11.5 Estimating a value function

Value functions have appeared in a surprising range of contexts in this book.

(i) The usual home for value functions is within the field of optimization. In the setting of this book, this means MDPs. Chapter 9 provides many examples, following the introduction for the single server queue presented in Chapter 3.

(ii) The stability theory for Markov chains and networks in this book is centered around Condition (V3). This is closely related to Poisson's inequality, which is itself a generalization of the average-cost value function.

(iii) Theorem 8.4.1 contains general conditions ensuring that the h-MaxWeight policy is stabilizing. The essence of the proof is that the function h is an approximation to Poisson's equation under the assumptions of the theorem.

(iv) We have just seen how approximate solutions to Poisson's equation can be used to dramatically accelerate simulation.

With the exception of (i), each of these techniques is easily applied in a wide range of settings. The basic reason for this success is that in each of these three cases we are *approximating* a value function. In the case of (iii) the function h in the h-MaxWeight policy is only a crude approximation to the average-cost dynamic programming equation; The simplicity of this policy is a consequence of this modest goal. In contrast, the 'curse of dimensionality' arises in optimization when we seek an exact solution.

In this final section of the book we consider methods to construct approximations via simulation. Our goal is to 'learn' the value function based on experiments on the network. Of course, learning brings its own curses. This is summarized in the following remark from [350]:

> A large state space presents two major issues. The most obvious one is the storage problem, as it becomes impractical to store the value function (or optimal action) explicitly for each state. The other is the generalization problem, assuming that limited experience does not provide sufficient data for each and every state.

The first issue is resolved by restricting to a parameterized family of approximate value functions. The learning problem is then reduced to finding the best approximation within this class.

If we are lucky, or have some insight on the structure of the value function, then a parameterization can also resolve the second issue. For example, if it is known that the value function is convex, then the family of approximations can be constructed to share this property. This imposes some continuity so that if a great deal is learned about the value function evaluated at a particular state x_0, then this information is useful for learning the value function at nearby points.

In the case of networks, there are natural parameterizations to consider based on results from previous chapters.

(i) The fluid value function J^* is the natural starting point to approximate the solution to the average-cost value function. In the case of the single server queue, Theorem 3.0.1 can be applied to conclude that the following parameterized family includes the actual value function,

$$h^\theta(x) = \theta_1 J^*(x) + \theta_2 x, \qquad x \in \mathbb{R}_+,\ \theta \in \mathbb{R}^2,$$

where $\theta_1 = 1$ when $h^\theta = h^*$ solves Poisson's equation. The discussion in Section 3.4.5 suggests similar approaches to approximate the discounted-cost value function. Example 11.4.4 illustrates how this approximation technique extends to networks.

(ii) The family of all quadratic functions can be regarded as a parameterized family. Linear programming methods were proposed in Section 8.6 to construct a solution to (V3).

(iii) The perturbed value function introduced in Section 4.9 is another example of a parameterized family of functions that can potentially approximate a value function. For example, given the family of functions $\{h(x) = h_0(\tilde{x})\}$ where h_0 ranges over some class, and the perturbation $\tilde{x}$ defined in (4.93) depends on the parameter $\theta > 0$, what is the best value of θ and h_0?

Each of the parameterizations in (i)–(iii) can be used to obtain an approximate value function for control or simulation. How then can we find the best approximation?

The evaluation criterion will depend on the context. In the case of simulation we might choose the approximation so that the resulting asymptotic variance is minimal. For control, the ultimate goal is to optimize performance over the class of policies considered. The algorithms described here can be used to approximate the value function for application in approximate value iteration or policy iteration. In this case, the metric to evaluate the approximation should reflect our goal to optimize performance.

In the remainder of this section we return to the general Markov setting in which $\boldsymbol{X}$ denotes a Markov chain without control on a state space X with transition matrix P, and unique invariant measure π. It isn't necessary to assume that X is countable, but we do assume there is a fixed state $x^* \in \mathsf{X}$ satisfying $\pi(x^*) > 0$. A function $c\colon \mathsf{X} \to \mathbb{R}$ is given, and our goal is to estimate a value function such as the solution to Poisson's equation, or the discounted-cost value function.

The basic approach to compute the best approximation is stochastic approximation or one of its variants.

11.5.1 Stochastic approximation

There are many different kinds of value functions that we might attempt to approximate. Regardless of its particular form, we assume that a parameterized family of approximations $\{h^\theta : \theta \in \mathbb{R}^{\ell_h}\}$ is given. In the case of a linear parametrization we suppose that we are given ℓ_h functions on X, denoted $\{\psi_i : 1 \le i \le \ell_h\}$, and define

$$h^\theta(x) := \theta^{\mathrm{T}} \psi(x) = \sum_{i=1}^{\ell_h} \theta_i \psi_i(x), \qquad x \in \mathsf{X}. \tag{11.60}$$

We then seek the best approximation in the given class based on a particular metric to describe the distance between h^θ and the value function h of interest.

Throughout most of this section we consider the L_2 *error*,

$$\mathcal{E}(\theta) = \|h - h^\theta\|_\pi^2 := \mathsf{E}_\pi[|h(X(0)) - h^\theta(X(0))|^2], \tag{11.61}$$

or a related *weighted* L_2-norm. On writing (11.61) as the sum,

$$\|h - h^\theta\|_\pi^2 = \sum |h(x) - h^\theta(x)|^2 \, \pi(x)$$

we see that this notion of distance penalizes the difference $|h(x) - h^\theta(x)|$ more strongly for states with larger steady-state probability $\pi(x)$. These states are visited more frequently by the chain, and are hence 'more important'.

This is valid motivation, but the most important reason for considering (11.61) is the fact that we can so easily construct an algorithm to minimize the error over θ.

It is assumed that h^θ is a smooth function of θ, so that the following gradient exists for each x,

$$\psi^\theta(x) := \nabla_\theta h^\theta \, (x). \tag{11.62}$$

The gradient is independent of θ when the parameterization is linear so that h^θ is expressed as the sum (11.60). We can formulate necessary conditions for optimality by differentiating the error with respect to θ, and setting this equal to zero. In the case of the L_2 error (11.61) the derivative with respect to θ has the probabilistic representation,

$$\nabla_\theta \|h^\theta - h\|_\pi^2 = 2\mathsf{E}_\pi[(h^\theta(X) - h(X))\psi^\theta(X)] \,, \tag{11.63}$$

provided one can justify exchanging the derivative and expectation. When the parameterization is linear and both h and ψ have finite second moments then (11.63) is justified, and on setting the derivative equal to zero we obtain the optimal parameter,

$$\theta^* = M_\psi^{-1} b_\psi, \qquad \text{where } M_\psi = \mathsf{E}[\psi(X)\psi(X)^{\mathrm{T}}], \quad b_\psi = \mathsf{E}[h(X)\psi(X)], \tag{11.64}$$

provided the matrix M_ψ is invertible.

The usual steepest descent algorithm to compute a minimum of $\|h^\theta - h\|_\pi^2$ is given by the ODE,

$$\begin{aligned} \tfrac{d}{d\theta}\theta(t) &= -\frac{a}{2}\nabla_\theta \|h^\theta - h_\gamma\|_\pi^2 \\ &= -a\mathsf{E}_\pi[(h^{\theta(t)}(X(k)) - h(X(k)))\psi^{\theta(t)}(X(k))]. \qquad t \ge 0, \end{aligned} \tag{11.65}$$

where $a > 0$ is a fixed gain. One stochastic approximation algorithm is constructed as the approximation of (11.65),

$$\theta(k+1) - \theta(k) = -a_k[(h^{\theta(k)}(X(k)) - h(X(k)))\psi^{\theta(k)}(X(k))], \quad k \ge 0, \tag{11.66}$$

where $\{a_k\}$ is a positive gain sequence satisfying (11.4).

The recursion (11.66) may very well approximate the ODE (11.65), and both may converge to the optimal value θ^*. Unfortunately there is a fundamental problem with either approach: the value function h appearing on the right hand side is not available *since this is the function we wish to approximate*! To obtain a practical algorithm we must find an alternative representation for the derivative (11.63). This step is the essential contribution of temporal difference learning.

11.5.2 Least Squares Temporal Difference learning for discounted cost

So far we have suppressed the precise definition of the value function. To construct a practical algorithm we are forced to be more explicit. The discounted-cost value function h_γ defined in (9.34) is the simplest starting point.

To simplify the discussion further we begin with a linear parameterization. This is the most common situation in practice, and in this case we can construct an efficient algorithm without much trouble. The optimal parameter θ^* has the representation (11.64) in this case, and our goal is to obtain a sequence of matrices $\{M(n)\}$ and vectors $\{b(n)\}$ such that $n^{-1}M(n) \to M_\psi$ and $n^{-1}b(n) \to b_\psi$ as $n \to \infty$, so that $\theta(n) = M(n)^{-1}b(n)$ is an asymptotically consistent estimator of θ^*.

To estimate the matrix M_ψ we can directly apply Monte-Carlo. Define the matrix sequence,

$$M(n) = M(0) + \sum_{t=0}^{n-1} \psi(X(t))\psi^{\mathrm{T}}(X(t)), \qquad n \geq 1, \tag{11.67}$$

where the matrix $M(0) > 0$ is introduced to ensure that $M(n)$ is invertible for each n. Provided the LLN holds for $\boldsymbol{X}$ we will have $n^{-1}M(n) \to M_\psi$ with probability one.

We assume that the LLN does hold. In fact, since all of the expectations here are in steady-state, we can and will assume that $\boldsymbol{X}$ is a stationary process, defined on the two-sided time-interval. We also let denote by X a generic random variable distributed according to π.

What about b_ψ? The defintion $b_\psi = \mathsf{E}[h(X(0))\psi(X(0))]$ involves the value function $h = h_\gamma$ we wish to estimate, so we must find an alternate representation. For this we rely on the definition of the value function,

$$h_\gamma(x) = \sum_{t=0}^{\infty} (1+\gamma)^{-t-1} \mathsf{E}[c(X(t)) \mid X(0) = x]$$

On multiplying each side by $\psi(x)$ we obtain,

$$h_\gamma(x)\psi(x) = \sum_{t=0}^{\infty} (1+\gamma)^{-t-1} \mathsf{E}[c(X(t))\psi(X(0)) \mid X(0) = x]$$

The vector we are attempting to estimate is precisely,

$$b_\psi = \mathsf{E}[h_\gamma(X)\psi(X)] = \sum_{x \in \mathsf{X}} h_\gamma(x)\psi(x)\,\pi(x)$$

Multiplying both sides of the previous equation by $\pi(x)$ and summing over x shows that b_ψ is also expressed as a sum,

$$\begin{aligned} b_\psi &= \sum_{t=0}^{\infty} (1+\gamma)^{-t-1} \sum_{x \in \mathsf{X}} \mathsf{E}[c(X(t))\psi(X(0)) \mid X(0) = x]\,\pi(x) \\ &= \sum_{t=0}^{\infty} (1+\gamma)^{-t-1} \mathsf{E}[c(X(t))\psi(X(0))] \end{aligned} \tag{11.68}$$

Each of the expectations in the sum at right involves functions that are known to us - the cost function c and the basis vector ψ.

The only remaining difficulty is that these expectations involve the process in the future, which complicates the application of Monte-Carlo. The *key step* to obtain a practical algorithm is to apply stationarity of $\boldsymbol{X}$, which implies that for any integer i,

$$\mathsf{E}[c(X(t))\psi(X(0))] = \mathsf{E}[c(X(t+i))\psi(X(i))] \tag{11.69}$$

In particular, we can set $i = -t$, so that (11.68) becomes,

$$b_\psi = \sum_{t=0}^{\infty}(1+\gamma)^{-t-1}\mathsf{E}[c(X(0))\psi(X(-t))] \tag{11.70}$$

One final transformation is needed. Assume that we have absolute integrability

$$\sum_{t=0}^{\infty}(1+\gamma)^{-t-1}\mathsf{E}[c(X(0))\|\psi(X(-t))\|] < \infty,$$

so that Fubini's Theorem can be applied to give,

$$b_\psi = \mathsf{E}\Big[c(X(0))\Big(\sum_{t=0}^{\infty}(1+\gamma)^{-t-1}\psi(X(-t))\Big)\Big]$$

Then, if we define for any k the random variable,

$$\varphi(k) = \sum_{t=0}^{\infty}(1+\gamma)^{-t-1}\psi(X(k-t))$$

the final expression can be expressed $b_\psi = \mathsf{E}[c(X(0))\varphi(0)]$. In this form we can apply Monte-Carlo: We have $n^{-1}b(n) \to b_\psi$ as $n \to \infty$, with

$$b(n) = b(0) + \sum_{t=0}^{n-1} c(X(t))\varphi(t), \tag{11.71}$$

and $b(0) \in \mathbb{R}^{\ell_h}$ an arbitrary initialization. The resulting algorithm is generalized slightly in the following formal definition.

The Matrix Inversion Lemma [221] is used to obtain a recursive algorithm that avoids repeated inversion of $M(n)$. This identity is proven on multiplying each side of (11.72) by $G^{-1} + HK^{\mathrm{T}}$

Lemma 11.5.1. (Matrix Inversion Lemma) *Suppose that G, H, and K are respectively $m\times m$, $m\times n$, and $n\times m$ matrices. If G and the sum $(I+K^{\mathrm{T}}GH)$ are invertible, then*

$$(G^{-1} + HK^{\mathrm{T}})^{-1} = G - GH(I + K^{\mathrm{T}}GH)^{-1}K^{\mathrm{T}}G. \tag{11.72}$$

□

In applying the Matrix Inversion we take $G(n) = M^{-1}(n)$ and $H = K = \psi(X(n))$ to obtain the formula for $G(n+1) = M^{-1}(n+1)$ given in (11.73b).

Definition 11.5.1. ***LSTD Learning for Discounted Cost***

For given initial conditions $G(0) > 0$, $b(0) \in \mathbb{R}^{\ell_h}$, $\varphi(0) \in \mathbb{R}^{\ell_h}$, the Least-Squares TD (LSTD) algorithm is defined by the sequence of parameter estimates,

$$\theta(n) = G(n)b(n), \tag{11.73a}$$

together with the three recursive equations,

$$G(n+1) = G(n) - \frac{G(n)\psi(X(n))\psi(X(n))^{\mathrm{T}}G(n)}{1+\psi(X(n))^{\mathrm{T}}G(n)\psi(X(n))} \tag{11.73b}$$

$$\varphi(n+1) = (1+\gamma)^{-1}[\varphi(t) + \psi(X(n+1))] \tag{11.73c}$$

$$b(n+1) = b(n) + \varphi(n)c(X(n)) \tag{11.73d}$$

The vector $\varphi(k)$ is called an *eligibility vector*. ■

To establish convergence of LSTD it is necessary to extend the Strong Law of Large beyond static functions of $X(t)$. This can be accomplished by extending the definition of $\boldsymbol{X}$. Define the bivariate process,

$$X'(t) = (X(t), \varphi(t)), \qquad t \geq 0. \tag{11.74}$$

This is a general state space Markov chain, but one that has attractive stability properties provided ψ is π-integrable.

Rather than develop properties of the more complex stochastic process (11.74), in the proof of Theorem 11.5.2 we extend the LLN from $\boldsymbol{X}$ to $\boldsymbol{X}'$ through brute-force calculation in a very special case.

Theorem 11.5.2. (Convergence of LSTD for Discounted Cost) *Suppose that $\boldsymbol{X}$ is an ergodic, finite state-space Markov chain, and that $M_\psi > 0$. Then with probability one, from each initial condition, $\lim_{n\to\infty} G(n) = M_\psi^{-1}$ and $\lim_{n\to\infty} n^{-1}b(n) = b_\psi$. Hence the LSTD algorithm is convergent:*

$$\theta(n) = G(n)b(n) \to \theta^* \quad \text{as } n \to \infty.$$

Proof. From the Matrix Inversion Lemma we have,

$$nG(n) = n\Big(M(0) + \sum_{t=0}^{n-1} \psi(X(t))\psi^{\mathrm{T}}(X(t))\Big)^{-1},$$

so that convergence of $nG(n)$ follows by the LLN for $\boldsymbol{X}$.

We now consider the sequence $\{b(n)\}$. Based on (11.73c) we obtain,

$$\varphi(t) = (1+\gamma)^{-t-1}\varphi(0) + \sum_{k=0}^{t}(1+\gamma)^{-k-1}\psi(X(t-k))\,,$$

so that ignoring transient terms involving the initial conditions,

$$\lim_{n\to\infty}\frac{1}{n}b(n) = \lim_{n\to\infty}\frac{1}{n}\sum_{t=0}^{n-1}c(X(t))\Big(\sum_{k=0}^{t}(1+\gamma)^{-k-1}\psi(X(t-k))\Big).$$

Following a change in the order of summation, the right hand side becomes

$$\begin{aligned}\frac{1}{n}\sum_{k=0}^{n-1}(1+\gamma)^{-k-1}\Big(\sum_{t=k}^{n-1}c(X(t))\psi(X(t-k))\Big)\\ = \sum_{k=0}^{n-1}(1+\gamma)^{-k-1}\frac{n-k}{n}\Big(\frac{1}{n-k}\sum_{t=0}^{n-k-1}c(X(t+k))\psi(X(t))\Big).\end{aligned}$$

For any fixed k the LLN gives,

$$\lim_{n\to\infty}\frac{1}{n-k}\sum_{t=0}^{n-k-1}c(X(t+k))\psi(X(t)) = \mathsf{E}[c(X(k))\psi(X(0))]$$

With a bit more book-keeping we obtain from this the desired conclusion,

$$\lim_{n\to\infty}\frac{1}{n}b(n) = \sum_{k=0}^{\infty}(1+\gamma)^{-k-1}\mathsf{E}[c(X(k))\psi(X(0))] = b_\psi.$$

□

11.5.3 Adjoint equations and TD learning

We now consider nonlinear parameterizations and derive the standard TD algorithm for value function approximation. This is based on transformations similar to those used in the case of a linear parameterization. In particular, a version of the invariance equation (11.69) is again critical.

A more compact, and perhaps more elegant construction of the algorithm is obtained by casting the approximation problem in a vector space setting. The vector space is the *Hilbert space* denoted $L_2(\pi)$, defined as the set of real-valued functions on X whose second moment under π is finite. We define an inner product on this Hilbert space that is consistent with the L_2 error (11.61),

$$\langle f, g\rangle_\pi = \pi(fg) = \sum f(x)g(x)\,\pi(x),$$

so that for any two functions $f, g \in L_2(\pi)$, the L_2 norm of their difference is expressed,

$$\|f-g\|_\pi^2 := \langle f-g, f-g\rangle_\pi = \mathsf{E}[(f(X)-g(X))^2].$$

If the state space is finite, with N states, then the functions f and g can be viewed as N-dimensional column vectors, and π an N-dimensional row vector. The inner-product is an ordinary inner-product in $\mathbb{R}^N$,

$$\langle f, g\rangle_\pi = f^{\mathrm{T}}\Pi g,$$

where $\Pi = \text{diag}\,(\pi)$.

Next we express the discounted-cost value function as a matrix-vector product. For each t, the t-step transition matrix P^t is the t-fold product of P with itself. For a given discount rate $\gamma > 0$, the resolvent matrix is defined as the infinite sum,

$$R_\gamma = \sum_{t=0}^{\infty}(1+\gamma)^{-t-1}P^t.$$

This is actually a power-series expansion for the matrix inverse,

$$R_\gamma = [\gamma I - \mathcal{D}]^{-1} = [(1+\gamma)I - P]^{-1}$$

where the generator is defined in (8.1). With the cost function c interpreted as a column vector we can express the value function as the product $h_\gamma = R_\gamma c$.

In this new notation, the derivative (11.63) can be expressed as the inner product,

$$\nabla_\theta \|h^\theta - h_\gamma\|_\pi^2 = 2\langle (h^\theta - R_\gamma c), \psi^\theta\rangle_\pi. \tag{11.75}$$

We now interpret the transformations performed to construct the LSTD algorithm as the application of an adjoint operation in $L_2(\pi)$.

The adjoint of a real $N \times N$ matrix is simply its transpose. In the vector space setting of this section, the adjoint of the resolvent $R_\gamma^\dagger$ is characterized by the set of equations,

$$\langle R_\gamma f, g\rangle_\pi = \langle f, R_\gamma^\dagger g\rangle_\pi, \qquad f, g \in L_2(\pi). \tag{11.76}$$

A sample path representation for $\langle R_\gamma f, g\rangle_\pi$ leads to a useful representation for the adjoint. We begin with the definition,

$$\langle R_\gamma f, g\rangle_\pi = \mathsf{E}\big[(R_\gamma f\,(X))(g(X))\big] = \mathsf{E}\Big[\Big(\sum_{t=0}^{\infty}(1+\gamma)^{-t-1}P^t f\,(X(0))\Big)g(X(0))\Big]$$

We have by the smoothing property of the conditional expectation,

$$\mathsf{E}[P^t f\,(X(0))g(X(0))] = \mathsf{E}\big[\mathsf{E}[f(X(t)) \mid X(0)]g(X(0))\big] = \mathsf{E}[f(X(t))g(X(0))]$$

and then applying (11.69) we obtain,

$$\langle R_\gamma f, g\rangle_\pi = \sum_{t=0}^{\infty}(1+\gamma)^{-t-1}\mathsf{E}[f(X(0))g(X(-t))].$$

The right hand side can be expressed $\langle f, R_\gamma^\dagger g\rangle_\pi$, where

$$R_\gamma^\dagger g\,(x) = \sum_{t=0}^{\infty} (1+\gamma)^{-t-1} \mathsf{E}[g(X(-t)) \mid X(0) = x], \qquad x \in \mathsf{X}. \tag{11.77}$$

That is, the adjoint $R_\gamma^\dagger$ is the ordinary resolvent for the time-reversed process $\{X(-t) : t \in \mathbb{Z}_+\}$.

We now obtain an alternate expression for the derivative (11.63) based on the equivalent form (11.75). The resolvent is invertible, with $R_\gamma^{-1} = (1+\gamma)I - P$. Hence the difference $h^\theta - h_\gamma$ can be expressed,

$$h^\theta - h_\gamma = h^\theta - R_\gamma c = R_\gamma[R_\gamma^{-1} h^\theta - c] = R_\gamma[(1+\gamma)h^\theta - Ph^\theta - c]. \tag{11.78}$$

Based on this expression, the representation (11.75), and the adjoint equation (11.76), we obtain

$$\tfrac{1}{2}\nabla_\theta \|h^\theta - h_\gamma\|_\pi^2 = \left\langle (1+\gamma)h^\theta - Ph^\theta - c, R_\gamma^\dagger \psi^\theta \right\rangle_\pi, \tag{11.79}$$

or written as an expectation,

$$\tfrac{1}{2}\nabla_\theta \|h^\theta - h_\gamma\|_\pi^2 = \mathsf{E}[d^\theta(t)\varphi^\theta(t)], \tag{11.80}$$

where $d^\theta(t) := (1+\gamma)h^\theta(X(t)) - h^\theta(X(t+1)) - c(X(t))$, and $\varphi^\theta(t) := R_\gamma^\dagger \psi\,(X(t))$.

We now have sufficient motivation to construct the TD learning algorithm based on the ODE (11.65).

Definition 11.5.2. ***TD Learning for Discounted Cost***

The TD-algorithm constructs recursively a sequence of estimates $\{\theta(n)\}$ based on the following,

(i) *Temporal differences* defined by,

$$d(k) := -[(1+\gamma)h^{\theta(k)}(X(k)) - h^{\theta(k)}(X(k+1)) - c(X(k))]\,, \qquad k \ge 1. \tag{11.81}$$

(ii) *Eligibility vectors*: the sequence of ℓ_h-dimensional vectors,

$$\varphi(k) = \sum_{t=0}^{k} (1+\gamma)^{-t-1} \psi^{\theta(k-t)}(X(k-t))\,, \qquad k \ge 1, \tag{11.82}$$

expressed recursively via,

$$\varphi(k+1) = (1+\gamma)^{-1}[\varphi(k) + \psi^{\theta(k+1)}(X(k+1))], \qquad k \ge 0,\ \varphi(0) = 0.$$

The estimates of θ^* are defined by,

$$\theta(n+1) - \theta(n) = a_n d(n)\varphi(n+1), \qquad n \ge 0, \tag{11.83}$$

where the non-negative gain sequence $\{a_n\}$ satisfies (11.4). ∎

Based on (11.80), for large k the following approximation is suggested,

$$\mathsf{E}[d(k)\varphi(k+1)] \approx -\tfrac{1}{2}\nabla_\theta \|h^\theta - h_\gamma\|_\pi^2, \qquad \theta(i) \equiv \theta.$$

The TD algorithm is the stochastic approximation algorithm associated with the ODE (11.65), based on this approximation.

11.5.4 Average cost

Much of Part III of the book has concentrated on average-cost control and average-cost performance evaluation, and most of the approximation results have focused on Poisson's equation rather than the discounted-cost analog. We now generalize the results of Sections 11.5.2 and 11.5.3 to this setting.

Recall that a solution to Poisson's equation, also called the relative value function, can be expressed as a SPP value function with respect to the relative cost $\tilde{c} := c - \eta$. For a fixed state $x^* \in \mathsf{X}$, we take the particular form,

$$h_{\tilde{c}}(x) = \mathsf{E}_x\Big[\sum_{t=0}^{\sigma_{x^*}} \tilde{c}(X(t))\Big], \qquad x \in \mathsf{X}, \tag{11.84}$$

where σ_{x^*} denotes the first hitting time to x^*.

To extend the Hilbert space framework to this setting we express the value function as an infinite-horizon sum,

$$h_{\tilde{c}}(x) = \sum_{t=0}^{\infty} \mathsf{E}_x\big[\mathbf{1}\{\sigma_{x^*} \ge t\}\tilde{c}(X(t))\big].$$

Next we express each term in the sum in 'matrix-vector product' notation. Letting $\mathbf{1}_{\{x^*\}^c}P$ denote the matrix whose 'row' corresponding to x^* has been set to zero, we have the probabilistic interpretation, for any function g,

$$\mathbf{1}_{\{x^*\}^c}Pg\,(x) = \mathsf{E}_x\big[\mathbf{1}\{\sigma_{x^*} \ge 1\}g(X(1))\big].$$

With $(\mathbf{1}_{\{x^*\}^c}P)^t$ the t-fold matrix product we similarly have,

$$(\mathbf{1}_{\{x^*\}^c}P)^t g\,(x) = \mathsf{E}_x\big[\mathbf{1}\{\sigma_{x^*} \ge t\}g(X(t))\big].$$

Hence the relative value function can be expressed as $h_{\tilde{c}} = R_0\tilde{c}$, where the *potential matrix* is defined for arbitrary functions g via

$$R_0 g\,(x) := \sum_{t=0}^{\infty} (\mathbf{1}_{\{x^*\}^c}P)^t g\,(x), \qquad x \in \mathsf{X}.$$

Just as in our consideration of the resolvent, the potential matrix can be expressed as a matrix inverse

$$R_0 = \sum_{t=0}^{\infty} (\mathbf{1}_{\{x^*\}^c}P)^t = [I - \mathbf{1}_{\{x^*\}^c}P]^{-1}. \tag{11.85}$$

A similar representation for the relative value function is given in (A.31).

We can follow the derivation of $R_\gamma^\dagger$ in (11.77) to express the adjoint as the potential matrix for the time-reversed process,

$$R_0^\dagger g\,(x) = \mathsf{E}\Big[\sum_{\tilde{\sigma}_{x^*}^{[0]} \le t \le 0} g(X(t)) \mid X(0) = x\Big], \qquad x \in \mathsf{X},\ g \in L_2(\pi), \tag{11.86}$$

where for any k,

$$\tilde{\sigma}_{x^*}^{[k]} = \max\{t \le k : X(t) = x^*\}.$$

Consider the error (11.61) and its gradient (11.63). In the Hilbert space notation introduced in Section 11.5.3, we obtain a representation similar to (11.75),

$$\nabla_\theta \mathcal{E}(\theta) = 2\langle (h^\theta - h_{\tilde{c}}), \psi^\theta \rangle_\pi.$$

Applying the representation (11.85) we obtain, exactly as in (11.78),

$$h^\theta - h_{\tilde{c}} = R_0[R_0^{-1}h^\theta - \tilde{c}] = R_0[h^\theta - \mathbf{1}_{\{x^*\}^c} P h^\theta - \tilde{c}].$$

Hence, the first-order condition for optimality of θ becomes,

$$0 = \tfrac{1}{2}\nabla_\theta \mathcal{E}(\theta) = \langle [h^\theta - \mathbf{1}_{\{x^*\}^c} P h^\theta - \tilde{c}], R_0^\dagger \psi^\theta \rangle_\pi.$$

Following the derivation of TD learning for the discounted-cost value function we arrive at a TD learning algorithm to estimate $h_{\tilde{c}}$.

Definition 11.5.3. ***TD Learning for Poisson's Equation***

The TD-algorithm constructs recursively a sequence of estimates $\{\theta(k)\}$ based on the following,

(i) *Temporal differences,*

$$d(k) := -\left[h^{\theta(k)}(X(k)) - \mathbf{1}_{\{x^*\}^c}(X(k)) h^{\theta(k)}(X(k+1)) - \big(c(X(k)) - \eta(k)\big)\right].$$

(ii) *Eligibility vectors,* the sequence of ℓ_h-dimensional vectors,

$$\varphi(k) = \sum_{t=\tilde{\sigma}_{x^*}^{[k]}}^{k} \psi^{\theta(k-t)}(X(k-t)), \qquad k \ge 1, \tag{11.87}$$

or written recursively,

$$\varphi(k+1) = \mathbf{1}\{X(k) \ne x^*\}\varphi(k) + \psi^{\theta(k+1)}(X(k+1)), \qquad k \ge 0$$

(iii) Estimates $\{\eta(k)\}$ of η are obtained using Monte-Carlo (11.1), or any other consistent method.

Estimates $\{\theta(n)\}$ of the optimal parameter are then obtained using the TD recursion (11.83). ■

When the parameterization is linear we can again use the Monte-Carlo estimates (11.73a), where the definition of $\{b(n)\}$ is redefined by,

$$b(n) = \sum_{t=0}^{n-1} \big(c(X(t)) - \eta(n)\big)\varphi(t), \qquad n \ge 1,$$

with $\{\varphi(t)\}$ generated using the recursion (11.87).

Unfortunately, neither the TD or LSTD algorithms are effective in queueing models due to the very large variance of the estimates. This will be seen in Example 11.5.1, but first we introduce a method to reduce the variance. For this we modify the norm through the introduction of a weighting function $\Omega\colon \mathsf{X} \to [1,\infty]$. Define for two functions f, g the new inner-product

$$\langle f, g\rangle_{\pi,\Omega} = \sum \big(f(x)g(x)/\Omega(x)\big)\, \pi(x),$$

with associated weighted norm,

$$\|f-g\|^2_{\pi,\Omega} := \langle f-g, f-g\rangle_{\pi,\Omega} = \mathsf{E}[(f(X)-g(X))^2/\Omega(X)]. \tag{11.88}$$

We use this norm to define the error between h^θ and $h_{\tilde c}$ for a given parameter θ:

$$\mathcal{E}(\theta) = \|h^\theta - h_{\tilde c}\|^2_{\pi,\Omega}. \tag{11.89}$$

As in the foregoing, we obtain a representation for the derivative of $\mathcal{E}(\theta)$; Setting this equal to zero gives the first-order necessary condition for optimality of θ. Under general conditions (to justify taking the derivative inside the inner product) the derivative can be expressed $\nabla_\theta \mathcal{E}(\theta) = 2\langle h^\theta - h_{\tilde c}, \psi^\theta\rangle_{\pi,\Omega}$,, where ψ^θ is the gradient of h^θ. For a linear parameterization this is independent of θ, giving

$$\tfrac{1}{2}\nabla_\theta \mathcal{E}(\theta) = \langle h^\theta, \psi\rangle_{\pi,\Omega} - \langle h_{\tilde c}, \psi\rangle_{\pi,\Omega}$$

On denoting

$$M_\psi = \mathsf{E}[\psi(X)\psi(X)^{\mathrm{T}}\Omega^{-1}(X)], \tag{11.90}$$

we have $\langle h^\theta, \psi\rangle_{\pi,\Omega} = M_\psi\theta$. From this expression and the definition of the adjoint we have,

$$\tfrac{1}{2}\nabla_\theta \mathcal{E}(\theta) = M_\psi\theta - \langle \tilde c, R_0^\dagger\psi\rangle_{\pi,\Omega}$$

and setting this equal to zero gives the unique optimizer, provided the matrix M_ψ is invertible.

To obtain an algorithm we must first interpret these inner products. Based on (11.90), the matrix M_ψ can be estimated using an obvious modification of (11.67).

The inner-product $\langle \tilde c, R_0^\dagger\psi\rangle_{\pi,\Omega}$ plays the role of the vector b_ψ that was introduced in the construction of LSTD for discounted-cost. The adjoint $R_0^\dagger$ is again given by (11.86), so that the LSTD algorithm for average cost is obtained as a minor modification of Definition 11.5.1.

Definition 11.5.4. ***LSTD Learning for Average Cost with State Weighting***

For given initial conditions $G(0) > 0$, $b(0) \in \mathbb{R}^{\ell_h}$, $\varphi(0) \in \mathbb{R}^{\ell_h}$, the sequence of parameter estimates are defined by

$$\theta(n) = G(n)b(n), \tag{11.91a}$$

where the sequences $\boldsymbol{G}$ and $\boldsymbol{b}$ are defined by the recursive equations,

$$G(n+1) = G(n) - \frac{G(n)\psi(X(n))\psi(X(n))^{\mathrm{T}}G(n)}{\Omega(X(n)) + \psi(X(n))^{\mathrm{T}}G(n)\psi(X(n))} \tag{11.91b}$$

$$b(n+1) = b(n) + \varphi(n)\big(c(X(n)) - \eta(n)\big) \tag{11.91c}$$

$$\varphi(n+1) = \mathbf{1}\{X(n) \neq x^*\}\varphi(n) + \psi(X(n+1))/\Omega(X(n+1)). \tag{11.91d}$$

■

Example 11.5.1. ***LSTD for the M/M/1 queue***

The solution to Poisson's equation for the M/M/1 queue is the quadratic given in Proposition 3.4.2. For arbitrary $\theta \in \mathbb{R}^2$ define,

$$h^\theta(x) = \theta_1 x + \theta_2 x^2, \qquad x \in \mathbb{R}_+. \tag{11.92}$$

Then, with $\theta_1^* = \theta_2^* = \frac{1}{2}(\mu - \alpha)^{-1}$ this is the solution given in Proposition 3.4.2.

To estimate θ^* we first apply the LSTD algorithm with $\Omega \equiv 1$. Observe that the recursion (11.91b) is designed to estimate the inverse of M_ψ. This involves estimating indirectly the mean of the *fourth moment* of the queue-length process since,

$$M_\psi = M_\psi = \mathsf{E}[\psi(Q)\psi(Q)^{\mathrm{T}}] = \mathsf{E}\Big[\begin{pmatrix} Q & Q^3 \\ Q^3 & Q^4 \end{pmatrix}\Big].$$

The asymptotic variance of the standard estimator of $Q(t)^4$ is of order $(1-\rho)^{-10}$ (!) Hence we can expect high variances when using the LSTD algorithm.

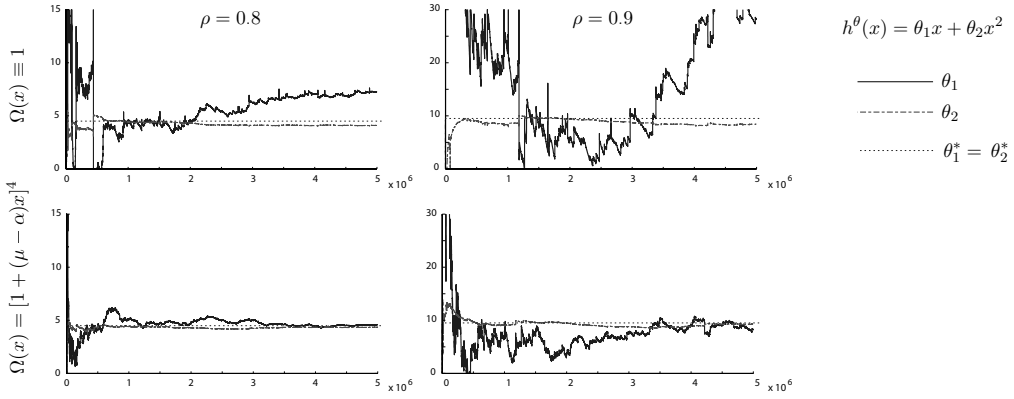

Figure 11.10: LSTD estimates for the relative value function in the M/M/1 queue based on Definition 11.5.4.

Shown in Figure 11.10 are results from several experiments using the LSTD algorithm. In one set of experiments the weighting function was set to unity, and in the other the polynomial,

$$\Omega(x) = (1 + (\mu - \alpha)x)^p, \qquad x \geq 0.$$

Several values of p were tried in experiments; the best value of $p = 4$ was chosen in the figure.

Consequences of the high variance are evident in Figure 11.10. For loads of $\rho = 0.8$ or higher the estimates show high variability even after 5 million iterations. The introduction of the weighting function *significantly* reduces variability. ∎

11.5.5 Optimizing shadow functions

The last performance criterion to be considered is the asymptotic variance. For simplicity we restrict to linear parametrizations with $h^\theta = \theta^{\mathrm{T}}\psi$.

The first step towards constructing a recursive algorithm is to obtain an expression for the asymptotic variance in terms of the adjoint. Proposition 11.5.3 provides a useful formula for the uncontrolled estimator.

Proposition 11.5.3. *The asymptotic variance in (11.26) can be expressed,*

$$\sigma^2_{\text{CLT}} = \pi(2\tilde{c}h^\dagger_{\tilde{c}} - \tilde{c}^2), \tag{11.93}$$

where $h^\dagger_{\tilde{c}} := R_0^\dagger \tilde{c}$, and the adjoint is defined in (11.86).

Proof. The representation (11.26) combined with Poisson's equation gives,

$$\sigma^2_{\text{CLT}} = \pi(h^2_{\tilde{c}} - (Ph_{\tilde{c}})^2) = \pi(h^2_{\tilde{c}} - (h_{\tilde{c}} - \tilde{c})^2) = \pi(2h_{\tilde{c}}\tilde{c} - \tilde{c}^2).$$

From the definition of the adjoint and $h_{\tilde{c}}$ we have $\pi(h_{\tilde{c}}\tilde{c}) = \langle R_0\tilde{c}, \tilde{c}\rangle_\pi = \langle \tilde{c}, R_0^\dagger\tilde{c}\rangle_\pi$, and (11.93) then follows. □

The value of Proposition 11.5.3 is that the steady-state mean $\pi(\tilde{c}h^\dagger_{\tilde{c}})$ is easily estimated using standard Monte-Carlo since $h^\dagger_{\tilde{c}}\tilde{c}$ can be expressed in terms of the history of the process. Indeed, define

$$\varphi_c(k+1) = \mathbf{1}\{X(k) \neq x^*\}\varphi_c(k) + \big(c(X(k+1)) - \eta(k+1)\big), \qquad k \geq 0,$$

where $\{\eta(k)\}$ are consistent estimates of η. Then, under general conditions,

$$\pi(\tilde{c}h_{\tilde{c}}) = \lim_{k\to\infty} \pi\big(\tilde{c}(X(k))\varphi_c(k)\big) = \lim_{n\to\infty} \frac{1}{n}\sum_{k=0}^{n-1}\big(c(X(k)) - \eta(k)\big)\varphi_c(k) \qquad a.s.$$

We now turn to shadow functions. Proposition 11.4.1 tells us that the optimal parameter θ^* is the solution to

$$\Sigma_\psi\theta = \langle \psi, h_{\tilde{c}}\rangle_{\text{CLT}}\,, \tag{11.94}$$

where Σ_ψ is defined in (11.43). Applying the adjoint technique once more we obtain the following expression for the θ^*:

Proposition 11.5.4. *Under the assumptions of Proposition 11.4.1 the optimal parameter (11.44) can be expressed,*

$$\theta^* = \Sigma_\psi^{-1} b_\psi,$$

where

$$b_\psi = \langle \tilde{c}, R_0^\dagger(\psi - \psi^1) + \psi^1\rangle_\pi, \tag{11.95}$$

$R_0^\dagger$ is defined in (11.86), and $\psi^1 := P\psi$.

Proof. In view of (11.94), it is enough to establish that the vector b_ψ defined in (11.95) coincides with the vector $\langle \psi, h_{\tilde{c}} \rangle_{\text{CLT}}$. Poisson's equation for $h_{\tilde{c}}$ gives,

$$\langle \psi, h_{\tilde{c}} \rangle_{\text{CLT}} = \langle \psi, h_{\tilde{c}} \rangle_\pi - \langle P\psi, Ph_{\tilde{c}} \rangle_\pi = \langle \psi, h_{\tilde{c}} \rangle_\pi - \langle \psi^1, h_{\tilde{c}} - \tilde{c} \rangle_\pi,$$

or $\langle \psi, h_{\tilde{c}} \rangle_{\text{CLT}} = \langle \psi - \psi^1, R_0 \tilde{c} \rangle_\pi + \langle \psi^1, \tilde{c} \rangle_\pi$. From the defining property of the adjoint,

$$\langle \psi - \psi^1, R_0 \tilde{c} \rangle_\pi = \langle R_0^\dagger (\psi - \psi^1), \tilde{c} \rangle_\pi$$

we obtain the desired conclusion that $b_\psi = \langle \psi, h_{\tilde{c}} \rangle_{\text{CLT}}$. ☐

To estimate the optimizer θ^* we can separately estimate Σ_ψ and $\eta := \pi(c)$ (required to construct $\tilde{c}$.) It is also necessary to estimate expectations involving the two functions $R_0^\dagger \psi$, and $R_0^\dagger \psi^1$.

The random variables $\{\varphi(t)\}$ defined in (11.87) will be used for estimating expectations involving $R_0^\dagger \psi$, and we introduce a second sequence of eligibility vectors for $R_0^\dagger \psi^1$:

$$\varphi^1(k) = \sum_{t=\tilde{\sigma}_{x^*}^{[k]}}^{k} \psi^1(X(k-t)), \qquad k \geq 1. \tag{11.96}$$

We can then estimate the vector b_ψ defined in (11.95) as $n^{-1} b(n)$ with

$$b(n) = \sum_{t=0}^{n-1} \Big[\big(\varphi(t) - \varphi^1(t) + \psi^1(X(t)) \big) \big(c(X(t)) - \eta(n) \big) \Big] \tag{11.97}$$

The matrix Σ_ψ can be estimated using standard Monte-Carlo $n^{-1}\Sigma(n)$, with

$$\Sigma(n) = \sum_{t=0}^{n-1} \Big[\psi(X(t)) \psi^{\text{T}}(X(t)) - \psi^1(X(t)) \psi^{1\,\text{T}}(X(t)) \Big]. \tag{11.98}$$

Hence we obtain estimates of θ^* using,

$$\theta(n) = \Sigma(n)^{-1} b(n), \qquad n \geq 1.$$

Once again we obtain a recursive algorithm based on the Matrix Inversion Lemma 11.5.1. Define the two $\ell_h \times 2$ matrices $\Psi(n) = [\psi(X(n)) \mid \psi^1(X(n))]$, $\Psi^-(n) = [\psi(X(n)) \mid -\psi^1(X(n))]$ so that,

$$\Psi(n)\Psi^-(n)^{\text{T}} = \psi(X(n))\psi(X(n))^{\text{T}} - \psi^1(X(n))\psi^1(X(n))^{\text{T}}.$$

Then, with $G(n) = \Sigma(n)^{-1}$, the inverse $G(n+1) = [G(n)^{-1} + \Psi(n)\Psi^-(n)^{\text{T}}]^{-1}$ is expressed as (11.99) using the Matrix Inversion Lemma.

Definition 11.5.5. ***LSTD Learning for Shadow Functions***

For given initial conditions $G(0) > 0$, $b(0) \in \mathbb{R}^{\ell_h}$, $\varphi(0), \varphi^1(0) \in \mathbb{R}^{\ell_h}$, the LSTD algorithm is defined by the sequence of parameter estimates,

$$
\begin{aligned}
\theta(n+1) &= G(n+1)b(n+1) \\
G(n+1) &= G(n) - G(n)\Psi(n)[I + \Psi^-(n)^{\mathrm{T}}G(n)\Psi(n)]^{-1}\Psi^-(n)^{\mathrm{T}}G(n) \\
b(n+1) &= b(n) + \big(\varphi(n) - \varphi^1(n) + \psi^1(X(n))\big)\big(c(X(n)) - \eta(n+1)\big) \\
\varphi(n+1) &= \mathbf{1}\{X(n) \neq x^*\}\varphi(n) + \psi(X(n+1)) \\
\varphi^1(n+1) &= \mathbf{1}\{X(n) \neq x^*\}\varphi^1(n) + \psi^1(X(n+1)).
\end{aligned}
$$

■

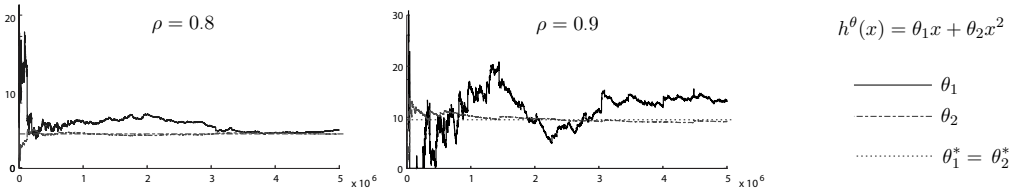

Figure 11.11: LSTD estimates for optimizing shadow functions in the M/M/1 queue using Definition 11.5.5.

Example 11.5.2. ***LSTD for the M/M/1 queue***

Figure 11.11 shows results from the LSTD algorthim based on the basis functions $\psi_1(x) \equiv x$, $\psi_2(x) \equiv x^2$. The two experiments shown in the figure are typical results for $\rho = 0.8$ and 0.9.

Note that the sequence $\{\theta(n)\}$ is convergent in this setting, but the variance is again high. This can be seen in the figure, and it can be shown analytically that it is even more variable than the estimates of the mean $\eta = \rho/(1-\rho)$. Nevertheless, convergence of $\{\theta_2(n)\}$ to a value reasonably close to θ_2^* occurs within $500,000$ iterations in each experiment.

Convergence of $\{\theta_1(n)\}$ is *much slower*. This is to be expected since the asymptotic variance of the smoothed estimator is less sensitive to this coefficient. ■

11.6 Notes

This chapter spans several disciplines, and the topics surveyed here cover only a small fraction of two fields, simulation and machine learning. *Modern simulation and modeling* by Rubinstein and Melamed [419] provides a broad and accessible introduction to simulation. Machine learning is a rapidly evolving discipline. Some of the most elegant recent work has emerged from Tsitsiklis and his progeny. The monographs Sutton and Barto [464] and Bertsekas and Tsitsiklis [53] contain introductions to this field, and some specific papers in the area of TD learning are discussed below.

The stochastic approximation algorithm (11.3) was introduced in Robbins and Monro's celebrated 1951 paper [408]. One year later, Kiefer and Wolfowitz introduced a variation intended to solve optimization problems in which the objective function is not smooth, or the derivative is not easily computed [299]. It is remarkable that this highly applicable methodology was developed almost fifty years before computers became fast enough to make these algorithms widely implementable. Nevel'son and Has'minskiĭ [384] is a valuable classic text; See Kushner and Yin [330], Benveniste, Métivier and Priouret [45], or Chen [102] for modern treatments.

A highly influential paper on simulation of Markov chains with an emphasis consistent with this chapter is Hordijk et. al. [270]. In particular, an emphasis of the paper is estimation of confidence bounds, as well as the asymptotic variance (11.6). Heidelberger's thesis on related topics [251, 250] contains a kernel of the control variate techniques based on the shadow functions described in Section 11.4.

The fantastically large runlengths required for accurate simulation in queueing models was first recognized by Whitt [492] and Asmussen [24]. For more on related themes see the work of Glynn, Iglehart and Whitt [276, 212, 214, 492, 218], and especially the survey [215].

Section 11.2.1 on the CLT for Markov models is based on [216, 367]. Much of Section 11.1 is adapted from the survey [252]. In particular, Proposition 11.1.1 is taken from [252], following [65, Exercise 29.4] (1986 edition).

One interpretation of Proposition 11.3.1 is that the moments and asymptotic variance of the CRW model converge to those of an associated RBM as $\rho \uparrow \infty$, which follows from Kingman's original 1961 result [303]. Related results for generalized Jackson networks are contained in [199, 89].

There are several excellent treatments of large deviations and overflow probabilities for queues. See [219, 155, 446], and in particular the book *Big Queues* [200].

The failure of the LDP for the single server queue as summarized in Proposition 11.3.4 (as well as Theorem 11.2.3) is taken from [364]. The proof presented here based on Figure 3.2 is inspired by the papers of Guillemin and Pinchon [223], and Borovkov, Boxma, and Palmowski [78] where the *most likely area* under a tent is computed in an asymptotic setting in which the area tends to infinity. This result can be used to establish a non-zero limit in (11.37), rather than a lower bound.

A valuable reference on control variates and other variance reduction techniques is the *Handbook of Simulation* by Henderson and Nelson [256]. See also Nelson [382], Rubinstein and Melamed [419], and the survey by Glynn and Szechtman [211]. Much of Section 11.4 is based on joint work with Henderson, and his thesis [257]. The

quadratic estimator was introduced in [254], which was inspired by Henderson's thesis and prior work with Glynn on control variates for the GI/G/1 queue [257, 253]. Section 11.4.5 on the *fluid estimator* is adapted from [258, 255]. Theory supporting these algorithms is contained in [364, 363, 314].

The histograms shown in Figures 11.1 and 11.8 were provided as a gift from Prof. Ken Duffy.

Veatch in [480] explores bounded perturbations of the fluid value function in approximate dynamic programming. Related techniques are considered in current research to refine the fluid estimator.

Proposition 11.4.1 is a variant on established control variate techniques (see [333].) The selection of an optimal control variate requires knowledge of a covariance matrix Σ, such as given in (11.43). While it is true that this can be estimated using Monte-Carlo or a version of TD learning, there is the danger of increased variance associated with the additional estimation of Σ [333]. The "loss factor" is discussed in [332, 382] for terminating simulations, and in [340] for steady-state simulation. It is argued that estimation should be performed "off-line" based on a large amount of data. Once an estimate of Σ is obtained, this is held fixed for application in subsequent simulations.

The reader is referred to the monographs [464, 53] for general background on TD learning. More detailed treatments can be found in [474, 418, 381, 310, 49] and the references that follow.

The Least-Squares Temporal Difference Learning algorithm (LSTD) was introduced for the discounted-cost value function in Bradtke and Barto [82]. The regeneration approach to average-cost TD learning in Section 11.5.4 is based on [310]. Methods to *construct* a basis $\{\psi_i\}$ based on observations of a Markov model are described in [474, 350].

The TD learning algorithm is typically presented in a modified form. For example, in the discounted-cost case the eligibility vectors defined in (11.82) are modified through the introduction of a 'forgetting factor' $\lambda \in [0, 1]$,

$$\varphi(k+1) = (1+\gamma)^{-1}[\lambda\varphi(k) + \psi(X(k+1))], \qquad \varphi(0) = 0.$$

The resulting algorithm (11.83) is called TD(λ), where the definition of the temporal differences remain unchanged. Under general conditions, the algorithm remains convergent to some $\theta(\infty) \in \mathbb{R}^{\ell_h}$, but it is no longer consistent. That is, in general $\theta(\infty) \neq \theta^*$, although bounds on the error $\|\theta(\infty) - \theta^*\|_\pi$ as a function of λ can be constructed [474, 290].

The introduction of λ is difficult to justify, given the fact that variance reduction can be achieved through state-weighting, as in (11.88), and this approach does not introduce bias.

Optimization of control variates in i.i.d. models is part of the original formulation of this technique. In the Markov setting this step is substantially more difficult and solutions are more recent. The first algorithms for this purpose were introduced in [465, 301]. The LSTD algorithm for shadow function optimization contained in Section 11.5.5 is new, and this is the first such algorithm that is asymptotically consistent.

Revise below!! And use $\widehat{\eta}^{\text{w}}$.