estimator, we update the parameter after the departure of each successive group of  $N$  customers. Other choices of updating times can be used, for example, update at the end of successive busy periods or at the end of the first busy period following the departure of the next  $N$  customers, or a random mixture of all of these methods, and so forth. Every reasonable choice of updating times has the same limit properties [145]. We use the “customer group number” instead of real time as the index of the stochastic approximation. For fixed  $\theta$ , define the estimator on the  $n$ th interval (this contains the departures  $[nN + 1, \dots, nN + N]$ )

$$\hat{Y}_n(\theta) = -\frac{1}{N} \sum_{i=nN+1}^{nN+N} \sum_{j=v_i(\theta)}^i Z_j(\theta). \quad (5.7)$$

Recall that  $v_{nN}(\theta)$  is the index of the first arrival in the busy period in which arrival (equivalently, departure)  $nN$  occurs. Then, by (5.6)

$$\lim_n \frac{1}{nN} \sum_{i=1}^{nN} \sum_{j=v_i(\theta)}^i Z_j(\theta) = \hat{L}_\theta(\theta), \quad (5.8)$$

with probability one and in expectation for each initial condition.

The step size in the algorithm will be a small constant  $\epsilon$ . Hence, the  $\theta_n^\epsilon$  depend on  $\epsilon$ , and so do the service times and queue lengths. In the physical system,  $\theta_0^\epsilon$  is used up to the time of departure of the  $N$ th customer, then  $\theta_1^\epsilon$  is used up to the time of departure of the  $2N$ th customer, and so forth. For the actual physical system with the time varying parameter, let  $\zeta_i^\epsilon$  denote the actual service time of the  $i$ th customer and  $Z_i^\epsilon$  the derivative of the inverse function at  $\zeta_i^\epsilon$ . Let  $v_i^\epsilon$  be the index of the first arrival in the busy period in which customer  $i$  arrives. To update  $\theta_n^\epsilon$ , we use the estimator

$$\hat{Y}_n^\epsilon = -\frac{1}{N} \sum_{i=nN+1}^{nN+N} \sum_{j=v_i^\epsilon}^i Z_j^\epsilon. \quad (5.9)$$

A reasonable stochastic approximation algorithm for updating  $\theta$  is

$$\theta_{n+1}^{\epsilon} = \Pi_{[a,b]} \left[ \theta_n^{\epsilon} + \epsilon \hat{Y}_n^{\epsilon} - \epsilon K_{\theta}(\theta_n^{\epsilon}) \right]. \quad (5.10)$$

One could use a decreasing sequence  $\epsilon_n \rightarrow 0$  in place of  $\epsilon$ , but in such applications one generally uses a small and constant value, which would allow tracking of slow changes in the parameter.

The statistical structure of  $\{\hat{Y}_n^{\epsilon}\}$  is complicated. We know something about the unbiasedness and consistency properties for large  $N$  when the parameter does not change, but  $\hat{Y}_n^{\epsilon}$  is not an unbiased estimator of the derivative of  $-\hat{L}(\cdot)$  at  $\theta = \theta_n$ . It might be close to an unbiased estimator for large  $N$ , but  $N$  should not and need not be too large in a practical algorithm. Letting  $N$  be too large simply slows down the rate at which the new data is absorbed into the overall averaging, and the convergence holds for any  $N$ . It is not *a priori* clear that the algorithm will converge to the correct limit. However, the general methods to be developed in Chapter 8 will allow us to prove (in Chapter 9) the convergence under reasonably weak conditions. Again, the key is the slow rate of change of  $\theta_n$  when  $\epsilon$  is small, which allows us to effectively combine many successive  $\hat{Y}_n^{\epsilon}$  with essentially the same parameter value. Indeed, if  $\theta_n$  varies slowly, as it would when  $\epsilon$  is small, then (5.8) suggests that for large  $\delta/\epsilon$  and small  $\epsilon$  and  $\delta$ , the sum of  $(\epsilon/\delta)\hat{Y}_i^{\epsilon}$  for  $i \in [n, n + \delta/\epsilon)$  would be a good estimator of the derivative at  $\theta = \theta_n^{\epsilon}$ . This will be formalized in Chapters 8 and 9. An important variation is the decentralized algorithm (Chapter 12), where there are several processors, each taking its own measurements, updating its own subset of parameters at intervals that are not necessarily synchronized between the processors, and passing information on the updates to other processors from time to time [145, 154, 239].

**Other examples using IPA.** Reference [241] contains a detailed discussion of stochastic approximation with various forms of IPA estimators, together with simulation data that both justifies the approach and illustrates the qualitative properties of the convergence.

There is a large literature on IPA-type estimators, sometimes in conjunction with a stochastic approximation-type procedure. From the point of view of stochastic approximation, the structure of many of these problems is similar, and the same proof (say those of Chapters 8 and 9) can be used. Typical applications are single queues, queueing networks, and various production scheduling problems. We only list a few references to give some flavor of the developments. In most cases, the results of this book simplify and extend the stochastic approximation parts of the development and allow the use of more flexible algorithms. Additionally, one can treat the multiclass problem [186], admission control [243], flow control in a closed network [244], routing in an open network [239], and routing in a network [100], as well as balancing the noise and bias and allocating total

computational budget in optimization of steady state simulation models with likelihood ratio, IPA, or finite difference estimators [162].

Reference [46] concerns the optimization of the performance of an unreliable machine. The machine has a capacity constraint and is used to produce several types of parts, each with a given demand rate. The control problem is the decision concerning what to produce at any given time. The decision rule is determined by thresholds in the space of available inventory, and these thresholds are to be optimized. An appropriate cost function is given, and IPA-type estimators for the desired pathwise derivatives are constructed. A related problem, where there are two unreliable machines working in tandem, is treated in [260, 272]. The formulation of the stochastic approximation part is dealt with by the more efficient methods of this book in [145]. Optimization of an inventory system is discussed in [86]. Reference [245] concerns the optimization of a high-speed communications network. Messages may appear at any node and are to be routed through the network to the appropriate destination. Traffic is managed by a “token” system. With this method, each message moving through the system must be accompanied by a token. When the message arrives at the destination, the token is sent back to some input node to be reused. The control parameters are the probabilities  $p_{ij}$  that a token arriving with a message at destination  $i$  will be sent to arrival node  $j$ . If an arriving message finds a token waiting at the arrival node, then the message is routed to the appropriate destination. Otherwise, the message is queued at the arrival node pending the appearance of a free token there. A common performance criterion is the average time for the transmissions. One wants to choose  $p_{ij}$  to minimize this quantity. The optimizing probabilities satisfy the Kuhn–Tucker condition for a certain constrained optimization problem. An algorithm of the stochastic approximation type is constructed in [245] such that the stationary points of the mean ODE are just the Kuhn–Tucker points. Reference [4] obtains IPA estimators of first and second derivatives with respect to service time parameters for a closed queueing network.

## 2.6 Passive Stochastic Approximation

Let  $\theta$  and  $Y_n$  be  $\mathbb{R}^r$ -valued. In a basic form of the Robbins–Monro algorithm, one observes  $Y_n = g(\theta, \xi_n)$  at values of the parameter  $\theta$  selected by the experimenter and seeks the value  $\bar{\theta}$  such that  $\bar{g}(\bar{\theta}) = Eg(\bar{\theta}, \xi_n) = 0$ . In some applications, the values of  $\theta_n$  are externally generated and cannot be chosen by the experimenter. Suppose that  $\{\theta_n\}$  is a random sequence, not selected by the experimenter, who still wishes to find the root (assumed unique for simplicity) of the function  $\bar{g}(\cdot)$  using measurements  $Y_n = \bar{g}(\theta_n) + \xi_n$  where  $\xi_n$  is a “noise” sequence.

In [95], Härdle and Nixdorf suggested an approach to the problem and termed it *passive stochastic approximation*. The approach can be traced

back to an early work of Révész [202]. Problems with real-valued  $\theta$  are treated in [95], and multivariate cases are handled in [181]. The main idea is to combine the stochastic approximation methods with nonparametric estimation procedures. Now let  $\theta$  and  $Y_n$  be  $\mathbb{R}^r$ -valued and  $g(\cdot)$  an  $\mathbb{R}^r$ -valued function on  $\mathbb{R}^r$ . Let  $K(\cdot)$  be a real-valued kernel function on  $\mathbb{R}^r$ ; it is non-negative, symmetric about the origin, and it takes its maximum value at  $\theta = 0$  and decreases monotonically to zero as any component of the argument increases to infinity. The root of the equation  $\bar{g}(\theta) = 0$  is approximated by the sequence  $\{\varpi_n\}$  defined by

$$\varpi_{n+1} = \varpi_n + \frac{\epsilon_n}{h_n^r} K\left(\frac{\theta_n - \varpi_n}{h_n}\right) Y_n, \quad (6.1)$$

where  $h_n$  represents the “window” width. The kernel function  $K(\cdot)$  plays a crucial role. If  $\varpi_n$  and  $\theta_n$  are far apart,  $K((\theta_n - \varpi_n)/h_n)$  will be very small, and the measurement  $Y_n$  has little effect on the iteration.

For robustness purposes, one may use the constant step size and constant window width (see [275]) algorithm

$$\varpi_{n+1}^\epsilon = \varpi_n^\epsilon + \frac{\epsilon}{\delta^r} K\left(\frac{\theta_n^\epsilon - \varpi_n^\epsilon}{\delta}\right) Y_n, \quad \text{for } n \geq 0. \quad (6.2)$$

The rate of convergence of  $\varpi_n^\epsilon$  to  $\bar{\theta}$  is slower than that for the classical Robbins–Monro method and depends on the smoothness of  $\bar{g}(\cdot)$  in  $\theta$ . As the number of continuous  $\theta$ -derivatives increases to infinity, the rate approaches that of the Robbins–Monro process [275]. A similar result holds when the constant step size is replaced by decreasing step sizes.

An alternative approach is to fit a nonlinear curve to the data and then use the fitted function to estimate the root. Nevertheless, this approach can be quite costly in its data requirements.

See [95, 202] for applications to nonlinear regression problems. An application to chemical process control is in [275], and [274] contains the results of numerical experiments on a binary distillation column model.

## 2.7 Learning in Repeated Stochastic Games

The theory of learning optimal strategies in repeated stochastic games leads to recursive stochastic algorithms whose mean ODEs can have quite interesting behavior. Many such algorithms have a special structure (e.g., they often satisfy the Kamke condition, which is defined above Theorem 4.4.1) and implies properties of the ODE that can be exploited to either prove convergence or to get insight into the asymptotic behavior. We will discuss only one relatively simple class of such problems. Many recursive algorithms that are based on either explicit or implicit competitive behavior have properties similar to those of repeated stochastic games; see, for example, the

proportional-fair sharing recursive algorithm in Chapter 3, which is used to allocate resources in time-varying wireless systems. The reference [38] concerns learning in another type of (implicit) game that arises in decentralized access control in communications. It has the characteristics of a game owing to the way that the various users affect each other's performance. Such problems have arisen in the economics literature [79]. There is a large literature concerning other uses of recursive algorithms to study "learning" phenomena in economics; for example, see [69, 175].

**A game model.** Consider a cooperative game with two players, each having two possible actions. Each player knows only its own payoff matrix at each time but keeps a record of the actions used by the other player. The hope is that each player will learn its optimal strategy, if there is one. Let  $\theta_{n,i}$  denote the fraction of the first  $n$  plays in which player  $i$  used its first action, called  $a_{i1}$ . Then  $\theta_n$  can be written recursively as

$$\theta_{n+1,i} = \theta_{n,i} + \frac{1}{n+1} \left[ I_{\{a_{i1} \text{ used at time } n+1\}} - \theta_{n,i} \right]. \quad (7.1)$$

At each time, player  $i$  first observes the payoff that it would receive under each of its actions, supposes that the other player chooses its action at random according to the historical record to date, and then chooses the action that maximizes its conditional average return. This scenario has been called *fictitious play* [11, 79]. Although the game is called cooperative, the players do not coordinate their actions. The following format and assumptions are taken from [11].

The actual payoffs to each player depend on the strategies of both players. Let  $\{v_{kl}; k, l = 1, 2\}$  be given real numbers. Suppose that player 1 uses  $a_{1k}$  and player 2 uses action  $a_{2l}$  at time  $n$ . Then the deterministic part of the payoff to each player is  $v_{kl}$ . A small random perturbation is added to this deterministic payoff. For small  $\delta > 0$ , the actual payoff to player 1 is  $u_{n,1,kl}^\delta = v_{kl} + \delta\eta_{n,1k}$  and that to player 2 is  $u_{n,2,kl}^\delta = v_{kl} + \delta\eta_{n,2l}$ , where  $(\eta_{n,ik}; i = 1, 2, k = 1, 2), n = 1, 2, \dots$ , are sequences of random variables which are identically distributed, have zero mean, and are independent in  $n$ . Thus, the part  $v_{kl}$  of the payoff is common to both players, but the random part is not: It plays the role of the effect of "private information." One of the interesting issues concerns the effect of such small "noisy" perturbations on the asymptotic behavior. Each player knows its own possible payoffs, but not those of the other player.<sup>1</sup>

Define the mean payoff to each player under each of its possible actions, given that the other player acts at random according to its historical record:

$$\bar{u}_{n+1,1,k}^\delta(\theta_{n,2}) = u_{n+1,1,k1}^\delta \theta_{n,2} + u_{n+1,1,k2}^\delta (1 - \theta_{n,2}).$$

---

<sup>1</sup>In [11], The difference  $\tilde{\eta}_{i,n} = \eta_{n,i1} - \eta_{n,i2}$  is assumed to have a density that is continuous and positive on the entire real line, with the density at value  $\eta$  going to zero fast enough so that it is as  $o(1/|\tilde{\eta}|)$ .



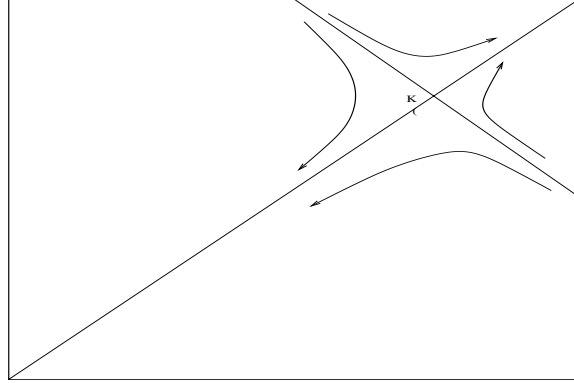


Figure 7.1. Phase plot for the perturbed game.

$$\bar{u}_{n+1,2,l}^\delta(\theta_{n,1}) = u_{n+1,2,1l}^\delta \theta_{n,1} + u_{n+1,2,2l}^\delta (1 - \theta_{n,1}).$$

Define the conditional probability that  $a_{11}$  is better for player 1 than is  $a_{12}$ :

$$\bar{h}_1^\delta(\theta_{n,2}) = P \{ \bar{u}_{n+1,1,1}^\delta(\theta_{n,2}) \geq \bar{u}_{n+1,1,2}^\delta(\theta_{n,2}) \}$$

and define  $\bar{h}_2^\delta(\theta_{n,1})$  for player 2 analogously.

The mean ODE for (1.1) is

$$\dot{\theta}^1 = \bar{g}_1(\theta) = \bar{h}_1^\delta(\theta^2) - \theta^1, \quad \dot{\theta}^2 = \bar{g}_2(\theta) = \bar{h}_2^\delta(\theta^1) - \theta^2. \quad (7.2)$$

Following [11], we say that  $\theta$  is a *Nash distribution equilibrium* if  $\bar{h}_1^\delta(\theta^2) = \theta^1$ ,  $\bar{h}_2^\delta(\theta^1) = \theta^2$ .

**Discussion.** The paper [11] contains additional references, a detailed discussion of convergence and nonconvergence, as well as of many other issues that arise, interpretations of the results in terms of the “information patterns,” and of the differences between the perturbed and unperturbed repeated games. As  $\delta \rightarrow 0$ , the Nash distribution equilibria for (7.2) converge to those for the system with  $\delta = 0$ . If there are more than two possible actions for each player or if there are more than two players, the behavior can be quite complicated, even chaotic. For example, there are three player, 2-action games whose sample average frequencies  $\theta_n$  converge to a non-degenerate limit cycle.

**Example.** This example is taken from [11]. Let  $v_{11} > v_{21}$ ,  $v_{22} > v_{12}$ ,  $v_{11} \geq v_{12}$ . Then, as  $\delta \rightarrow 0$  there are three Nash distribution equilibria;  $\theta = (1, 1)$ ,  $\theta = (0, 0)$  and  $\theta = (p^*, p^*)$ , where  $p^* = K/(1 + K)$  for  $K = (v_{22} - v_{12})/(v_{11} - v_{21})$ . The points  $\theta = (0, 0)$  and  $\theta = (1, 1)$  are stable. The mixed equilibrium at  $(p^*, p^*)$  is unstable. Then  $\theta_n$  converges (with probability one) to one of the pure strategies. [The noise in (7.1) destabilizes

the mixed strategy, as can be shown by using Theorem 5.8.1.] Let  $K > 1$ . Then, as  $\delta \rightarrow 0$ , the domain of attraction of  $(0, 0)$  is much larger than that of  $(1, 1)$ . This suggests that convergence is “more likely” to the point  $(0, 0)$ , the “risk dominating strategy.” The phase plot for the ODE is given in Figure 7.1.



<http://www.springer.com/978-0-387-00894-3>

Stochastic Approximation and Recursive Algorithms and  
Applications

Kushner, H.; Yin, G.

2003, XXII, 478 p., Hardcover

ISBN: 978-0-387-00894-3