Chapter 9

On-policy Prediction with Approximation

In this chapter, we begin our study of function approximation in reinforcement learning by considering its use in estimating the state-value function from on-policy data, that is, in approximating v_π from experience generated using a known policy π. The novelty in this chapter is that the approximate value function is represented not as a table but as a parameterized functional form with parameter vector $\boldsymbol{\theta} \in \mathbb{R}^n$. We will write $\hat{v}(s,\boldsymbol{\theta}) \approx v_\pi(s)$ for the approximated value of state s given parameter vector $\boldsymbol{\theta}$. For example, $\hat{v}$ might be a linear function in features of the state, with $\boldsymbol{\theta}$ the vector of feature weights. More generally, $\hat{v}$ might be the function computed by a multi-layer artificial neural network, with $\boldsymbol{\theta}$ the vector of connection weights in all the layers. By adjusting the weights, any of a wide range of different functions can be implemented by the network. Or $\hat{v}$ might be the function computed by a decision tree, where $\boldsymbol{\theta}$ is all the parameters defining the split points and leaf values of the tree. Typically, the number of parameters (the number of components of $\boldsymbol{\theta}$) is much less than the number of states ($n \ll |\mathcal{S}|$), and changing one parameter changes the estimated value of many states. Consequently, when a single state is updated, the change generalizes from that state to affect the values of many other states. Such *generalization* makes the learning potentially more powerful but also potentially more difficult to manage and understand.

9.1 Value-function Approximation

All of the prediction methods covered in this book have been described as backups, that is, as updates to an estimated value function that shift its value at particular states toward a "backed-up value" for that state. Let us refer to an individual backup by the notation $s \mapsto g$, where s is the state backed up and g is the backed-up value, or target, that s's estimated value is shifted toward. For example, the Monte Carlo backup for value prediction is $S_t \mapsto G_t$, the TD(0) backup is $S_t \mapsto R_{t+1}+\gamma\hat{v}(S_{t+1},\boldsymbol{\theta}_t)$, and the n-step TD backup is $S_t \mapsto G_t^{(n)}$. In the DP policy-evaluation backup,

$s \mapsto \mathbb{E}_\pi[R_{t+1} + \gamma\hat{v}(S_{t+1},\boldsymbol{\theta}_t) \mid S_t = s]$, an arbitrary state s is backed up, whereas in the other cases the state encountered in actual experience, S_t, is backed up.

It is natural to interpret each backup as specifying an example of the desired input–output behavior of the value function. In a sense, the backup $s \mapsto g$ means that the estimated value for state s should be more like the number g. Up to now, the actual update implementing the backup has been trivial: the table entry for s's estimated value has simply been shifted a fraction of the way toward g, and the estimated values of all other states were left unchanged. Now we permit arbitrarily complex and sophisticated methods to implement the backup, and updating at s generalizes so that the estimated values of many other states are changed as well. Machine learning methods that learn to mimic input–output examples in this way are called *supervised learning* methods, and when the outputs are numbers, like g, the process is often called *function approximation*. Function approximation methods expect to receive examples of the desired input–output behavior of the function they are trying to approximate. We use these methods for value prediction simply by passing to them the $s \mapsto g$ of each backup as a training example. We then interpret the approximate function they produce as an estimated value function.

Viewing each backup as a conventional training example in this way enables us to use any of a wide range of existing function approximation methods for value prediction. In principle, we can use any method for supervised learning from examples, including artificial neural networks, decision trees, and various kinds of multivariate regression. However, not all function approximation methods are equally well suited for use in reinforcement learning. The most sophisticated neural network and statistical methods all assume a static training set over which multiple passes are made. In reinforcement learning, however, it is important that learning be able to occur online, while interacting with the environment or with a model of the environment. To do this requires methods that are able to learn efficiently from incrementally acquired data. In addition, reinforcement learning generally requires function approximation methods able to handle nonstationary target functions (target functions that change over time). For example, in GPI control methods we often seek to learn q_π while π changes. Even if the policy remains the same, the target values of training examples are nonstationary if they are generated by bootstrapping methods (DP and TD learning). Methods that cannot easily handle such nonstationarity are less suitable for reinforcement learning.

9.2 The Prediction Objective (MSVE)

Up to now we have not specified an explicit objective for prediction. In the tabular case a continuous measure of prediction quality was not necessary because the learned value function could come to equal the true value function exactly. Moreover, the learned values at each state were decoupled—an update at one state affected no other. But with genuine approximation, an update at one state affects many others, and it is not possible to get all states exactly correct. By assumption we have far more states than parameters, so making one state's estimate more accurate invariably

means making others' less accurate. We are obligated then to say which states we care most about. We must specify a weighting or distribution $d(s) \geq 0$ representing how much we care about the error in each state s. By the error in a state s we mean the square of the difference between the approximate value $\hat{v}(s,\boldsymbol{\theta})$ and the true value $v_\pi(s)$. Weighting this over the state space by the distribution d, we obtain a natural objective function, the *Mean Square Value Error*, or MSVE:

$$\text{MSVE}(\boldsymbol{\theta}) \doteq \sum_{s\in\mathcal{S}} d(s)\Big[v_\pi(s) - \hat{v}(s,\boldsymbol{\theta})\Big]^2. \tag{9.1}$$

The square root of this measure, the RMSVE, gives a rough measure of how much the approximate values differ from the true values and is often used in plots. Typically one chooses $d(s)$ to be the fraction of time spent in s under the target policy π. This is called the *on-policy distribution*; we focus entirely on this case in this chapter. In continuing tasks, the on-policy distribution is the stationary distribution under π.

The on-policy distribution in episodic tasks

In an episodic task, the on-policy distribution is a little different in that it depends on how the initial states of episodes are chosen. Let $h(s)$ denote the probability that an episode begins in each state s, and let $\eta(s)$ denote the number of time steps spent, on average, in state s in a single episode. Time is spent in a state s if episodes start in it, or if transitions are made into it from a state $\bar{s}$ in which time is spent:

$$\eta(s) = h(s) + \sum_{\bar{s}} \eta(\bar{s}) \sum_a \pi(a|\bar{s}) p(s|\bar{s},a). \tag{9.2}$$

This can be solved in vector-matrix form for the expected visitation times $\eta(s)$, from which the on-policy distribution is obtained by normalizing so that it sums to one:

$$d(s) = \frac{\eta(s)}{\sum_{s'} \eta(s')}. \tag{9.3}$$

The two cases, continuing and episodic, behave similarly, but with approximation they must be treated separately in formal analyses, as we will see repeatedly in this part of the book. This completes the specification of the learning objective.

It is not completely clear that the MSVE is the right performance objective for reinforcement learning. Remember that our ultimate purpose, the reason we are learning a value function, is to use it in finding a better policy. The best value function for this purpose is not necessarily the best for minimizing MSVE. Nevertheless, it is not yet clear what a more useful alternative goal for value prediction might be. For now, we will focus on MSVE.

An ideal goal in terms of MSVE would be to find a *global optimum*, a parameter vector $\boldsymbol{\theta}^*$ for which $\text{MSVE}(\boldsymbol{\theta}^*) \leq \text{MSVE}(\boldsymbol{\theta})$ for all possible $\boldsymbol{\theta}$. Reaching this goal is sometimes possible for simple function approximators such as linear ones, but is rarely

possible for complex function approximators such as artificial neural networks and decision trees. Short of this, complex function approximators may seek to converge instead to a *local optimum*, a parameter vector $\boldsymbol{\theta}^*$ for which $\text{MSVE}(\boldsymbol{\theta}^*) \leq \text{MSVE}(\boldsymbol{\theta})$ for all $\boldsymbol{\theta}$ in some neighborhood of $\boldsymbol{\theta}^*$. Although this guarantee is only slightly reassuring, it is typically the best that can be said for nonlinear function approximators, and often it is enough. Still, for many cases of interest in reinforcement learning, convergence to an optimum, or even to within a bounded distance from an optimum cannot be assured. Some methods may in fact diverge, with their MSVE approaching infinity in the limit.

In the last two sections we have outlined a framework for combining a wide range of reinforcement learning methods for value prediction with a wide range of function approximation methods, using the backups of the former to generate training examples for the latter. We have also described a MSVE performance measure which these methods may aspire to minimize. The range of possible methods is far too large to cover all, and anyway too little is known about most of them to make a reliable evaluation or recommendation. Of necessity, we consider only a few possibilities. In the rest of this chapter we focus on function approximation methods based on gradient principles, and on linear gradient-descent methods in particular. We focus on these methods in part because we consider them to be particularly promising and because they reveal key theoretical issues, but also because they are simple and our space is limited.

9.3 Stochastic-gradient and Semi-gradient Methods

We now develop in detail one class of learning methods for function approximation in value prediction, those based on stochastic gradient descent (SGD). SGD methods are among the most widely used of all function approximation methods and are particularly well suited to online reinforcement learning.

In gradient-descent methods, the parameter vector is a column vector with a fixed number of real valued components, $\boldsymbol{\theta} \doteq (\theta_1, \theta_2, \ldots, \theta_n)^\top$,[1] and the approximate value function $\hat{v}(s,\boldsymbol{\theta})$ is a smooth differentiable function of $\boldsymbol{\theta}$ for all $s \in \mathcal{S}$. We will be updating $\boldsymbol{\theta}$ at each of a series of discrete time steps, $t = 0, 1, 2, 3, \ldots$, so we will need a notation $\boldsymbol{\theta}_t$ for the parameter vector at each step. For now, let us assume that, on each step, we observe a new example $S_t \mapsto v_\pi(S_t)$ consisting of a (possibly randomly selected) state S_t and its true value under the policy. These states might be successive states from an interaction with the environment, but for now we do not assume so. Even though we are given the exact, correct values, $v_\pi(S_t)$ for each S_t, there is still a difficult problem because our function approximator has limited resources and thus limited resolution. In particular, there is generally no $\boldsymbol{\theta}$ that gets all the states, or even all the examples, exactly correct. In addition, we must generalize to all the other states that have not appeared in examples.

[1] The $^\top$ denotes transpose, needed here to turn the horizontal row vector in the text into a vertical column vector; in this book all vectors are column vectors unless transposed.

We assume that states appear in examples with the same distribution, d, over which we are trying to minimize the MSVE as given by (9.1). A good strategy in this case is to try to minimize error on the observed examples. *Stochastic gradient-descent* (SGD) methods do this by adjusting the parameter vector after each example by a small amount in the direction that would most reduce the error on that example:

$$\boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t - \frac{1}{2}\alpha\nabla\Big[v_\pi(S_t) - \hat{v}(S_t,\boldsymbol{\theta}_t)\Big]^2 \tag{9.4}$$

$$= \boldsymbol{\theta}_t + \alpha\Big[v_\pi(S_t) - \hat{v}(S_t,\boldsymbol{\theta}_t)\Big]\nabla\hat{v}(S_t,\boldsymbol{\theta}_t), \tag{9.5}$$

where α is a positive step-size hyperparameter, and $\nabla f(\boldsymbol{\theta})$, for any scalar expression $f(\boldsymbol{\theta})$, denotes the vector of partial derivatives with respect to the components of the parameter vector:

$$\nabla f(\boldsymbol{\theta}) \doteq \left(\frac{\partial f(\boldsymbol{\theta})}{\partial\theta_1}, \frac{\partial f(\boldsymbol{\theta})}{\partial\theta_2}, \ldots, \frac{\partial f(\boldsymbol{\theta})}{\partial\theta_n}\right)^\top. \tag{9.6}$$

This derivative vector is the *gradient* of f with respect to $\boldsymbol{\theta}$. SGD methods are "gradient descent" methods because the overall step in $\boldsymbol{\theta}_t$ is proportional to the negative gradient of the example's squared error (9.4). This is the direction in which the error falls most rapidly. Gradient descent methods are called "stochastic" when the update is done, as here, on only a single example, which might have been selected stochastically. Over many examples, making small steps, the overall effect is to minimize an average performance measure such as the MSVE.

It may not be immediately apparent why SGD takes only a small step in the direction of the gradient. Could we not move all the way in this direction and completely eliminate the error on the example? In many cases this could be done, but usually it is not desirable. Remember that we do not seek or expect to find a value function that has zero error for all states, but only an approximation that balances the errors in different states. If we completely corrected each example in one step, then we would not find such a balance. In fact, the convergence results for SGD methods assume that α decreases over time. If it decreases in such a way as to satisfy the standard stochastic approximation conditions (2.7), then the SGD method (9.5) is guaranteed to converge to a local optimum.

We turn now to the case in which the target output, here denoted $U_t \in \mathbb{R}$, of the tth training example, $S_t \mapsto U_t$, is not the true value, $v_\pi(S_t)$, but some, possibly random, approximation to it. For example, U_t might be a noise-corrupted version of $v_\pi(S_t)$, or it might be one of the bootstrapping targets using $\hat{v}$ mentioned in the previous section. In these cases we cannot perform the exact update (9.5) because $v_\pi(S_t)$ is unknown, but we can approximate it by substituting U_t in place of $v_\pi(S_t)$. This yields the following general SGD method for state-value prediction:

$$\boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t + \alpha\Big[U_t - \hat{v}(S_t,\boldsymbol{\theta}_t)\Big]\nabla\hat{v}(S_t,\boldsymbol{\theta}_t). \tag{9.7}$$

If U_t is an *unbiased* estimate, that is, if $\mathbb{E}[U_t] = v_\pi(S_t)$, for each t, then $\boldsymbol{\theta}_t$ is guaranteed to converge to a local optimum under the usual stochastic approximation conditions (2.7) for decreasing α.

Gradient Monte Carlo for approximating $\hat{v} \approx v_\pi$

Input: the policy π to be evaluated
Input: a differentiable function $\hat{v} : \mathcal{S} \times \mathbb{R}^n \to \mathbb{R}$

Initialize value-function parameter $\boldsymbol{\theta}$ as appropriate (e.g., $\boldsymbol{\theta} = \mathbf{0}$)
Repeat forever:
  Generate an episode $S_0, A_0, R_1, S_1, A_1, \ldots, R_T, S_T$ using π
  For $t = 0, 1, \ldots, T-1$:
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha\big[G_t - \hat{v}(S_t,\boldsymbol{\theta})\big]\nabla\hat{v}(S_t,\boldsymbol{\theta})$

For example, suppose the states in the examples are the states generated by interaction (or simulated interaction) with the environment using policy π. Because the true value of a state is the expected value of the return following it, the Monte Carlo target $U_t \doteq G_t$ is by definition an unbiased estimate of $v_\pi(S_t)$. With this choice, the general SGD method (9.7) converges to a locally optimal approximation to $v_\pi(S_t)$. Thus, the gradient-descent version of Monte Carlo state-value prediction is guaranteed to find a locally optimal solution. Pseudocode for a complete algorithm is shown in the box.

One does not obtain the same guarantees if a bootstrapping estimate of $v_\pi(S_t)$ is used as the target U_t in (9.7). Bootstrapping targets such as n-step returns $G_t^{(n)}$ or the DP target $\sum_{a,s',r} \pi(a|S_t)p(s',r|S_t,a)[r + \gamma\hat{v}(s',\boldsymbol{\theta}_t)]$ all depend on the current value of the parameter $\boldsymbol{\theta}_t$, which implies that they will be biased and that they will not produce a true gradient-descent method. One way to look at this is that the key step from (9.4) to (9.5) relies on the target being independent of $\boldsymbol{\theta}_t$. This step would not be valid if a bootstrapping estimate was used in place of $v_\pi(S_t)$. Bootstrapping methods are not in fact instances of true gradient descent (Barnard, 1993). They take into account the effect of changing the parameter $\boldsymbol{\theta}_t$ on the estimate, but ignore its effect on the target. They include only a part of the gradient and, accordingly, we call them *semi-gradient methods*.

Although semi-gradient (bootstrapping) methods do not converge as robustly as gradient methods, they do converge reliably in important cases such as the linear case discussed in the next section. Moreover, they offer important advantages which makes them often clearly preferred. One reason for this is that they are typically significantly faster to learn, as we have seen in Chapters 6 and 7. Another is that they enable learning to be continual and online, without waiting for the end of an episode. This enables them to be used on continuing problems and provides computational advantages. A prototypical semi-gradient method is semi-gradient TD(0), which uses $U_t \doteq R_{t+1} + \gamma\hat{v}(S_{t+1},\boldsymbol{\theta})$ as its target. Complete pseudocode for this method is given in the box at the top of the next page.

Semi-gradient TD(0) for estimating $\hat{v} \approx v_\pi$

Input: the policy π to be evaluated
Input: a differentiable function $\hat{v} : \mathcal{S}^+ \times \mathbb{R}^n \to \mathbb{R}$ such that $\hat{v}(\text{terminal},\cdot) = 0$

Initialize value-function parameter $\boldsymbol{\theta}$ arbitrarily (e.g., $\boldsymbol{\theta} = \mathbf{0}$)
Repeat (for each episode):
  Initialize S
  Repeat (for each step of episode):
    Choose $A \sim \pi(\cdot|S)$
    Take action A, observe R, S'
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha\big[R + \gamma\hat{v}(S',\boldsymbol{\theta}) - \hat{v}(S,\boldsymbol{\theta})\big]\nabla\hat{v}(S,\boldsymbol{\theta})$
    $S \leftarrow S'$
  until S' is terminal

Example 9.1: State Aggregation on the 1000-state Random Walk *State aggregation* is a simple form of generalizing function approximation in which states are grouped together, with one estimated value (one component of the parameter vector $\boldsymbol{\theta}$) for each group. The value of a state is estimated as its group's component, and when the state is updated, that component alone is updated. State aggregation is a special case of SGD (9.7) in which the gradient, $\nabla\hat{v}(S_t,\boldsymbol{\theta}_t)$, is 1 for S_t's group's component and 0 for the other components.

Consider a 1000-state version of the random walk task (Examples 6.2 and 7.1). The states are numbered from 1 to 1000, left to right, and all episodes begin near the center, in state 500. State transitions are from the current state to one of the 100 states to its left, or to one of the 100 states to its right, all with equal probability. If the current state is near an edge, with fewer that 100 neighbors on one side, a transition occurs to the terminal state with probability equal to that which would have been given to the missing neighbor states (thus, state 1 has a 0.5 chance of terminating, and state 2 has a .495 chance of terminating). As usual, termination on the left produces a reward of -1, and termination on the right produces a reward of $+1$. All other transitions have a reward of zero. We use this task as a running example throughout this section.

Figure 9.1 shows the true value function v_π for this task. It is nearly a straight line, but tilted slightly toward the horizontal and curving further in this direction for the last 100 states at each end. Also shown is the final approximate value function learned by the gradient Monte-Carlo algorithm with state aggregation after 100,000 episodes with a step size of $\alpha = 2 \times 10^{-5}$. For the state aggregation, the 1000 states were partitioned into 10 groups of 100 states each (i.e., states 1–100 were one group, states 101-200 were another, and so on). The staircase effect shown in the figure is typical of state aggregation; within each group, the approximate value is constant, and it changes abruptly from one group to the next. These approximate values are close to the global minimum of the MSVE (9.1).

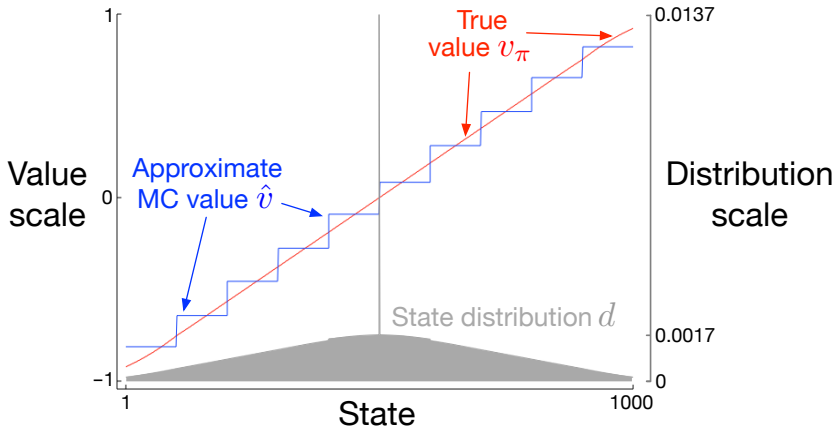


Figure 9.1: Function approximation by state aggregation on the 1000-state random walk task, using the gradient Monte Carlo algorithm (page 190).

Some of the details of the approximate values are best appreciated by reference to the state distribution d for this task, shown in the lower portion of the figure with a right-side scale. Recall that d is a weighting rather than a distribution in episodic tasks like this. Numerically it corresponds to the number of times each state is visited, on average, in a single episode. State 500, in the center, is the first state of every episode, but it is rarely visited again. On average, it is visited about 1.14 times per episode. The states reachable in one step from the start state are the second most visited, about 0.14 times per episode each. From there d falls off almost linearly, reaching about 0.012 at the extreme states 1 and 1000. The start state falls within the 401-500 group just to the left of the center. This state is given much more weight in the approximation that the other states in the group. Since the correct value for it is almost zero (because it is almost at the center), its group's value should be slightly higher that the unweighted average of the true values within the group, and indeed it appears to be so. The effect is small, however, presumably because the other states in the group are so many. A larger effect to a similar result is most evident in the leftmost groups, whose values are clearly shifted higher than the unweighted average of the true values of states within the group, and in the rightmost groups, whose values are clearly shifted lower. This is due to the states in these areas having the greatest asymmetry in their weightings by d. For example, in the leftmost group, state 99 is weighted more than 3 times more strongly than state 0. Thus the estimate for the group is biased toward the true value of state 99, which is higher than the true value of state 0. ■

9.4 Linear Methods

One of the most important special cases of function approximation is that in which the approximate function, $\hat{v}(\cdot,\boldsymbol{\theta})$, is a linear function of the parameter vector, $\boldsymbol{\theta}$.

Corresponding to every state s, there is a real-valued vector of features $\boldsymbol{\phi}(s) \doteq (\phi_1(s), \phi_2(s), \ldots, \phi_n(s))^\top$, with the same number of components as $\boldsymbol{\theta}$. The features may be constructed from the states in many different ways; we cover a few possibilities in the next sections. However the features are constructed, the approximate state-value function is given by

$$\hat{v}(s,\boldsymbol{\theta}) \doteq \boldsymbol{\theta}^\top \boldsymbol{\phi}(s) \doteq \sum_{i=1}^{n} \theta_i \phi_i(s). \tag{9.8}$$

In this case the approximate value function is said to be *linear in the parameters*, or simply *linear*. The individual functions $\phi_i : \mathcal{S} \to \mathbb{R}$ are called *basis functions* because they form a linear basis for the set of approximate functions of this form. Constructing n-dimensional feature vectors to represent states is the same as selecting a set of n basis functions.

It is natural to use SGD updates with linear function approximation. The gradient of the approximate value function with respect to $\boldsymbol{\theta}$ in this case is

$$\nabla \hat{v}(s,\boldsymbol{\theta}) = \boldsymbol{\phi}(s).$$

Thus, the general SGD update (9.7) reduces to a particularly simple form in the linear case.

Because it is so simple, the linear SGD case is one of the most favorable for mathematical analysis. Almost all useful convergence results for learning systems of all kinds are for linear (or simpler) function approximation methods.

In particular, in the linear case there is only one optimum (or, in degenerate cases, one set of equally good optima), and thus any method that is guaranteed to converge to or near a local optimum is automatically guaranteed to converge to or near the global optimum. For example, the gradient Monte Carlo algorithm presented in the previous section converges to the global optimum of the MSVE under linear function approximation if α is reduced over time according to the usual conditions.

The semi-gradient TD(0) algorithm presented in the previous section also converges under linear function approximation, but this does not follow from general results on SGD; a separate theorem is necessary. The parameter converged to is also not the global optimum, but rather to a point near the local optimum. It is useful to consider this important case in more detail, specifically for the continuing case. The update at each time t is

$$\begin{aligned} \boldsymbol{\theta}_{t+1} &\doteq \boldsymbol{\theta}_t + \alpha\Big(R_{t+1} + \gamma \boldsymbol{\theta}_t^\top \boldsymbol{\phi}_{t+1} - \boldsymbol{\theta}_t^\top \boldsymbol{\phi}_t\Big)\boldsymbol{\phi}_t \\ &= \boldsymbol{\theta}_t + \alpha\Big(R_{t+1}\boldsymbol{\phi}_t - \boldsymbol{\phi}_t\big(\boldsymbol{\phi}_t - \gamma\boldsymbol{\phi}_{t+1}\big)^\top \boldsymbol{\theta}_t\Big), \end{aligned} \tag{9.9}$$

where here we have used the notational shorthand $\boldsymbol{\phi}_t = \boldsymbol{\phi}(S_t)$. Once the system has reached steady state, for any given $\boldsymbol{\theta}_t$, the expected next parameter vector can be written

$$\mathbb{E}[\boldsymbol{\theta}_{t+1} | \boldsymbol{\theta}_t] = \boldsymbol{\theta}_t + \alpha(\mathbf{b} - \mathbf{A}\boldsymbol{\theta}_t), \tag{9.10}$$

where

$$\mathbf{b} \doteq \mathbb{E}[R_{t+1}\boldsymbol{\phi}_t] \in \mathbb{R}^n \quad \text{and} \quad \mathbf{A} \doteq \mathbb{E}\Big[\boldsymbol{\phi}_t\big(\boldsymbol{\phi}_t - \gamma\boldsymbol{\phi}_{t+1}\big)^\top\Big] \in \mathbb{R}^n \times \mathbb{R}^n \tag{9.11}$$

From (9.10) it is clear that, if the system converges, it must converge to the parameter $\boldsymbol{\theta}_{TD}$ at which

$$\begin{aligned} & \mathbf{b} - \mathbf{A}\boldsymbol{\theta}_{TD} = \mathbf{0} \\ \Rightarrow \quad & \mathbf{b} = \mathbf{A}\boldsymbol{\theta}_{TD} \\ \Rightarrow \quad & \boldsymbol{\theta}_{TD} \doteq \mathbf{A}^{-1}\mathbf{b}. \end{aligned} \tag{9.12}$$

This quantity is called the *TD fixpoint*. In fact linear semi-gradient TD(0) converges to this point. Some of the theory proving its convergence, and the existence of the inverse above, is given in the box.

Proof of Convergence of Linear TD(0)

What properties assure convergence of the linear TD(0) algorithm (9.9)? Some insight can be gained by rewriting (9.10) as

$$\mathbb{E}[\boldsymbol{\theta}_{t+1}|\boldsymbol{\theta}_t] = (\mathbf{I} - \alpha\mathbf{A})\boldsymbol{\theta}_t + \alpha\mathbf{b}. \tag{9.13}$$

Note that the matrix $\mathbf{A}$ multiplies the parameter $\boldsymbol{\theta}_t$ and not $\mathbf{b}$; only $\mathbf{A}$ is important to convergence. To develop intuition, consider the special case in which $\mathbf{A}$ is a diagonal matrix. If any of the diagonal elements are negative, then the corresponding diagonal element of $\mathbf{I} - \alpha\mathbf{A}$ will be greater than one, and the corresponding component of $\boldsymbol{\theta}_t$ will be amplified, which will lead to divergence if continued. On the other hand, if the diagonal elements of $\mathbf{A}$ are all positive, then α can be chosen smaller than one over the largest of them, such that $\mathbf{I} - \alpha\mathbf{A}$ is diagonal with all diagonal elements between 0 and 1. In this case the first term of the update tends to shrink $\boldsymbol{\theta}_t$, and stability is assured. In general case, $\boldsymbol{\theta}_t$ will be reduced toward zero whenever $\mathbf{A}$ is *positive definite*, meaning $y^\top\mathbf{A}y > 0$ for real vector y. Positive definiteness also ensures that the inverse $\mathbf{A}^{-1}$ exists.

For linear TD(0), in the continuing case with $\gamma < 1$, the $\mathbf{A}$ matrix (9.11) can be written

$$\begin{aligned} \mathbf{A} &= \sum_s d(s) \sum_a \pi(a|s) \sum_{r,s'} p(r, s'|s, a)\boldsymbol{\phi}(s)\big(\boldsymbol{\phi}(s) - \gamma\boldsymbol{\phi}(s')\big)^\top \\ &= \sum_s d(s) \sum_{s'} p(s'|s)\boldsymbol{\phi}(s)\big(\boldsymbol{\phi}(s) - \gamma\boldsymbol{\phi}(s')\big)^\top \\ &= \sum_s d(s)\boldsymbol{\phi}(s)\Big(\boldsymbol{\phi}(s) - \gamma\sum_{s'} p(s'|s)\boldsymbol{\phi}(s')\Big)^\top \\ &= \boldsymbol{\Phi}^\top\mathbf{D}(\mathbf{I} - \gamma\mathbf{P})\boldsymbol{\Phi}, \end{aligned}$$

where $d(s)$ is the stationary distribution under π, $p(s'|s)$ is the probability of transition from s to s' under policy π, $\mathbf{P}$ is the $|\mathcal{S}| \times |\mathcal{S}|$ matrix of these probabilities, $\mathbf{D}$ is the $|\mathcal{S}| \times |\mathcal{S}|$ diagonal matrix with the $d(s)$ on its diagonal, and $\mathbf{\Phi}$ is the $|\mathcal{S}| \times n$ matrix with $\boldsymbol{\phi}(s)$ as its rows. From here it is clear that the inner matrix $\mathbf{D}(\mathbf{I} - \gamma\mathbf{P})$ is key to determining the positive definiteness of $\mathbf{A}$.

For a key matrix of this type, positive definiteness is assured if all of its columns sum to a nonnegative number. This was shown by Sutton (1988, p. 27) based on two previously established theorems. One theorem says that any matrix $\mathbf{M}$ is positive definite if and only if the symmetric matrix $\mathbf{S} = \mathbf{M} + \mathbf{M}^\top$ is positive definite (Sutton 1988, appendix). The second theorem says that any symmetric real matrix $\mathbf{S}$ is positive definite if all of its diagonal entries are positive and greater than the sum of the corresponding off-diagonal entries (Varga 1962, p. 23). For our key matrix, $\mathbf{D}(\mathbf{I} - \gamma\mathbf{P})$, the diagonal entries are positive and the off-diagonal entries are negative, so all we have to show is that each row sum plus the corresponding column sum is positive. The row sums are all positive because $\mathbf{P}$ is a stochastic matrix and $\gamma < 1$. Thus it only remains to show that the column sums are nonnegative. Note that the row vector of the column sums of any matrix $\mathbf{M}$ can be written as $\mathbf{1}^\top\mathbf{M}$, where $\mathbf{1}$ is the column vector with all components equal to 1. Let $\mathbf{d}$ denote the $|\mathcal{S}|$-vector of the $d(s)$. The column sums of our key matrix, then, are:

$$\begin{aligned}
\mathbf{1}^\top\mathbf{D}(\mathbf{I} - \gamma\mathbf{P}) &= \mathbf{d}^\top(\mathbf{I} - \gamma\mathbf{P}) \\
&= \mathbf{d}^\top - \gamma\mathbf{d}^\top\mathbf{P} \\
&= \mathbf{d}^\top - \gamma\mathbf{d}^\top \qquad \text{(because } \mathbf{d} \text{ is the stationary distribution)} \\
&= (1 - \gamma)\mathbf{d},
\end{aligned}$$

all components of which are positive. Thus, the key matrix and its $\mathbf{A}$ matrix are positive definite, and on-policy TD(0) is stable. (Additional conditions and a schedule for reducing α over time are needed to prove convergence with probability one.)

At the TD fixpoint, it has also been proven (in the continuing case) that the MSVE is within a bounded expansion of the lowest possible error:

$$\text{MSVE}(\boldsymbol{\theta}_{TD}) \leq \frac{1}{1-\gamma} \min_{\boldsymbol{\theta}} \text{MSVE}(\boldsymbol{\theta}). \tag{9.14}$$

That is, the asymptotic error of the TD method is no more than $\frac{1}{1-\gamma}$ times the smallest possible error, that attained in the limit by the Monte Carlo method. Because γ is often near one, this expansion factor can be quite large, so there is substantial potential loss in asymptotic performance with the TD method. On the other hand, recall that the TD methods are often of vastly reduced variance compared to Monte Carlo methods, and thus faster, as we saw in Chapters 6 and 7. Which method will be best depends on the nature of the approximation and problem, and on how long

learning contiunues.

A bound analogous to (9.14) applies to other on-policy bootstrapping methods as well. For example, linear semi-gradient DP (Eq. 9.7 with $U_t \doteq \sum_a \pi(a|S_t) \sum_{s',r} p(s',r|S_t,a)[r+\gamma\hat{v}(s',\boldsymbol{\theta}_t)]$) with backups according the on-policy distribution will also converge to the TD fixpoint. One-step semi-gradient *action-value* methods, such as semi-gradient Sarsa(0) covered in the next chapter converge to analogous fixpoint and an analogous bound. For episodic tasks, there is a slightly different but related bound (see Bertsekas and Tsitsiklis, 1996). There are also a few technical conditions on the rewards, features, and decrease in the step-size parameter, which we have omitted here. The full details can be found in the original paper (Tsitsiklis and Van Roy, 1997).

Critical to the these convergence results is that states are backed up according to the on-policy distribution. For other backup distributions, bootstrapping methods using function approximation may actually diverge to infinity. Examples of this and a discussion of possible solution methods are given in Chapter 11.

Example 9.2: Bootstrapping on the 1000-state Random Walk State aggregation is a special case of linear function approximation, so let's return to the 1000-state random walk to illustrate some of the observations made in this chapter. The left panel of Figure 9.2 shows the final value function learned by the semi-gradient TD(0) algorithm (page 191) using the same state aggregation as in Example 9.1. We see that the near-asymptotic TD approximation is indeed farther from the true values than the Monte Carlo approximation shown in Figure 9.1.

Nevertheless, TD methods retain large potential advantages in learning rate, and generalize MC methods, as we investigated fully with the multi-step TD methods of Chapter 7. The right panel of Figure 9.2 shows results with an n-step semi-gradient TD method using state aggregation and the 1000-state random walk that are strikingly similar to those we obtained earlier with tabular methods and the 19-state

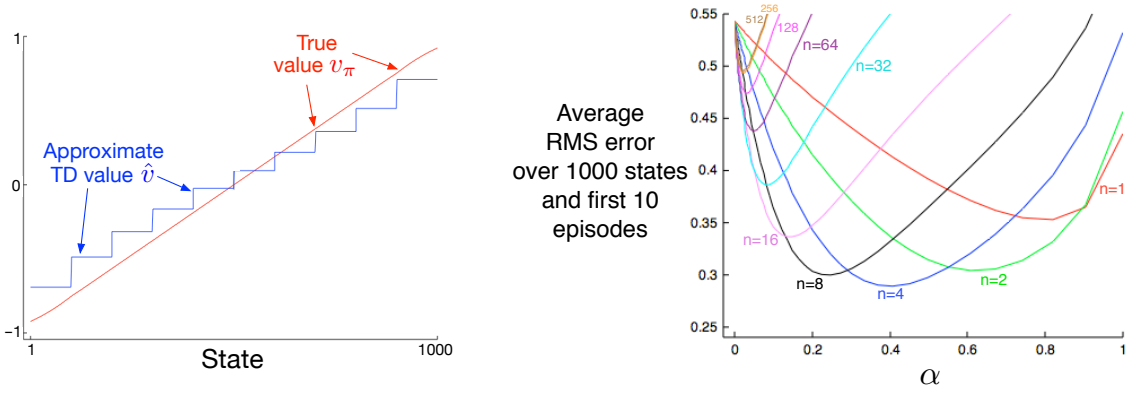


Figure 9.2: Bootstrapping with state aggregation on the 1000-state random walk task. Left: Asymptotic values of semi-gradient TD are worse than the asymptotic MC values in Figure 9.1. Right: Performance of n-step methods with state-aggregation are strikingly similar to those with tabular representations (cf. Figure 7.2).

n-step semi-gradient TD for estimating $\hat{v} \approx v_\pi$

Input: the policy π to be evaluated
Input: a differentiable function $\hat{v} : \mathcal{S}^+ \times \mathbb{R}^n \to \mathbb{R}$ such that $\hat{v}(\text{terminal},\cdot) = 0$
Hyperparameters: step size $\alpha \in (0, 1]$, a positive integer n

Initialize value-function parameter $\boldsymbol{\theta}$ arbitrarily (e.g., $\boldsymbol{\theta} = \mathbf{0}$)
Repeat (for each episode):
  Initialize and store $S_0 \neq$ terminal
  For $t = 0, 1, 2, \ldots$:
  | Take an action according to $\pi(\cdot|S_t)$
  | Observe and store the next reward as R_{t+1} and the next state as S_{t+1}
  | $\tau \leftarrow t - n + 1$ (τ is a time whose state's estimate might be updated)
  | If $\tau \geq 0$:
  | $G \leftarrow \sum_{i=\tau+1}^{\tau+n} \gamma^{i-\tau-1} R_i \; + \; \gamma^n \hat{v}(S_{\tau+n},\boldsymbol{\theta})$ $(G_\tau^{(n)})$
  | $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \left[G - \hat{v}(S_\tau,\boldsymbol{\theta})\right] \nabla \hat{v}(S_\tau,\boldsymbol{\theta})$
  Until S_{t+1} is terminal, then: $T \leftarrow t + 1$
  For $\tau = T - n + 1, \ldots, T - 1$
  $G \leftarrow \sum_{i=\tau+1}^{T} \gamma^{i-\tau-1} R_i$ (G_τ)
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \left[G - \hat{v}(S_\tau,\boldsymbol{\theta})\right] \nabla \hat{v}(S_\tau,\boldsymbol{\theta})$

random walk. To obtain such quantitatively similar results we switched the state aggregation to 20 groups of 50 states each. The 20 groups are then quantitatively close to the 19 states of the tabular problem. In particular, the state transitions of at-most 100 states to the right or left, or 50 states on average, were quantitively analogous to the single-state state transitions of the tabular system. To complete the match, we use here the same performance measure—an unweighted average of all the states over the first 10 episodes—rather than a MSVE objective as is otherwise more appropriate when using function approximation. ■

The semi-gradient n-step TD algorithm we used in this example is the natural extension of the tabular algorithm presented in Chapter 7 (on page 148) to semi-gradient function approximation, as given in the box above.

9.5 Feature Construction for Linear Methods

Linear methods are interesting because of their convergence guarantees, but also because in practice they can be very efficient in terms of both data and computation. Whether or not this is so depends critically on how the states are represented in terms of the features, which we investigate in this large section. Choosing features appropriate to the task is an important way of adding prior domain knowledge to reinforcement learning systems. Intuitively, the features should correspond to the natural features of the task, those along which generalization is most appropriate. If we are valuing geometric objects, for example, we might want to have features

for each possible shape, color, size, or function. If we are valuing states of a mobile robot, then we might want to have features for locations, degrees of remaining battery power, recent sonar readings, and so on.

In general, we also need features for combinations of these natural qualities. This is because the linear form prohibits the representation of interactions between features, such as the presence of feature i being good only in the absence of feature j. For example, in the pole-balancing task (Example 3.4), a high angular velocity may be either good or bad depending on the angular position. If the angle is high, then high angular velocity means an imminent danger of falling—a bad state—whereas if the angle is low, then high angular velocity means the pole is righting itself—a good state. In cases with such interactions one needs to introduce features for conjunctions of feature values when using linear function approximation methods. In the following subsections we consider a variety of general ways of doing this.

Exercise 9.1 How could we reproduce the tabular case within the linear framework?

Exercise 9.2 How could we reproduce the state aggregation case within the linear framework?

9.5.1 Polynomials

For multi-dimensional continuous state spaces, function approximation for reinforcement learning has much in common with the familiar tasks of interpolation and regression, which aim to define functions between and/or beyond given samples of function values. Various families of polynomials are commonly used for these tasks and can sometimes be good choices for reinforcement learning. Here we discuss only the most basic polynomial family.

Suppose a reinforcement learning problem's state space is two-dimensional so that each state is a real vector $s = (s_1, s_2)^\top$. You might choose to represent each s with the feature vector $(1, s_1, s_2, s_1 s_2)^\top$ in order to take the interaction of the state variables into account by weighting the product $s_1 s_2$ in an appropriate way. Or you might choose to use feature vectors like $(1, s_1, s_2, s_1 s_2, s_1^2, s_2^2, s_1 s_2^2, s_1^2 s_2, s_1^2 s_2^2)^\top$ to take more complex interactions into account. Using these features means that functions are approximated as multi-dimensional quadratic functions—even though the approximation is still linear in the parameters that have to be learned.

These example feature vectors are the result of selecting sets of polynomial basis functions, which are defined for any dimension and can encompass highly-complex interactions among the state variables:

For d state variables taking real values, every state s is a d-dimensional vector $(s_1, s_2, \ldots, s_d)^\top$ of real numbers. Each d-dimensional polynomial basis function ϕ_i can be written as

$$\phi_i(s) = \Pi_{j=1}^{d} s_j^{c_{i,j}}, \tag{9.15}$$

where each $c_{i,j}$ is an integer in the set $\{0, 1, \ldots, N\}$ for an integer $N \geq 0$. These

functions make up the order-N polynomial basis, which contains $(N+1)^d$ different functions.

Higher-order polynomial bases allow for more accurate approximations of more complicated functions. But because the number of functions in an order-N polynomial basis grows exponentially with the state space dimension (for $N > 0$), it is generally necessary to select a subset of them for function approximation. This can be done using prior beliefs about the nature of the function to be approximated, and some automated selection methods developed for polynomial regression can be adapted to deal with the incremental and nonstationary nature of reinforcement learning.

Exercise 9.3 Why does (9.15) define $(N+1)^d$ distinct functions for dimension d?

Exercise 9.4 Give N and the $c_{i,j}$ defining the basis functions that produce feature vectors $(1, s_1, s_2, s_1 s_2, s_1^2, s_2^2, s_1 s_2^2, s_1^2 s_2, s_1^2 s_2^2)^\top$.

9.5.2 Fourier Basis

Another linear function approximation method is based on the time-honored Fourier series, which expresses periodic functions as a weighted sum of sine and cosine basis basis functions of different frequencies. (A function f is periodic if $f(x) = f(x+T)$ for all x and some period T.) The Fourier series and the more general Fourier transform are widely used in applied sciences because—among many other reasons—if a function to be approximated is known, then the basis function weights are given by simple formulae and, with enough basis functions, essentially any function can be approximated as accurately as desired. In reinforcement learning, where the functions to be approximated are *unknown*, Fourier basis functions are of interest because they are easy to use and can perform well in a range of reinforcement learning problems. The Fourier basis has been developed in a simple form suitable for reinforcement learning problems with multi-dimensional continuous state spaces and functions that are not periodic.

First consider the one-dimensional case. The usual Fourier series representation of a function of one-dimension having period T represents the function as a linear combination of sine and cosine functions that are each periodic with periods that evenly divide T (in other words, whose frequencies are integer multiples of a fundamental frequency $1/T$). But if you are interested in approximating an aperiodic function defined over a bounded interval, you can use these Fourier basis functions with T set to the length the interval. The function of interest is then just one period of the periodic linear combination of the sine and cosine basis functions.

Furthermore, if you set T to twice the length of the interval of interest and restrict attention to the approximation over the half interval $[0, T/2]$, you can use just the cosine basis functions. This is possible because you can represent any *even* function, that is, any function that is symmetric about the origin, with just the cosine basis functions. So any function over the half-period $[0, T/2]$ can be approximated as

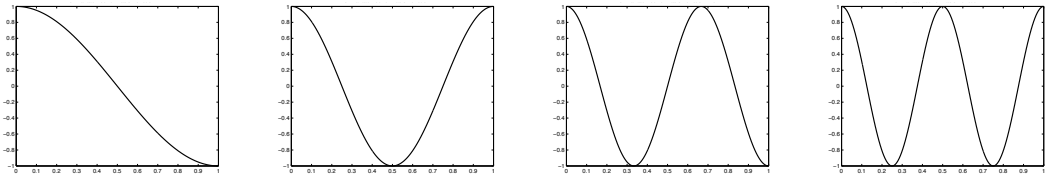


Figure 9.3: One-dimensonal Fourier cosine basis functions ϕ_i, $i = 1, 2, 3, 4$, for approximating functions over the interval $[0, 1]$; ϕ_0 is a constant function. From Konidaris et al. (2011), permission pending.

closely as desired with enough cosine basis functions. (Saying "any function" is not exactly correct because the function has to be mathematically well-behaved, but we skip this technicality here.) Similarly, it is possible to use just the sine basis functions, linear combinations of which are always *odd* functions, that is functions that are anti-symmetric about the origin. But it is generally better to keep just the cosine basis functions because "half-even" functions tend to be easier to approximate than "half-odd" functions since the latter are often discontinuous at the origin.

Following this logic and letting $T = 2$ so that the functions are defined over the half-T interval $[0, 1]$, the one-dimensional order-N Fourier cosine basis consists of the $N + 1$ functions

$$\phi_i(s) = \cos(i\pi s), \quad s \in [0, 1],$$

for $i = 0, \ldots, N$. Figure 9.3 shows one-dimensional Fourier cosine basis functions ϕ_i, for $i = 1, 2, 3, 4$; ϕ_0 is a constant function. Unlike polynomial basis functions, Fourier basis functions are always bounded and do not require exponentiation.

This same reasoning applies to the Fourier cosine series approximation in the multi-dimensional case:

> For a state space that is the d-dimensional unit hypercube with the origin in one corner, states are vectors $s = (s_1, \ldots, s_d)^\top$, $s_i \in [0, 1]$. Each function in the order-N Fourier cosine basis can be written
>
> $$\phi_i(s) = \cos(\pi c^i \cdot s), \tag{9.16}$$
>
> where $c^i = (c^i_1, \ldots, c^i_d)^\top$, with $c^i_j \in \{0, \ldots, N\}$ for $j = 1, \ldots, d$ and $i = 0, \ldots, (N + 1)^d$. This defines a function for each of the $(N + 1)^d$ possible integer vectors c^i. The dot-product $c^i \cdot s$ has the effect of assigning an integer in $\{0, \ldots, N\}$ to each dimension. As in the one-dimensional case, this integer determines the function's frequency along that dimension. The basis functions can of course be shifted and scaled to suit the bounded state space of a particular application.

As an example, consider the $d = 2$ case in which $s = (s_1, s_2)$, where each $c^i = (c^i_1, c^i_2)^\top$. Figure 9.4 shows a selection of 6 Fourier cosine basis functions, each labeled by the vector c^i that defines it (s_1 is horizontal axis and c^i is shown as a row vector with the index i omitted). Any zero in c means the function is constant along

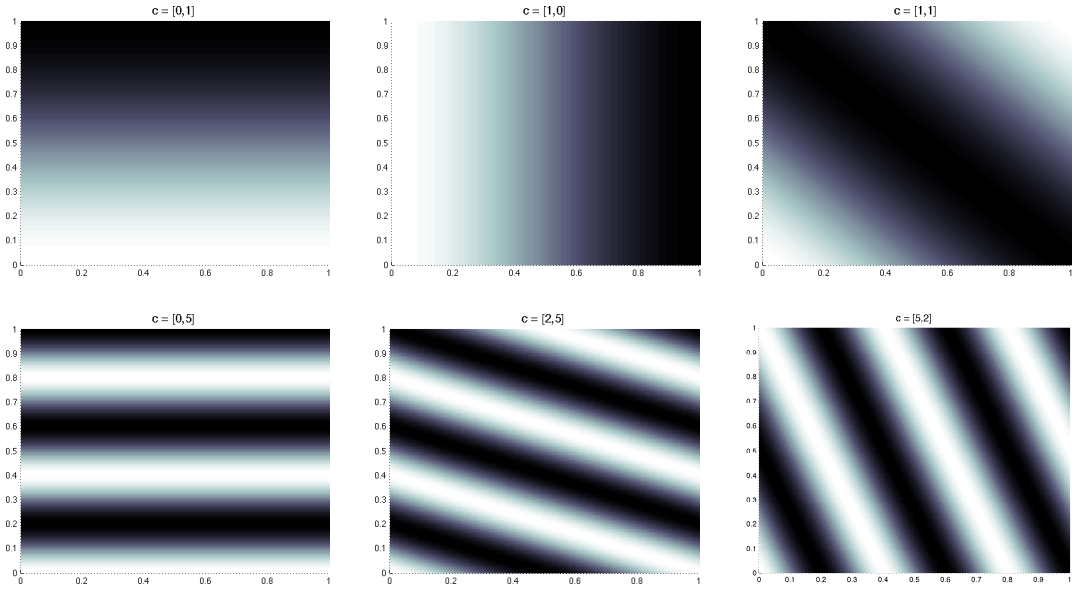


Figure 9.4: A selection of two-dimensional Fourier cosine basis functions ϕ_i, $i = 0, 1, 2, 3, 4, 5$. From Konidaris et al. (2011), permission pending.

that dimension. So if $c = (0, 0)$, the function is constant over both dimensions; if $c = (c_1, 0)$ the function is constant over the second dimension and varies over the first with frequency depending on c_1; and similarly, for $c = (0, c_2)$. When $c = (c_1, c_2)$ with neither $c_j = 0$, the basis function varies along both dimensions and represents an interaction between the two state variables. The values of c_1 and c_2 determine the frequency along each dimension, and their gives the direction of the interaction. Konidaris et al. (2011) found that when using Fourier cosine basis functions with a learning algorithm such as (9.7), semi-gradient TD(0), or semi-gradient Sarsa(λ), it is helpful to use a different step-size parameter for each basis function. If α is the basic step-size parameter, they suggest setting the step-size parameter for basis function ϕ_i to $\alpha_i = \alpha/\sqrt{(c_1^i)^2 + \cdots + (c_d^i)^2}$ (except when each $c_j^i = 0$, in which case $\alpha_i = \alpha$). Fourier cosine basis functions with Sarsa(λ) were found to produce good performance compared to several other collections of basis functions, including polynomial and radial basis functions, on several reinforcement learning tasks. Not surprisingly, however, Fourier basis functions have trouble with discontinuities because it is difficult to avoid "ringing" around points of discontinuity unless very high frequency basis functions are included.

As is true for polynomial approximation, the number of basis functions in the order-N Fourier cosine basis grows exponentially with the state space dimension. This makes it necessary to select a subset of these functions if the state space has high dimension (e.g., $d > 5$). This can be done using prior beliefs about he nature of the function to be approximated, and some automated selection methods can be adapted to deal with the incremental and nonstationary nature of reinforcement learning. Advantages of Fourier basis functions in this regard are that it is easy

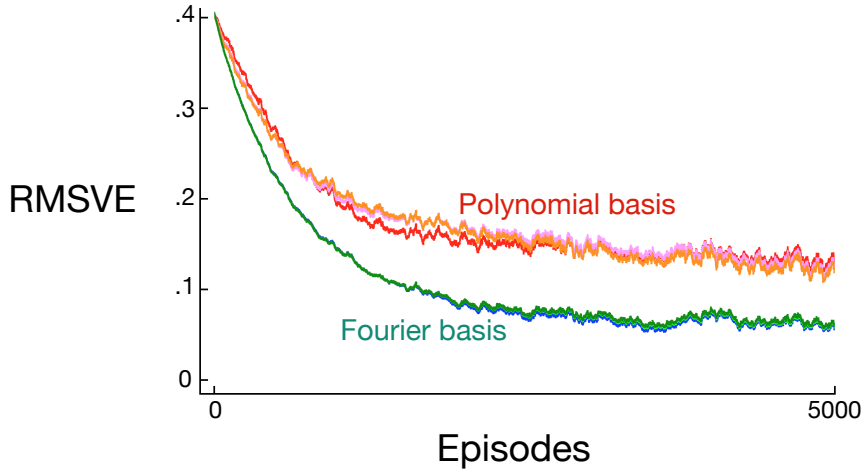


Figure 9.5: Fourier basis vs polynomials on the 1000-state random walk. Shown are learning curves for the gradient MC method with Fourier and polynomial bases of degree 5, 10, and 20. The step-size parameters were roughly optimized for each case: $\alpha = 0.0001$ for the polynomial basis and $\alpha = 0.00005$ for the Fourier basis.

to select functions by setting the c^i vectors to account for suspected interactions among the state variables, and by limiting the values in the c^j vectors so that the approximation can filter out high frequency components considered to be noise.

Figure 9.5 shows learning curves comparing the Fourier and polynomial bases on the 1000-state random walk example. In general, we do not recommend using the polynomial basis for online learning.

Exercise 9.5 Why does (9.16) define $(N+1)^d$ distinct functions for dimension d?

9.5.3 Coarse Coding

Consider a task in which the state set is continuous and two-dimensional. A state in this case is a point in 2-space, a vector with two real components. One kind of feature for this case is those corresponding to *circles* in state space, as shown in Figure 9.6. If the state is inside a circle, then the corresponding feature has the value 1 and is said to be *present*; otherwise the feature is 0 and is said to be *absent*. This kind of 1–0-valued feature is called a *binary feature*. Given a state, which binary features are present indicate within which circles the state lies, and thus coarsely code for its location. Representing a state with features that overlap in this way (although they need not be circles or binary) is known as *coarse coding*.

Assuming linear gradient-descent function approximation, consider the effect of the size and density of the circles. Corresponding to each circle is a single parameter (a component of $\boldsymbol{\theta}$) that is affected by learning. If we train at one state, a point in the space, then the parameters of all circles intersecting that state will be affected. Thus, by (9.8), the approximate value function will be affected at all states within the union of the circles, with a greater effect the more circles a point has "in common" with

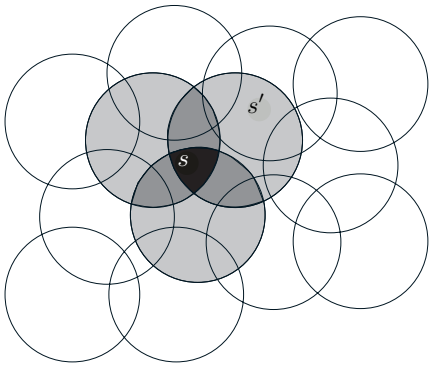


Figure 9.6: Coarse coding. Generalization from state s to state s' depends on the number of their features whose receptive fields (in this case, circles) overlap. These states have one feature in common, so there will be slight generalization between them.

the state, as shown in Figure 9.6. If the circles are small, then the generalization will be over a short distance, as in Figure 9.7a, whereas if they are large, it will be over a large distance, as in Figure 9.7b. Moreover, the shape of the features will determine the nature of the generalization. For example, if they are not strictly circular, but are elongated in one direction, then generalization will be similarly affected, as in Figure 9.7c.

Features with large receptive fields give broad generalization, but might also seem to limit the learned function to a coarse approximation, unable to make discriminations much finer than the width of the receptive fields. Happily, this is not the case. Initial generalization from one point to another is indeed controlled by the size and shape of the receptive fields, but acuity, the finest discrimination ultimately possible, is controlled more by the total number of features.

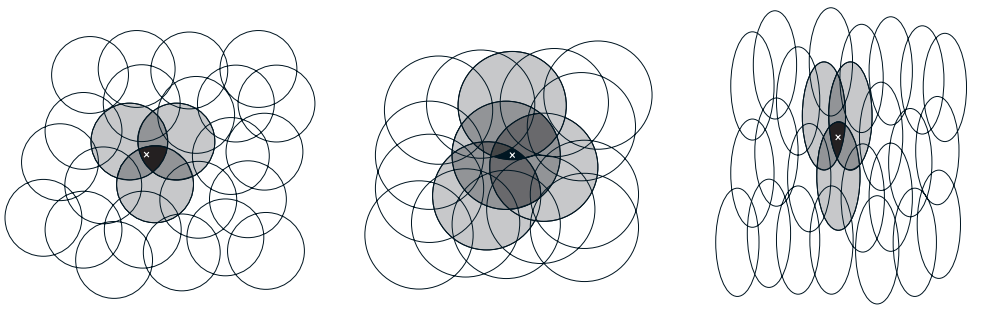


Figure 9.7: Generalization in linear function approximation methods is determined by the sizes and shapes of the features' receptive fields. All three of these cases have roughly the same number and density of features.

Example 9.3: Coarseness of Coarse Coding This example illustrates the effect on learning of the size of the receptive fields in coarse coding. Linear function approximation based on coarse coding and (9.7) was used to learn a one-dimensional square-wave function (shown at the top of Figure 9.8). The values of this function were used as the targets, U_t. With just one dimension, the receptive fields were intervals rather than circles. Learning was repeated with three different sizes of the intervals: narrow, medium, and broad, as shown at the bottom of the figure. All three cases had the same density of features, about 50 over the extent of the function being learned. Training examples were generated uniformly at random over this extent. The step-size parameter was $\alpha = \frac{0.2}{m}$, where m is the number of features that were present at one time. Figure 9.8 shows the functions learned in all three cases over the course of learning. Note that the width of the features had a strong effect early in learning. With broad features, the generalization tended to be broad; with narrow features, only the close neighbors of each trained point were changed, causing the function learned to be more bumpy. However, the final function learned was affected only slightly by the width of the features. Receptive field shape tends to have a strong effect on generalization but little effect on asymptotic solution quality.

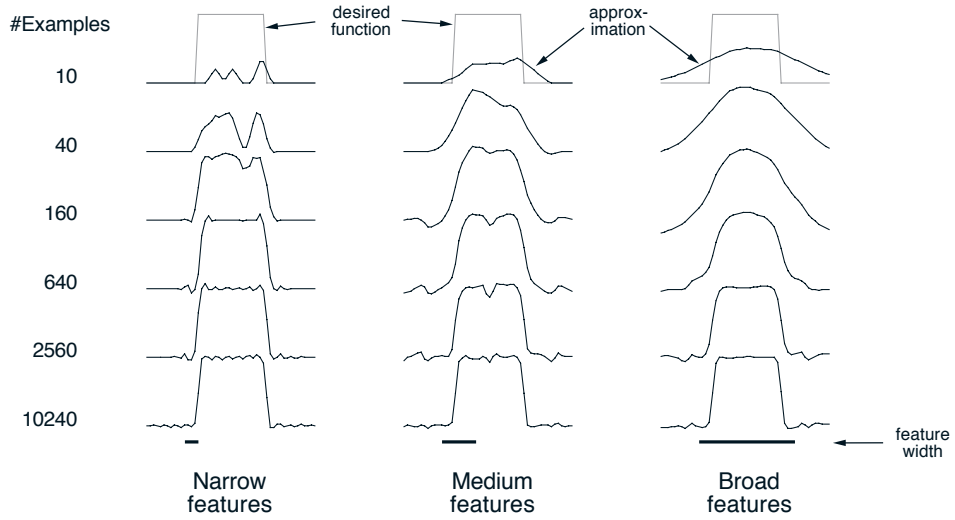


Figure 9.8: Example of feature width's strong effect on initial generalization (first row) and weak effect on asymptotic accuracy (last row).

■

9.5.4 Tile Coding

Tile coding is a form of coarse coding for multi-dimensional continuous spaces that is flexible and computationally efficient. It may be the most practical feature representation for modern sequential digital computers. Open-source software is available for many kinds of tile coding.

In tile coding the receptive fields of the features are grouped into partitions of the input space. Each such partition is called a *tiling*, and each element of the partition is called a *tile*. For example, the simplest tiling of a two-dimensional state space is a uniform grid such as that shown on the left side of Figure 9.9. The tiles or receptive field here are squares rather than the circles in Figure 9.6. If just this single tiling were used, then the state indicated by the white spot would be represented by the single feature whose tile it falls within; generalization would be complete to all states within the same tile and nonexistent to states outside it. With just one tiling, we would not have coarse coding by just a case of state aggregation.

To get the strengths of coarse coding requires overlapping receptive fields, and by definition the tiles of a partition do not overlap. To get true coarse coding with tile coding, multiple tilings are used, each offset by a fraction of a tile width. A simple case with four tilings is shown on the right side of Figure 9.9. Every state, such as that indicated by the white spot, falls in exactly one tile in each of the four tilings. These four tiles correspond to four features that become active when the state occurs. Specifically, the feature vector $\boldsymbol{\phi}(s)$ has one component for each tile in each tiling. In this example there are $4 \times 4 \times 4 = 64$ components, all of which will be 0 except for the four corresponding to the tiles that s falls within. Figure 9.10 shows the advantage of multiple offset tilings (coarse coding) over a single tiling on the 1000-state random walk example.

An immediate practical advantage of tile coding is that, because it works with partitions, the overall number of features that are active at one time is the same for any state. Exactly one feature is present in each tiling, so the total number of features present is always the same as the number of tilings. This allows the step-

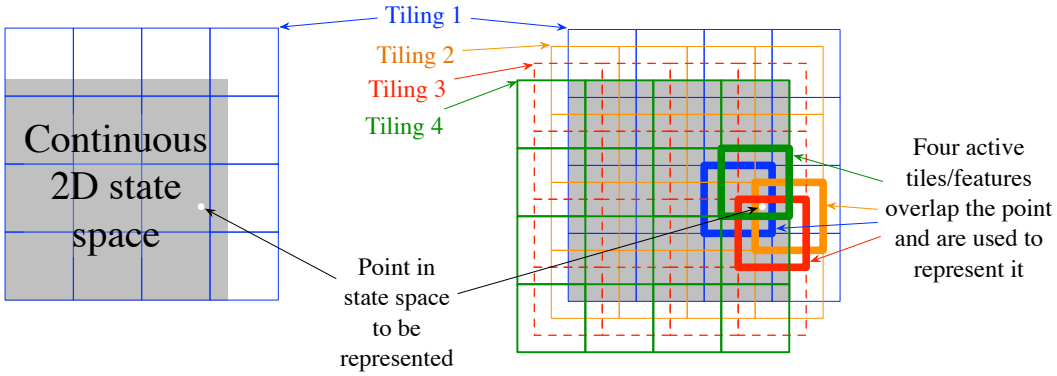


Figure 9.9: Multiple, overlapping grid-tilings on a limited two-dimensional space. These tilings are offset from one another by a uniform amount in each dimension.

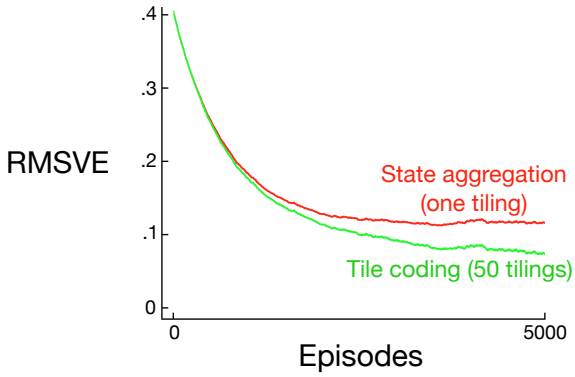


Figure 9.10: Why we use coarse coding. Shown are learning curves on the 1000-state random walk example for the gradient MC algorithm with a single and with multiple tilings. The space of 1000 states was treated as a single continuous dimension, covered with tiles each 200 states wide. The multiple tilings were offset from each other by 4 states. The step-size parameter was set so that the initial learning rate in the two cases was the same, $\alpha = 0.0001$ for the single tiling and $\alpha = 0.0001/50$ for the 50 tilings.

size parameter, α, to be set in an easy, intuitive way. For example, choosing $\alpha = \frac{1}{m}$, where m is the number of tilings, results in exact one-trial learning. If the example $s \mapsto v$ is trained on, then whatever the prior estimate, $\hat{v}(s,\boldsymbol{\theta}_t)$, the new estimate will be $\hat{v}(s,\boldsymbol{\theta}_{t+1}) = v$. Usually one wishes to change more slowly than this, to allow for generalization and stochastic variation in target outputs. For example, one might choose $\alpha = \frac{1}{10m}$, in which case the estimate for the trained state would move one-tenth of the way to the target in one update, and neighboring states will be moved less, proportional to the number of tiles they have in common.

Tile coding also gains computational advantages from its use of binary feature vectors. Because each component is either 0 or 1, the weighted sum making up the approximate value function (9.8) is almost trivial to compute. Rather than performing n multiplications and additions, one simply computes the indices of the $m \ll n$ active features and then adds up the m corresponding components of the parameter vector.

Generalization occurs to states other than the one trained if the those states fall within any of the same tiles, proportional to the number of tiles in common. Even the choice of how to offset the tilings from each other affects generalization. If they are offset uniformly in each dimension, as they were in Figure 9.9, then different states can generalize in qualitatively different ways, as shown below in the upper half of Figure 9.11. Each of the eight subfigures show the pattern of generalization from a trained state to nearby points. In this example their are eight tilings, thus 64 subregions within a tile that generalize distinctly, but all according to one of these eight patterns. Note how uniform offsets result in a strong effect along the diagonal in many patterns. These artifacts can be avoided if the tilings are offset asymmetrically, as shown in the lower half of the figure. These lower generalization patterns are better because they are all well centered on the trained state with no

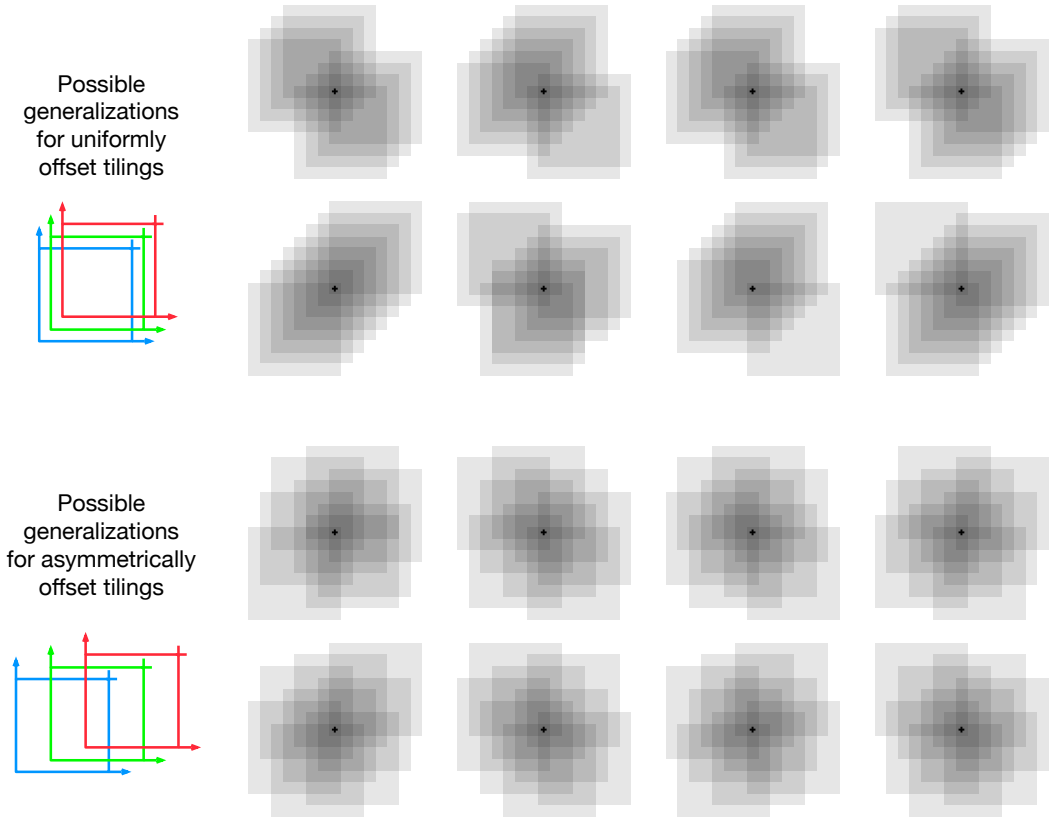


Figure 9.11: Why tile coding uses asymmetrical offsets. Shown is the strength of generalization from a trained state, indicated by the plus, to nearby states, for the case of eight tilings. If the tilings are uniformly offset (above), then there are diagonal artifacts and substantial variations in the generalization kernel, whereas with asymmetrically offset tilings the generalization kernel is more spherical and homogeneous.

obvious asymmetries.

Tilings in all cases are offset from each other by a fraction of a tile width in each dimension. If w denotes the tile width and k the number of tilings, then $\frac{w}{k}$ is a fundamental unit. Within small squares $\frac{w}{k}$ on a side, all states activate the same tiles, have the same feature representation, and the same approximated value. If a state is moved by $\frac{w}{k}$ in any cartesian direction, the feature representation changes by one component/tile. Uniformly offset tilings are offset from each other by exactly this unit distance. For a two-dimensional space, we say that each tiling is offset by the displacement vector $(1, 1)$, meaning that it is offset from the previous tiling by $\frac{w}{k}$ times this vector. In these terms, the asymmetrically offset tilings shown in the lower part of Figure 9.11 are offset by a displacement vector of $(1, 3)$.

Extensive studies have been made of the effect of different displacement vectors on the generalization of tile coding (Parks and Militzer, 1991; An, 1991; An, Miller and Parks, 1991; Miller, Glanz and Carter, 1991), assessing their homegeneity and tendency toward diagonal artifacts like those seen for the $(1, 1)$ displacement vectors.

Based on this work, Miller and Glanz (1996) recommend using displacement vectors consisting of the first odd integers. In particular, for a continuous space of dimension d, a good choice is to use the first odd integers $(1, 3, 5, 7, \ldots, 2d-1)$, with k (the number of tilings) set to an integer power of 2 greater than or equal to $4d$. This is what we have done to produce the tilings in the lower half of Figure 9.11, in which $d = 2$, $k = 2^3 \geq 4d$, and the displacement vector is $(1, 3)$. In a three-dimensional case, the first four tilings would be offset in total from a base position by $(0, 0, 0)$, $(1, 3, 5)$, $(2, 6, 10)$, and $(3, 9, 15)$. Open-source software that can efficiently make this sort of tilings for any d is readily available.

In choosing a tiling strategy, one has to pick the number of the tilings and the shape of the tiles. The number of tilings, along with the size of the tiles, determines the resolution or fineness of the asymptotic approximation, as in general coarse coding and illustrated in Figure 9.8. The shape of the tiles will determine the nature of generalization as in Figure 9.7. Square tiles will generalize roughly equally in each dimension as indicated in Figure 9.11 (lower). Tiles that are elongated along one dimension, such as the stripe tilings in Figure 9.12 b, will promote generalization along that dimension. The tilings in Figure 9.12 b are also denser and thinner on the left, promoting discrimination along the horizonal dimension at lower values along that dimension. The diagonal stripe tiling in Figure 9.12c will promote generalization along one diagonal. In higher dimensions, axis-aligned stripes correspond to ignoring some of the dimensions in some of the tilings, that is, to hyperplanar slices. Irregular tilings such as shown in Figure 9.12 a are also possible, though rare in practice and beyond the standard software.

In practice, it is often desirable to use different shaped tiles in different tilings. For example, one might use some vertical stripe tilings and some horizontal stripe tilings. This would encourage generalization along either dimension. However, with stripe tilings alone it is not possible to learn that a particular conjunction of horizontal and vertical coordinates has a distinctive value (whatever is learned for it will bleed into states with the same horizontal and vertical coordinates). For this one needs the conjunctive rectangular tiles such as originally shown in Figure 9.9. With multiple tilings—some horizontal, same vertical, and some conjunctive—one can get everything: a preference for generalizing along each dimension, yet the ability to learn

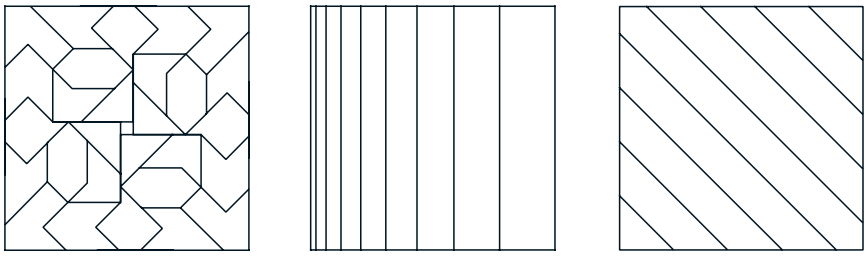
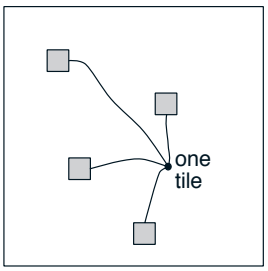


Figure 9.12: Tilings need not be grids. They can be arbitrarily shaped and non-uniform, while still in many cases being computationally efficient to compute.

specific values for conjunctions (see Section 16.3 for a case study using this). The choice of tilings determines generalization, and until this choice can be effectively automated, it is important that tile coding enables it to me done flexibly and in a way that people can understand.



Another important trick for reducing memory requirements is *hashing*—a consistent pseudo-random collapsing of a large tiling into a much smaller set of tiles. Hashing produces tiles consisting of noncontiguous, disjoint regions randomly spread throughout the state space, but that still form an exhaustive tiling. For example, one tile might consist of the four subtiles shown to the right. Through hashing, memory requirements are often reduced by large factors with little loss of performance. This is possible because high resolution is needed in only a small fraction of the state space. Hashing frees us from the curse of dimensionality in the sense that memory requirements need not be exponential in the number of dimensions, but need merely match the real demands of the task. Good open-source implementations of tile coding, including hashing, are widely available.

Exercise 9.6 Suppose we believe that one of two state dimensions is more likely to have an effect on the value function than is the other, that generalization should be primarily across this dimension rather than along it. What kind of tilings could be used to take advantage of this prior knowledge?

9.5.5 Radial Basis Functions

Radial basis functions (RBFs) are the natural generalization of coarse coding to continuous-valued features. Rather than each feature being either 0 or 1, it can be anything in the interval $[0, 1]$, reflecting various *degrees* to which the feature is present. A typical RBF feature, i, has a Gaussian (bell-shaped) response $\phi_i(s)$ dependent only on the distance between the state, s, and the feature's prototypical or center state, c_i, and relative to the feature's width, σ_i:

$$\phi_i(s) \doteq \exp\left(-\frac{||s - c_i||^2}{2\sigma_i^2}\right).$$

The norm or distance metric of course can be chosen in whatever way seems most appropriate to the states and task at hand. Figure 9.13 shows a one-dimensional

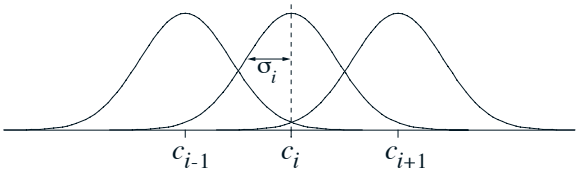


Figure 9.13: One-dimensional radial basis functions.

example with a Euclidean distance metric.

The primary advantage of RBFs over binary features is that they produce approximate functions that vary smoothly and are differentiable. Although this is appealing, in most cases it has no practical significance. Nevertheless, extensive studies of graded response functions such as RBFs in the context of tile coding (An, 1991; Miller et al., 1991; An, Miller and Parks, 1991; Lane, Handelman and Gelfand, 1992). All of these methods require substantial addition computational complexity (over tile coding) and often reduce performance when there are more more than two state dimensions. In high dimensions the edges of tiles are much more important, and has proven difficult to obtain well controlled graded tile activations near the edges.

An *RBF network* is a linear function approximator using RBFs for its features. Learning is defined by equations (9.7) and (9.8), exactly as in other linear function approximators. In addition, some learning methods for RBF networks change the centers and widths of the features as well, bringing them into the realm of nonlinear function approximators. Nonlinear methods may be able to fit target functions much more precisely. The downside to RBF networks, and to nonlinear RBF networks especially, is greater computational complexity and, often, more manual tuning before learning is robust and efficient.

9.6 Nonlinear Function Approximation: Artificial Neural Networks

Artificial neural networks (ANNs) are widely used for nonlinear function approximation. An ANN is a network of interconnected units that have some of the properties of neurons. ANNs have a long history, with latest advances in training deeply-layered ANNs being responsible for some of the most impressive abilities of machine learning systems, including reinforcement learning systems. In Chapter 16 we describe several stunning examples of reinforcement learning systems that use ANN function approximation.

Figure 9.14 shows a generic feedforward ANN, meaning that there are no loops in the network. This network has an output layer consisting of two output units, an input layer with four input units, and two hidden layers: layers that are neither input nor output layers. A real-valued parameter—a weight—is associated with each link. A weight roughly corresponds to the efficacy of a synaptic connection in a real neural network (see Section 15.1).

The units (the circles in Figure 9.14) are typically semi-linear units, meaning that they compute a weighted sum of their input signals and then apply to the result a nonlinear function, called the *activation function*, to produce the unit's output, or activation. Many different activation functions are used, but they are typically S-shaped, or sigmoid, functions such as the logistic function $f(x) = 1/1 + e^{-x}$, though sometimes the rectifier nonlinearity $f(x) = \max(0, x)$ is used. A step function like $f(x) = 1$ if $x \geq \theta$, and 0 otherwise, results in a binary unit with threshold θ. It

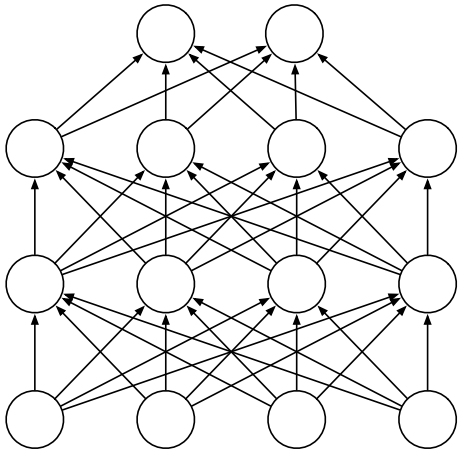


Figure 9.14: A generic feedforward neural network with four input units, two output units, and two hidden layers.

is often useful for units in different layers to use different activation functions. An ANN is recurrent if there is at least one loop in its connections. Although both feedforward and recurrent ANNs have been used in reinforcement learning, here we look only at the simpler feedforward case.

The activation of each output unit of a feedforward ANN is a nonlinear function of the activation patterns over the network's input units. The functions are parameterized by the network's connection weights. An ANN with no hidden layers can represent only a very small fraction of the possible input-output functions. However an ANN with a single hidden layer having a large enough finite number of sigmoid units can approximate any continuous function on a compact region of the network's input space to any degree of accuracy (Cybenko, 1989). This is also true for other nonlinear activation functions that satisfy mild conditions, but nonlinearity is essential: if all the units in a multi-layer feedforward ANN have linear activation functions, the entire network is equivalent to a network with no hidden layers (because linear functions of linear functions are themselves linear).

Despite this "universal approximation" property of one-hidden-layer ANNs, both experience and theory show that approximating the complex functions needed for many artificial intelligence tasks is made easier—indeed may require—abstractions that are hierarchical compositions of many layers of lower-level abstractions, that is, abstractions produced by deep architectures such as ANNs with many hidden layers. (See Bengio, 2009, for a thorough review.) The successive layers of a deep ANN compute increasingly abstract representations of the network's "raw" input, with each unit providing a feature contributing to a hierarchical representation of the overall input-output function of the network.

Creating these kinds of hierarchical representations without relying exclusively on hand-crafted features has been an enduring challenge for artificial intelligence.

This is why learning algorithms for ANNs with hidden layers have received so much attention over the years. ANNs typically learn by a stochastic gradient method (Section 9.3). Each weight is adjusted in a direction aimed at improving the network's overall performance as measured by an objective function to be either minimized or maximized. In the most common supervised learning case, the objective function is the expected error, or loss, over a set of labeled training examples. In reinforcement learning, ANNs can use TD errors to learn value functions, or they can aim to maximize expected reward as in a gradient bandit (Section 2.7) or a policy-gradient algorithm (Chapter 13). In all of these cases it is necessary to estimate how a change in each connection weight would influence the network's overall performance, in other words, to estimate the partial derivative of an objective function with respect to each weight, given the current values of all the network's weights. The gradient is the vector of these partial derivatives.

The most successful way to do this for ANNs with hidden layers (provided the units have differentiable activation functions) is the backpropagation algorithm, which consists of alternating forward and backward passes through the network (Rumelhart, Hinton, and Williams, 1986). Each forward pass computes the activation of each unit given the current activations of the network's input units. After each forward pass, a backward pass efficiently computes a partial derivative for each weight. (As in other stochastic gradient learning algorithms, the vector of these partial derivatives is an estimate of the true gradient.)

The backpropagation algorithm can produce good results for shallow networks having 1 or 2 hidden layers, but it does not work well for deeper ANNs. In fact, training a network with $k+1$ hidden layers can actually result in poorer performance than training a network with k hidden layers, even though the deeper network can represent all the functions that the shallower network can (Bengio, 2009). Explaining all results like these is not easy, but several factors are important. First, the large number of weights in a typical deep ANN makes it difficult to avoid the problem of overfitting, that is, the problem of failing to generalize correctly to cases on which the network has not been trained. Second, backpropagation does not work well for deep ANNs because the partial derivatives computed by its backward passes either decay rapidly toward the input end of the network, making learning by deep layers extremely slow, or they grow rapidly, making learning unstable. Methods for dealing with these problems are largely responsible for many impressive results achieved by systems that use deep ANNs.

Overfitting is a problem for any function approximation method that adjusts functions with many degrees of freedom on the basis of limited training data. It is less of a problem for on-line reinforcement learning that does not rely on limited training sets, but generalizing effectively is still an important issue. Overfitting is a problem for ANNs in general, but especially so for deep ANNs because they tend to have very large numbers of weights. Many methods have been developed for reducing overfitting. These include stopping training when performance begins to decrease on validation data different from the training data (cross validation), modifying the objective function to discourage complexity of the approximation (regularization),

and introducing dependencies among the weights to reduce the number of degrees of freedom (e.g., weight sharing).

A particularly effective method for reducing overfitting by deep ANNs is the dropout method introduced by Srivastava, Hinton, Krizhevsky, Sutskever, and Salakhutdinov (2014). During training, units are randomly removed from the network (dropped out) along with their connections. This can be thought of as training a large number of "thinned" networks. Combining the results of these thinned networks at test time is a way to improve generalization performance. The dropout method efficiently approximates this combination by multiplying each outgoing weight of a unit by the probability that that unit was retained during training. Srivastava et al. found that this method significantly improves generalization performance. It encourages individual hidden units to learn features that work well with random collections of other features. This increases the versatility of the features formed by the hidden units so that the network does overly specialize to rarely-occurring cases.

Hinton, Osindero, and Teh (2006) took a major step toward solving the problem of training the deep layers of a deep ANN in their work with deep belief networks, layered networks closely related to the deep ANNs discussed here. In their method, the deepest layers are trained one at a time using an unsupervised learning algorithm. Without relying on the overall objective function, unsupervised learning can extract features that capture statistical regularities of the input stream. The deepest layer is trained first, then with input provided by this trained layer, the next deepest layer is trained, and so on, until the weights in all, or many, of the network's initial layers are set to values that now act as initial values for supervised learning of the whole network to fine-tune it by backpropagation with respect to the overall objective function. Studies show that this approach generally works much better than backpropagation with weights initialized with random values. This could happen for many reasons, but one idea is that this way of initializing weights places the network in a region of parameter space from which a gradient-based algorithm can make good progress.

A type of deep ANN that has proven to be very successful in applications, including impressive reinforcement learning applications (Chapter 16) is the *deep convolutional network*. This type of network is specialized for processing high-dimensional data arranged in spatial arrays, such as images. It was inspired by how early visual processing works in the brain (LeCun, Bottou, Bengio and Haffner, 1998). Because of its special architecture, a deep convolutional network can be trained by backpropagation without resorting to methods like those described above to train the deep layers.

Figure 9.15 illustrates the architecture of a deep convolutional network. This instance, from LeCun et al. (1998), was designed to recognize hand-written characters. It consists of alternating convolutional and subsampling layers, followed by several fully connected final layers. Each convolutional layer produces a number of feature maps. A feature map is a pattern of activity over an array of units, where each unit performs the same operation on data in its receptive field, which is the part of the data it "sees" from the preceding layer (or from the external input in the case of the first convolutional layer). The units of a feature map are identical to one another

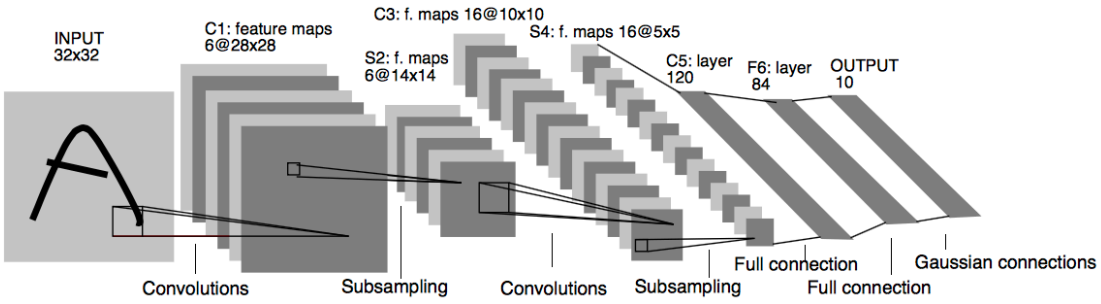


Figure 9.15: Deep Convolutional Network. Reprinted from LeCun, Bottou, Bengio, and Haffner (1998). Permission pending.

except that their receptive fields, which are all the same size and shape, are shifted to different locations on the arrays incoming data. Units in the same feature map share the same weights. This means that a feature map detects the same feature no matter where it is located in the input array. In the network in Figure 9.15, for example, the first convolutional layer produces 6 feature maps, each consisting of 28 $\times$ 28 units. Each unit in each feature map has a 5 $\times$ 5 receptive field, and these receptive fields overlap (in this case by four columns and five rows). Consequently, each of the 6 feature maps is specified by just 25 adjustable weights.

The subsampling layers of a deep convolutional network reduce the spatial resolution of the feature maps. Each feature map in a subsampling layer consists of units that average over a receptive field of units in the feature maps of the preceding convolutional layer. For example, each unit in each of the 6 feature maps in first subsampling layer of the network of Figure 9.15 average over a 2 $\times$ 2 non-overlapping receptive fields of a feature map produced by the first convolutional layer, resulting in six 14 $\times$ 14 feature maps. The subsampling layers reduce the network's sensitivity to the spacial locations of the features detected, that is, they help make the network's responses spatially invariant. This is useful because a feature detected at one place in an image is likely to be useful at other places as well.

Advances in the design and training of ANNs—of which we have only mentioned a few—all contribute to reinforcement learning. Although current reinforcement learning theory is mostly limited to methods using tabular or linear function approximation methods, the impressive performances of notable reinforcement learning applications of owe much of their success to nonlinear function approximation by ANNs, in particular, by deep ANNs.

9.7 Least-Squares TD

In Section 9.4 we established that TD(0) with linear function approximation converges asymptotically, for appropriately decreasing step sizes, to the TD fixpoint:

$$\boldsymbol{\theta}_{TD} = \mathbf{A}^{-1}\mathbf{b},$$

where

$$\mathbf{A} \doteq \mathbb{E}\Big[\boldsymbol{\phi}_t(\boldsymbol{\phi}_t - \gamma\boldsymbol{\phi}_{t+1})^\top\Big] \quad \text{and} \quad \mathbf{b} \doteq \mathbb{E}[R_{t+1}\boldsymbol{\phi}_t]\,.$$

Why, we might ask, must we compute this solution iteratively? This is wasteful of data! Could one not do better by computing estimates of $\mathbf{A}$ and $\mathbf{b}$, and then directly computing the TD fixpoint? The Least-Squares TD algorithm, commonly known as LSTD, does exactly this. It forms the natural estimates

$$\widehat{\mathbf{A}}_t \doteq \sum_{k=0}^{t} \boldsymbol{\phi}_k(\boldsymbol{\phi}_k - \gamma\boldsymbol{\phi}_{k+1})^\top + \varepsilon\mathbf{I} \quad \text{and} \quad \widehat{\mathbf{b}}_t \doteq \sum_{k=0}^{t} R_{t+1}\boldsymbol{\phi}_k \tag{9.17}$$

(where $\varepsilon\mathbf{I}$, for some small $\varepsilon > 0$, ensures that $\widehat{\mathbf{A}}_t$ is always invertible) and then estimates the TD fixpoint as

$$\boldsymbol{\theta}_{t+1} \doteq \widehat{\mathbf{A}}_t^{-1}\widehat{\mathbf{b}}_t. \tag{9.18}$$

This algorithm is the most data efficient form of linear TD(0), but it is also much more expensive computationally. Recall that semi-gradient TD(0) requires memory and per-step computation that is only $O(n)$.

How complex is LSTD? As it is written above the complexity seems to increase with t, but the two approximations in (9.17) could be implemented incrementally using the techniques we have covered earlier (e.g., in Chapter 2) so that they can be done in constant time per step. Even so, the update for $\widehat{\mathbf{A}}_t$ would involve an outer product (a column vector times a row vector) and thus would be a matrix update; its computational complexity would be $O(n^2)$, and of course the memory required to hold the $\widehat{\mathbf{A}}_t$ matrix would be $O(n^2)$.

A potentially greater problem is that our final computation (9.18) uses the inverse of $\widehat{\mathbf{A}}_t$, and the computational complexity of a general inverse computation is $O(n^3)$. Fortunately, an inverse of a matrix of our special form—a sum of outer products—can also be updated incrementally with only $O(n^2)$ computations, as

$$\begin{aligned}
\widehat{\mathbf{A}}_t^{-1} &= \Big(\widehat{\mathbf{A}}_{t-1} + \boldsymbol{\phi}_t(\boldsymbol{\phi}_t - \gamma\boldsymbol{\phi}_{t+1})^\top\Big)^{-1} && \text{(from (9.17))} \\
&= \widehat{\mathbf{A}}_{t-1}^{-1} - \frac{\widehat{\mathbf{A}}_{t-1}^{-1}\boldsymbol{\phi}_t(\boldsymbol{\phi}_t - \gamma\boldsymbol{\phi}_{t+1})^\top\widehat{\mathbf{A}}_{t-1}^{-1}}{1 + (\boldsymbol{\phi}_t - \gamma\boldsymbol{\phi}_{t+1})^\top\widehat{\mathbf{A}}_{t-1}^{-1}\boldsymbol{\phi}_t}, && (9.19)
\end{aligned}$$

with $\widehat{\mathbf{A}}_{-1} \doteq \varepsilon\mathbf{I}$. Although the identity (9.19), known as *the Sherman-Morrison formula*, is superficially complicated, it involves only vector-matrix and vector-vector multiplications and thus is only $O(n^2)$. Thus we can store and maintain the inverse matrix $\widehat{\mathbf{A}^{-1}}_t$, and then use it in (9.18), all with only $O(n^2)$ memory and per-step computation. The complete algorithm is given in the box.

Of course, $O(n^2)$ is still significantly more expensive than the $O(n)$ of semi-gradient TD. Whether the greater data efficiency of LSTD is worth this computational expense depends on how large n is, how important it is to learn quickly, and the expense of

LSTD for estimating $\hat{v} \approx v_\pi$ ($O(n^2)$ version)

Input: feature representation $\boldsymbol{\phi}(s) \in \mathbb{R}^n, \forall s \in \mathcal{S}, \boldsymbol{\phi}(\text{terminal}) \doteq \mathbf{0}$

$\widehat{\mathbf{A}^{-1}} \leftarrow \varepsilon^{-1}\mathbf{I}$ — An $n \times n$ matrix
$\widehat{\mathbf{b}} \leftarrow \mathbf{0}$ — An n-dimensional vector
Repeat (for each episode):
  Initialize S; obtain corresponding $\boldsymbol{\phi}$
  Repeat (for each step of episode):
    Choose $A \sim \pi(\cdot|S)$
    Take action A, observe R, S'; obtain corresponding $\boldsymbol{\phi}'$
    $\mathbf{v} \leftarrow \widehat{\mathbf{A}^{-1}}^\top(\boldsymbol{\phi} - \gamma\boldsymbol{\phi}')$
    $\widehat{\mathbf{A}^{-1}} \leftarrow \widehat{\mathbf{A}^{-1}} - \big(\widehat{\mathbf{A}^{-1}}\boldsymbol{\phi}\big)\mathbf{v}^\top / \big(1 + \mathbf{v}^\top\boldsymbol{\phi}\big)$
    $\widehat{\mathbf{b}} \leftarrow \widehat{\mathbf{b}} + R\boldsymbol{\phi}$
    $\boldsymbol{\theta} \leftarrow \widehat{\mathbf{A}^{-1}}\widehat{\mathbf{b}}$
    $S \leftarrow S'$; $\boldsymbol{\phi} \leftarrow \boldsymbol{\phi}'$
  until S' is terminal

other parts of the system. The fact that LSTD requires no step-size hyperparameter is sometimes also touted, but the advantage of this is probably overstated. LSTD does not require a step size, but it does requires ε; if ε is chosen too small the sequence of inverses can vary wildly, and if ε is chosen too large then learning is slowed. In addition, LSTD's lack of a step size parameter means that it never forgets. This is sometimes desirable, but it is problematic if the target policy π changes as it does in reinforcement learning and GPI. In control applications, LSTD typically has to be combined with some other mechanism to induce forgeting, mooting any initial advantage of not requiring a step size parameter.

9.8 Summary

Reinforcement learning systems must be capable of *generalization* if they are to be applicable to artificial intelligence or to large engineering applications. To achieve this, any of a broad range of existing methods for *supervised-learning function approximation* can be used simply by treating each backup as a training example. Perhaps the most suitable of these methods are those using *parameterized* function approximation and variations of *stochastic gradient descent* (SGD). In this chapter we have focused on the *on-policy* case with a *fixed policy*, also known as policy evaluation or prediction; a natural learning algorithm for this case is *n-step semi-gradient TD*, with includes gradient MC and semi-gradient TD(0) algorithms as the special cases when $n = \infty$ and $n = 1$ respectively.

We have also focused on *linear* function approximation, in which the value estimates are sums of features weighted by corresponding parameters. The linear case

is the most well understood theoretically and works well in practice when provided with appropriate features. Choosing the features is one of the most important ways of adding prior domain knowledge to reinforcement learning systems. They can be chosen as polynomials, but this case generalizes poorly in the online learning setting typically considered in reinforcement learning. Better is to choose features according the Fourier basis, or according to some form of coarse coding with sparse overlapping receptive fields. Tile coding is a form of coarse coding that is particularly computationally efficient and flexible. Radial basis functions are useful for one- or two-dimensional tasks in which a smoothly varying response is important. LSTD is the most data-efficient linear TD prediction method, but requires computation proportional to the square of the number of parameters, whereas all the other methods are of complexity linear in the number of parameters. Nonlinear methods include artificial neural networks trained by backpropagation and variations of SGD; these methods have become very popular in recent years under the name *deep reinforcement learning*.

Semi-gradient TD methods are not true gradient methods. In such bootstrapping methods (including DP), the parameter vector appears in the update target, yet this is not taken into account in computing the gradient—thus they are *semi*-gradient methods. As such, they cannot rely on classical SGD results. Nevertheless, linear semi-gradient n-step TD is guaranteed to converge under standard conditions, for all n, to a MSVE that is within a bound of the optimal error. Although the bound is always tighter for higher n, approaching zero as $n \to \infty$, in practice this choice results in very slow learning and some degree of bootstrapping (n=1) is preferrable.

Bibliographical and Historical Remarks

Generalization and function approximation have always been an integral part of reinforcement learning. Bertsekas and Tsitsiklis (1996), Bertsekas (2012), and Sugiyama et al. (2013) present the state of the art in function approximation in reinforcement learning. Some of the early work with function approximation in reinforcement learning is discussed at the end of this section.

9.3 Gradient-descent methods for the minimizing mean-squared error in supervised learning are well known. Widrow and Hoff (1960) introduced the least-mean-square (LMS) algorithm, which is the prototypical incremental gradient-descent algorithm. Details of this and related algorithms are provided in many texts (e.g., Widrow and Stearns, 1985; Bishop, 1995; Duda and Hart, 1973).

Semi-gradient TD(0) was first explored by Sutton (1984, 1988), as part of the linear TD(λ) algorithm that we will treat in Chapter 12. The term "semi-gradient" to describe these bootstrapping methods is new to the second edition of this book.

The earliest use of state aggregation in reinforcement learning may have been Michie and Chambers's BOXES system (1968). The theory of state aggre-

gation in reinforcement learning has been developed by Singh, Jaakkola, and Jordan (1995) and Tsitsiklis and Van Roy (1996). State aggregation has been used in dynamic programming from its earliest days (e.g., Bellman, 1957a).

9.4 Sutton (1988) proved convergence of linear TD(0) in the mean to the minimal MSVE solution for the case in which the feature vectors, $\{\boldsymbol{\phi}(s) : s \in \mathcal{S}\}$, are linearly independent. Convergence with probability 1 (and for general λ) was proved by several researchers at about the same time (Peng, 1993; Dayan and Sejnowski, 1994; Tsitsiklis, 1994; Gurvits, Lin, and Hanson, 1994). In addition, Jaakkola, Jordan, and Singh (1994) proved convergence under on-line updating. All of these results assumed linearly independent feature vectors, which implies at least as many components to $\boldsymbol{\theta}_t$ as there are states. Convergence for the more important case of general (dependent) feature vectors was first shown by Dayan (1992). A significant generalization and strengthening of Dayan's result was proved by Tsitsiklis and Van Roy (1997). They proved the main result presented in this section, the bound on the asymptotic error of linear bootstrapping methods.

9.5 Our presentation of the range of possibilities for linear function approximation is based on that by Barto (1990).

9.5.3 The term *coarse coding* is due to Hinton (1984), and our Figure 9.6 is based on one of his figures. Waltz and Fu (1965) provide an early example of this type of function approximation in a reinforcement learning system.

9.5.4 Tile coding, including hashing, was introduced by Albus (1971, 1981). He described it in terms of his "cerebellar model articulator controller," or CMAC, as tile coding is known in the literature. The term "tile coding" is new to this book, though the idea of describing CMAC in these terms is taken from Watkins (1989). Tile coding has been used in many reinforcement learning systems (e.g., Shewchuk and Dean, 1990; Lin and Kim, 1991; Miller, Scalera, and Kim, 1994; Sofge and White, 1992; Tham, 1994; Sutton, 1996; Watkins, 1989) as well as in other types of learning control systems (e.g., Kraft and Campagna, 1990; Kraft, Miller, and Dietz, 1992). This section draws heavily on the work of Miller and Glanz (1996).

9.5.5 Function approximation using radial basis functions (RBFs) has received wide attention ever since being related to neural networks by Broomhead and Lowe (1988). Powell (1987) reviewed earlier uses of RBFs, and Poggio and Girosi (1989, 1990) extensively developed and applied this approach.

9.7 LSTD is due to Bradtke and Barto (see Bradtke, 1993, 1994; Bradtke and Barto, 1996; Bradtke, Ydstie, and Barto, 1994), and was further developed by Boyan (2002). The incremental update of the inverse matrix has been known at least since 1949 (Sherman and Morrison, 1949).

The use of function approximation in reinforcement learning goes back to the early neural networks of Farley and Clark (1954; Clark and Farley, 1955), who used reinforcement learning to adjust the parameters of linear threshold functions representing policies. The earliest example we know of in which function approximation methods were used for learning value functions was Samuel's checkers player (1959, 1967). Samuel followed Shannon's (1950) suggestion that a value function did not have to be exact to be a useful guide to selecting moves in a game and that it might be approximated by linear combination of features. In addition to linear function approximation, Samuel experimented with lookup tables and hierarchical lookup tables called signature tables (Griffith, 1966, 1974; Page, 1977; Biermann, Fairfield, and Beres, 1982).

At about the same time as Samuel's work, Bellman and Dreyfus (1959) proposed using function approximation methods with DP. (It is tempting to think that Bellman and Samuel had some influence on one another, but we know of no reference to the other in the work of either.) There is now a fairly extensive literature on function approximation methods and DP, such as multigrid methods and methods using splines and orthogonal polynomials (e.g., Bellman and Dreyfus, 1959; Bellman, Kalaba, and Kotkin, 1973; Daniel, 1976; Whitt, 1978; Reetz, 1977; Schweitzer and Seidmann, 1985; Chow and Tsitsiklis, 1991; Kushner and Dupuis, 1992; Rust, 1996).

Holland's (1986) classifier system used a selective feature-match technique to generalize evaluation information across state–action pairs. Each classifier matched a subset of states having specified values for a subset of features, with the remaining features having arbitrary values ("wild cards"). These subsets were then used in a conventional state-aggregation approach to function approximation. Holland's idea was to use a genetic algorithm to evolve a set of classifiers that collectively would implement a useful action-value function. Holland's ideas influenced the early research of the authors on reinforcement learning, but we focused on different approaches to function approximation. As function approximators, classifiers are limited in several ways. First, they are state-aggregation methods, with concomitant limitations in scaling and in representing smooth functions efficiently. In addition, the matching rules of classifiers can implement only aggregation boundaries that are parallel to the feature axes. Perhaps the most important limitation of conventional classifier systems is that the classifiers are learned via the genetic algorithm, an evolutionary method. As we discussed in Chapter 1, there is available during learning much more detailed information about how to learn than can be used by evolutionary methods. This perspective led us to instead adapt supervised learning methods for use in reinforcement learning, specifically gradient-descent and neural network methods. These differences between Holland's approach and ours are not surprising because Holland's ideas were developed during a period when neural networks were generally regarded as being too weak in computational power to be useful, whereas our work was at the beginning of the period that saw widespread questioning of that conventional wisdom. There remain many opportunities for combining aspects of these different approaches.

A number of reinforcement learning studies using function approximation meth-

ods that we have not covered previously should be mentioned. Barto, Sutton, and Brouwer (1981) and Barto and Sutton (1981b) extended the idea of an associative memory network (e.g., Kohonen, 1977; Anderson, Silverstein, Ritz, and Jones, 1977) to reinforcement learning. Hampson (1983, 1989) was an early proponent of multilayer neural networks for learning value functions. Anderson (1986, 1987) coupled a TD algorithm with the error backpropagation algorithm to learn a value function. Barto and Anandan (1985) introduced a stochastic version of Widrow, Gupta, and Maitra's (1973) *selective bootstrap algorithm*, which they called the *associative reward-penalty* (A_{R-P}) *algorithm*. Williams (1986, 1987, 1988, 1992) extended this type of algorithm to a general class of REINFORCE algorithms, showing that they perform stochastic gradient ascent on the expected reinforcement. Gullapalli (1990) and Williams devised algorithms for learning generalizing policies for the case of continuous actions. Phansalkar and Thathachar (1995) proved both local and global convergence theorems for modified versions of REINFORCE algorithms. Christensen and Korf (1986) experimented with regression methods for modifying coefficients of linear value function approximations in the game of chess. Chapman and Kaelbling (1991) and Tan (1991) adapted decision-tree methods for learning value functions. Explanation-based learning methods have also been adapted for learning value functions, yielding compact representations (Yee, Saxena, Utgoff, and Barto, 1990; Dietterich and Flann, 1995).

References

Adams, C. D. and Dickinson, A. (1981). Instrumental responding following reinforcer devaluation. *The Quarterly Journal of Experimental Psychology*, 33(2):109–121.

Agrawal, R. (1995). Sample mean based index policies with $O(logn)$ regret for the multiarmed bandit problem. *Advances in Applied Probability*, 27:1054–1078.

Agre, P. E. (1988). *The Dynamic Structure of Everyday Life*. Ph.D. thesis, Massachusetts Institute of Technology. AI-TR 1085, MIT Artificial Intelligence Laboratory.

Agre, P. E., Chapman, D. (1990). What are plans for? *Robotics and Autonomous Systems*, 6:17–34.

Albus, J. S. (1971). A theory of cerebellar function. *Mathematical Biosciences*, 10:25–61.

Albus, J. S. (1981). *Brain, Behavior, and Robotics*. Byte Books, Peterborough, NH.

An, P.-C. E. (1991). *An Improved Multi-dimensional CMAC Neural network: Receptive Field Function and Placement* (Doctoral dissertation, PhD Thesis, Dept. Electrical and Computer Engineering, New Hampshire Univ., New Hampshire, USA).

An, P. C. E., Miller, W. T., Parks, P. C. (1991). Design improvements in associative memories for cerebellar model articulation controllers (CMAC). *Artificial Neural Networks*, pp. 1207–1210, Elsvier North-Holland.

Anderson, C. W. (1986). *Learning and Problem Solving with Multilayer Connectionist Systems*. Ph.D. thesis, University of Massachusetts, Amherst.

Anderson, C. W. (1987). Strategy learning with multilayer connectionist representations. *Proceedings of the Fourth International Workshop on Machine Learning*, pp. 103–114. Morgan Kaufmann, San Mateo, CA.

Anderson, J. A., Silverstein, J. W., Ritz, S. A., Jones, R. S. (1977). Distinctive features, categorical perception, and probability learning: Some applications of a neural model. *Psychological Review*, 84:413–451.

Andreae, J. H. (1963). STELLA: A scheme for a learning machine. In *Proceedings of the 2nd IFAC Congress, Basle*, pp. 497–502. Butterworths, London.

Andreae, J. H. (1969a). A learning machine with monologue. *International Journal of Man–Machine Studies*, 1:1–20.

Andreae, J. H. (1969b). Learning machines—a unified view. In A. R. Meetham and R. A. Hudson (eds.), *Encyclopedia of Information, Linguistics, and Control*, pp. 261–270. Pergamon, Oxford.

Andreae, J. H. (1977). *Thinking with the Teachable Machine*. Academic Press, London.

Arthur, W. B. (1991). Designing economic agents that act like human agents: A behavioral approach to bounded rationality. *The American Economic Review 81*(2):353-359.

Auer, P., Cesa-Bianchi, N., Fischer, P. (2002). Finite-time analysis of the multiarmed bandit

problem. *Machine learning*, 47(2-3):235–256.

Baird, L. C. (1995). Residual algorithms: Reinforcement learning with function approximation. In *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 30–37. Morgan Kaufmann, San Francisco.

Baldassarre, G. and Mirolli, M., editors (2013). *Intrinsically Motivated Learning in Natural and Artificial Systems*. Springer-Verlag, Berlin.

Balke, A., Pearl, J. (1994). Counterfactual probabilities: Computational methods, bounds and applications. In *Proceedings of the Tenth International Conference on Uncertainty in Artificial Intelligence* (pp. 46-54). Morgan Kaufmann.

Bao, G., Cassandras, C. G., Djaferis, T. E., Gandhi, A. D., Looze, D. P. (1994). Elevator dispatchers for down peak traffic. Technical report. ECE Department, University of Massachusetts, Amherst.

Baras, D. and Meir, R. (2007). Reinforcement learning, spike-time-dependent plasticity, and the BCM rule. *Neural Computation*, 19(8):2245–2279.

Barnard, E. (1993). Temporal-difference methods and Markov models. *IEEE Transactions on Systems, Man, and Cybernetics*, 23:357–365.

Bartlett, P. L. and Baxter, J. (1999). Hebbian synaptic modifications in spiking neurons that learn. Technical report, Research School of Information Sciences and Engineering, Australian National University.

Bartlett, P. L. and Baxter, J. (2000). A biologically plausible and locally optimal learning algorithm for spiking neurons. Rapport technique, Australian National University.

Barto, A. G. (1985). Learning by statistical cooperation of self-interested neuron-like computing elements. *Human Neurobiology*, 4:229–256.

Barto, A. G. (1986). Game-theoretic cooperativity in networks of self-interested units. In J. S. Denker (ed.), *Neural Networks for Computing*, pp. 41–46. American Institute of Physics, New York.

Barto, A. G. (1989). From chemotaxis to cooperativity: Abstract exercises in neuronal learning strategies. In Durbin, R., Maill, R., and Mitchison, G., editors, *The Computing Neuron*, pages 73–98. Addison-Wesley, Reading, MA.

Barto, A. G. (1990). Connectionist learning for control: An overview. In T. Miller, R. S. Sutton, and P. J. Werbos (eds.), *Neural Networks for Control*, pp. 5–58. MIT Press, Cambridge, MA.

Barto, A. G. (1991). Some learning tasks from a control perspective. In L. Nadel and D. L. Stein (eds.), *1990 Lectures in Complex Systems*, pp. 195–223. Addison-Wesley, Redwood City, CA.

Barto, A. G. (1992). Reinforcement learning and adaptive critic methods. In D. A. White and D. A. Sofge (eds.), *Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches*, pp. 469–491. Van Nostrand Reinhold, New York.

Barto, A. G. (1995a). Adaptive critics and the basal ganglia. In J. C. Houk, J. L. Davis, and D. G. Beiser (eds.), *Models of Information Processing in the Basal Ganglia*, pp. 215–232. MIT Press, Cambridge, MA.

Barto, A. G. (1995b). Reinforcement learning. In M. A. Arbib (ed.), *Handbook of Brain Theory and Neural Networks*, pp. 804–809. MIT Press, Cambridge, MA.

Barto, A. G. (2013). Intrinsic motivation and reinforcement learning. In Baldassarre, G. and Mirolli, M., editors, *Intrinsically Motivated Learning in Natural and Artificial Systems*, pages 17–47. Springer-Verlag, Berlin.

Barto, A. G., Anandan, P. (1985). Pattern recognizing stochastic learning automata. *IEEE Transactions on Systems, Man, and Cybernetics*, 15:360–375.

Barto, A. G., Anderson, C. W. (1985). Structural learning in connectionist systems. In *Program of the Seventh Annual Conference of the Cognitive Science Society*, pp. 43–54.

Barto, A. G., Anderson, C. W., Sutton, R. S. (1982). Synthesis of nonlinear control surfaces by a layered associative search network. *Biological Cybernetics*, 43:175–185.

Barto, A. G., Bradtke, S. J., Singh, S. P. (1991). Real-time learning and control using asynchronous dynamic programming. Technical Report 91-57. Department of Computer and Information Science, University of Massachusetts, Amherst.

Barto, A. G., Bradtke, S. J., Singh, S. P. (1995). Learning to act using real-time dynamic programming. *Artificial Intelligence*, 72:81–138.

Barto, A. G., Duff, M. (1994). Monte Carlo matrix inversion and reinforcement learning. In J. D. Cohen, G. Tesauro, and J. Alspector (eds.), *Advances in Neural Information Processing Systems: Proceedings of the 1993 Conference*, pp. 687–694. Morgan Kaufmann, San Francisco.

Barto, A. G., Jordan, M. I. (1987). Gradient following without back-propagation in layered networks. In M. Caudill and C. Butler (eds.), *Proceedings of the IEEE First Annual Conference on Neural Networks*, pp. II629–II636. SOS Printing, San Diego, CA.

Barto, A. G., Singh, S., and Chentanez, N. (2004). Intrinsically motivated learning of hierarchical collections of skills. In *International Conference on Developmental Learning (ICDL)*, LaJolla, CA.

Barto, A. G., Sutton, R. S. (1981a). Goal seeking components for adaptive intelligence: An initial assessment. Technical Report AFWAL-TR-81-1070. Air Force Wright Aeronautical Laboratories/Avionics Laboratory, Wright-Patterson AFB, OH.

Barto, A. G., Sutton, R. S. (1981b). Landmark learning: An illustration of associative search. *Biological Cybernetics*, 42:1–8.

Barto, A. G., Sutton, R. S. (1982). Simulation of anticipatory responses in classical conditioning by a neuron-like adaptive element. *Behavioural Brain Research*, 4:221–235.

Barto, A. G., Sutton, R. S., Anderson, C. W. (1983). Neuronlike elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, 13:835–846. Reprinted in J. A. Anderson and E. Rosenfeld (eds.), *Neurocomputing: Foundations of Research*, pp. 535–549. MIT Press, Cambridge, MA, 1988.

Barto, A. G., Sutton, R. S., Brouwer, P. S. (1981). Associative search network: A reinforcement learning associative memory. *Biological Cybernetics*, 40:201–211.

Bellman, R. E. (1956). A problem in the sequential design of experiments. *Sankhya*, 16:221–229.

Bellman, R. E. (1957a). *Dynamic Programming*. Princeton University Press, Princeton.

Bellman, R. E. (1957b). A Markov decision process. *Journal of Mathematical Mechanics*, 6:679–684.

Bellman, R. E., Dreyfus, S. E. (1959). Functional approximations and dynamic programming. *Mathematical Tables and Other Aids to Computation*, 13:247–251.

Bellman, R. E., Kalaba, R., Kotkin, B. (1973). Polynomial approximation—A new computational technique in dynamic programming: Allocation processes. *Mathematical Computation*, 17:155–161.

Bengio, Y. (2009). Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–27.

Berg, H. C. (1975). Chemotaxis in bacteria. *Annual review of biophysics and bioengineering*, 4(1):119–136.

Bernoulli, D. (1954). Exposition of a new theory on the measurement of risk. *Econometrica*, 22(1):23–36. English translation of the 1738 paper.

Berridge, K. C. and Kringelbach, M. L. (2008). Affective neuroscience of pleasure: reward in humans and animals. *Psychopharmacology*, 199(3):457–480.

Berridge, K. C. and Robinson, T. E. (1998). What is the role of dopamine in reward: hedonic impact, reward learning, or incentive salience? *Brain Research Reviews*, 28(3):309–369.

Berry, D. A., Fristedt, B. (1985). *Bandit Problems*. Chapman and Hall, London.

Bertsekas, D. P. (1982). Distributed dynamic programming. *IEEE Transactions on Automatic Control*, 27:610–616.

Bertsekas, D. P. (1983). Distributed asynchronous computation of fixed points. *Mathematical Programming*, 27:107–120.

Bertsekas, D. P. (1987). *Dynamic Programming: Deterministic and Stochastic Models*. Prentice-Hall, Englewood Cliffs, NJ.

Bertsekas, D. P. (2005). *Dynamic Programming and Optimal Control, Volume 1,* third edition. Athena Scientific, Belmont, MA.

Bertsekas, D. P. (2012). *Dynamic Programming and Optimal Control, Volume 2: Approximate Dynamic Programming*, fourth edition. Athena Scientific, Belmont, MA.

Bertsekas, D. P., Tsitsiklis, J. N. (1989). *Parallel and Distributed Computation: Numerical Methods*. Prentice-Hall, Englewood Cliffs, NJ.

Bertsekas, D. P., Tsitsiklis, J. N. (1996). *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA.

Bertsekas, D. P., Yu, H. (2009). Projected equation methods for approximate solution of large linear systems. *Journal of Computational and Applied Mathematics*, 227(1):27–50.

Biermann, A. W., Fairfield, J. R. C., Beres, T. R. (1982). Signature table systems and learning. *IEEE Transactions on Systems, Man, and Cybernetics*, 12:635–648.

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. Clarendon, Oxford.

Blodgett, H. C. (1929). The effect of the introduction of reward upon the maze performance of rats. *University of California Publications in Psychology*, 4:113–134.

Boakes, R. A. and Costa, D. S. J. (2014). Temporal contiguity in associative learning: Iinterference and decay from an historical perspective. *Journal of Experimental Psychology: Animal Learning and Cognition*, 40(4):381–400.

Booker, L. B. (1982). *Intelligent Behavior as an Adaptation to the Task Environment*. Ph.D. thesis, University of Michigan, Ann Arbor.

Boone, G. (1997). Minimum-time control of the acrobot. In *1997 International Conference on Robotics and Automation*, pp. 3281–3287. IEEE Robotics and Automation Society.

Boutilier, C., Dearden, R., Goldszmidt, M. (1995). Exploiting structure in policy construction. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pp. 1104–1111. Morgan Kaufmann.

Boyan, J. (2002). Technical update: Least-squares temporal difference learning. *Machine Learning 49*:233–246.

Boyan, J. A., Moore, A. W. (1995). Generalization in reinforcement learning: Safely approximating the value function. In G. Tesauro, D. S. Touretzky, and T. Leen (eds.), *Advances in Neural Information Processing Systems: Proceedings of the 1994 Confer-*

ence, pp. 369–376. MIT Press, Cambridge, MA.

Bradtke, S. J. (1993). Reinforcement learning applied to linear quadratic regulation. In S. J. Hanson, J. D. Cowan, and C. L. Giles (eds.), *Advances in Neural Information Processing Systems: Proceedings of the 1992 Conference*, pp. 295–302. Morgan Kaufmann, San Mateo, CA.

Bradtke, S. J. (1994). *Incremental Dynamic Programming for On-Line Adaptive Optimal Control*. Ph.D. thesis, University of Massachusetts, Amherst. Appeared as CMPSCI Technical Report 94-62.

Bradtke, S. J., Barto, A. G. (1996). Linear least–squares algorithms for temporal difference learning. *Machine Learning*, 22:33–57.

Bradtke, S. J., Ydstie, B. E., Barto, A. G. (1994). Adaptive linear quadratic control using policy iteration. In *Proceedings of the American Control Conference*, pp. 3475–3479. American Automatic Control Council, Evanston, IL.

Bradtke, S. J., Duff, M. O. (1995). Reinforcement learning methods for continuous-time Markov decision problems. In G. Tesauro, D. Touretzky, and T. Leen (eds.), *Advances in Neural Information Processing Systems: Proceedings of the 1994 Conference*, pp. 393–400. MIT Press, Cambridge, MA.

Brafman, R. I., Tennenholtz, M. (2003). R-max – a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3, 213–231.

Breland, K. and Breland, M. (1961). The misbehavior of organisms. *American Psychologist*, 16(11):681–684.

Bridle, J. S. (1990). Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimates of parameters. In D. S. Touretzky (ed.), *Advances in Neural Information Processing Systems: Proceedings of the 1989 Conference*, pp. 211–217. Morgan Kaufmann, San Mateo, CA.

Broomhead, D. S., Lowe, D. (1988). Multivariable functional interpolation and adaptive networks. *Complex Systems*, 2:321–355.

Bromberg-Martin, E. S., Matsumoto, M., Hong, S., and Hikosaka, O. (2010). A pallidus-habenula-dopamine pathway signals inferred stimulus values. *Journal of Neurophysiology*, 104(2):1068–1076.

Brown, J., Bullock, D., and Grossberg, S. (1999). How the basal ganglia use parallel excitatory and inhibitory learning pathways to selectively respond to unexpected rewarding cues. *The Journal of Neuroscience*, 19(23):10502–10511.

Bryson, A. E., Jr. (1996). Optimal control—1950 to 1985. *IEEE Control Systems*, 13(3):26–33.

Buchanan, B. G., Mitchell, T., Smith, R. G., and Jr., C. R. J. (1978). Models of learning systems. *Encyclopeadia of Computer Science and technology*, 11.

Burke, C. J., Dreher, J.-C., Seymour, B., and Tobler, P. N. (2014). State-dependent value representation: evidence from the stiatum. *Frontiers in Neuroscience*, 8.

Bush, R. R., Mosteller, F. (1955). *Stochastic Models for Learning*. Wiley, New York.

Byrne, J. H., Gingrich, K. J., Baxter, D. A. (1990). Computational capabilities of single neurons: Relationship to simple forms of associative and nonassociative learning in *aplysia*. In R. D. Hawkins and G. H. Bower (eds.), *Computational Models of Learning*, pp. 31–63. Academic Press, New York.

Calabresi, P., Picconi, B., Tozzi, A., and Filippo, M. D. (2007). Dopamine-mediated regulation of corticostriatal synaptic plasticity. *Trends in Neuroscience*, 30(5):211–219.

Camerer, C. (2003). *Behavioral game theory: Experiments in strategic interaction*. Princeton University Press.

Campbell, D. T. (1960). Blind variation and selective survival as a general strategy in knowledge-processes. In M. C. Yovits and S. Cameron (eds.), *Self-Organizing Systems*, pp. 205–231. Pergamon, New York.

Carlström, J., Nordström, E. (1997). Control of self-similar ATM call traffic by reinforcement learning. In *Proceedings of the International Workshop on Applications of Neural Networks to Telecommunications 3*, pp. 54–62. Erlbaum, Hillsdale, NJ.

Chapman, D., Kaelbling, L. P. (1991). Input generalization in delayed reinforcement learning: An algorithm and performance comparisons. In *Proceedings of the Twelfth International Conference on Artificial Intelligence*, pp. 726–731. Morgan Kaufmann, San Mateo, CA.

Chow, C.-S., Tsitsiklis, J. N. (1991). An optimal one-way multigrid algorithm for discrete-time stochastic control. *IEEE Transactions on Automatic Control*, 36:898–914.

Chrisman, L. (1992). Reinforcement learning with perceptual aliasing: The perceptual distinctions approach. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pp. 183–188. AAAI/MIT Press, Menlo Park, CA.

Christensen, J., Korf, R. E. (1986). A unified theory of heuristic evaluation functions and its application to learning. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pp. 148–152. Morgan Kaufmann, San Mateo, CA.

Cichosz, P. (1995). Truncating temporal differences: On the efficient implementation of TD(λ) for reinforcement learning. *Journal of Artificial Intelligence Research*, 2:287–318.

Clark, W. A., Farley, B. G. (1955). Generalization of pattern recognition in a self-organizing system. In *Proceedings of the 1955 Western Joint Computer Conference*, pp. 86–91.

Clouse, J. (1996). *On Integrating Apprentice Learning and Reinforcement Learning TITLE2*. Ph.D. thesis, University of Massachusetts, Amherst. Appeared as CMPSCI Technical Report 96-026.

Clouse, J., Utgoff, P. (1992). A teaching method for reinforcement learning systems. In *Proceedings of the Ninth International Machine Learning Conference*, pp. 92–101. Morgan Kaufmann, San Mateo, CA.

Colombetti, M., Dorigo, M. (1994). Training agent to perform sequential behavior. *Adaptive Behavior*, 2(3):247–275.

Connell, J. (1989). A colony architecture for an artificial creature. Technical Report AI-TR-1151. MIT Artificial Intelligence Laboratory, Cambridge, MA.

Connell, J., Mahadevan, S. (1993). *Robot Learning*. Kluwer Academic, Boston.

Contreras-Vidal, J. L. and Schultz, W. (1999). A predictive reinforcement model of dopamine neurons for learning approach behavior. *Journal of computational neuroscience*, 6(3):191–214.

Coulom, R. (2006). Efficient selectivity and backup operators in Monte-Carlo tree search. In *Proceedings of the 5th International Conference on Computers and Games*, pp. 72–83.

Courville, A. C., Daw, N. D., and Touretzky, D. S. (2006). Bayesian theories of conditioning in a changing world. *Trends in Cognitive Science*, 10(7):294–300.

Craik, K. J. W. (1943). *The Nature of Explanation*. Cambridge University Press, Cambridge.

Crites, R. H. (1996). *Large-Scale Dynamic Optimization Using Teams of Reinforcement Learning Agents*. Ph.D. thesis, University of Massachusetts, Amherst.

Crites, R. H., Barto, A. G. (1996). Improving elevator performance using reinforcement

learning. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo (eds.), *Advances in Neural Information Processing Systems: Proceedings of the 1995 Conference*, pp. 1017–1023. MIT Press, Cambridge, MA.

Cross, J. G. (1973). A stochastic learning model of economic behavior. *The Quarterly Journal of Economics 87*(2):239-266.

Crow, T. J. (1968). Cortical synapses and reinforcement: a hypothesis. *Nature*, 219:736–737.

Curtiss, J. H. (1954). A theoretical comparison of the efficiencies of two classical methods and a Monte Carlo method for computing one component of the solution of a set of linear algebraic equations. In H. A. Meyer (ed.), *Symposium on Monte Carlo Methods*, pp. 191–233. Wiley, New York.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314.

Cziko, G. (1995). *Without Miracles: Universal Selection Theory and the Second Darvinian Revolution*. MIT Press, Cambridge, MA.

Daniel, J. W. (1976). Splines and efficiency in dynamic programming. *Journal of Mathematical Analysis and Applications*, 54:402–407.

Daw, N. D., Courville, A. C., and Touretzky, D. S. (2003). Timing and partial observability in the dopamine system. In *Advances in neural information processing systems*, pages 99–106.

Daw, N., Niv, Y., and Dayan, P. (2005). Uncertainty based competition between prefrontal and dorsolateral striatal systems for behavioral control. *Nature Neuroscience*, 8(12):1704–1711.

Daw, N. D. and Shohamy, D. (2008). The cognitive neuroscience of motivation and learning. *Social Cognition*, 26(5):593–620.

Dayan, P. (1991). Reinforcement comparison. In D. S. Touretzky, J. L. Elman, T. J. Sejnowski, and G. E. Hinton (eds.), *Connectionist Models: Proceedings of the 1990 Summer School*, pp. 45–51. Morgan Kaufmann, San Mateo, CA.

Dayan, P. (1992). The convergence of TD(λ) for general λ. *Machine Learning*, 8:341–362.

Dayan, P. (2008). The role of value systems in decision making. In Engel, C. and Singer, W., editors, *Better Than Conscious?: Decision Making, the Human Mind, and Implications For Institutions (Strüngmann Forum Reports)*, pages 51–70. MIT Press, Cambridge, MA.

Dayan, P. and Abbott, L. F. (2001). *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*. MIT Press, Cambridge, MA.

Dayan, P. and Berridge, K. C. (2014). Model-based and model-free Pavlovian reward learning: Revaluation, revision, and revaluation. *Cognitive, Affective, & Behavioral Neuroscience*, 14(2):473–492.

Dayan, P., Hinton, G. E. (1993). Feudal reinforcement learning. In S. J. Hanson, J. D. Cohen, and C. L. Giles (eds.), *Advances in Neural Information Processing Systems: Proceedings of the 1992 Conference*, pp. 271–278. Morgan Kaufmann, San Mateo, CA.

Dayan, P. and Niv, Y. (2008). Reinforcement learning: the good, the bad and the ugly. *Current Opinion in Neurobiology*, 18(2):185–196.

Dayan, P., Sejnowski, T. (1994). TD(λ) converges with probability 1. *Machine Learning*, 14:295–301.

Dean, T., Lin, S.-H. (1995). Decomposition techniques for planning in stochastic domains. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelli-*

gence, pp. 1121–1127. Morgan Kaufmann. See also Technical Report CS-95-10, Brown University, Department of Computer Science, 1995.

DeJong, G., Spong, M. W. (1994). Swinging up the acrobot: An example of intelligent control. In *Proceedings of the American Control Conference*, pp. 2158–2162. American Automatic Control Council, Evanston, IL.

Denardo, E. V. (1967). Contraction mappings in the theory underlying dynamic programming. *SIAM Review*, 9:165–177.

Dennett, D. C. (1978). *Brainstorms*, pp. 71–89. Bradford/MIT Press, Cambridge, MA.

Deutsch, J. A. (1953). A new type of behaviour theory. *British Journal of Psychology. General Section*, 44(4):304–317.

Deutsch, J. A. (1954). A machine with insight. *Quarterly Journal of Experimental Psychology*, 6(1):6–11.

Dick, T. (2015). *A Regret-full Perspective on Policy Gradient Methods for Reinforcement Learning*. MSc Thesis, University of Alberta.

Dickinson, A. (1980). *Contemporary Animal Learning Theory*. Cambridge University Press, Cambridge.

Dickinson, A. (1985). Actions and habits: the development of behavioral autonomy. *Phil. Trans. R. Soc. Lond. B*, 308(1135):67–78.

Dickinson, A. and Balleine, B. W. (2002). The role of learning in motivation. In Gallistel, C. R., editor, *Stevens handbook of experimental psychology*, volume 3, pages 497–533. Wiley, NY.

Dietterich, T. and Buchanan, B. G. (1984). The role of the critic in learning systems. In Selfridge, O. G., Rissland, E. L., and Arbib, M. A., editors, *Adaptive Control of Ill-Defined Systems*, pages 127–147. Plenum Press, NY. Proceedings of the NATO Advanced Research Institute on Adaptive Control of Ill-defined Systems, NATO Conference Series II, Systems Science, Vol. 16.

Dietterich, T. G., Flann, N. S. (1995). Explanation-based learning and reinforcement learning: A unified view. In A. Prieditis and S. Russell (eds.), *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 176–184. Morgan Kaufmann, San Francisco.

Dolan, R. J. and Dayan, P. (2013). Goals and habits in the brain. *Neuron*, 80(2):312–325.

Doll, B. B., Simon, D. A., and Daw, N. D. (2012). The ubiquity of model-based reinforcement learning. *Current Opinion in Neurobiology*, 22:1–7.

Donahoe, J. W. and Burgos, J. E. (2000). Behavior analysis and revaluation. *Journal of the Experimental Analysis of Behavior*, 74(3):331–346.

Dorigo, M. and Colombetti, M. (1994). Robot shaping: Developing autonomous agents through learning. *Artificial Intelligence*, 71(2):321–370.

Doya, K. (1996). Temporal difference learning in continuous time and space. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo (eds.), *Advances in Neural Information Processing Systems: Proceedings of the 1995 Conference*, pp. 1073–1079. MIT Press, Cambridge, MA.

Doyle, P. G., Snell, J. L. (1984). *Random Walks and Electric Networks*. The Mathematical Association of America. Carus Mathematical Monograph 22.

Dreyfus, S. E., Law, A. M. (1977). *The Art and Theory of Dynamic Programming*. Academic Press, New York.

Duda, R. O., Hart, P. E. (1973). *Pattern Classification and Scene Analysis*. Wiley, New

York.

Duff, M. O. (1995). Q-learning for bandit problems. In A. Prieditis and S. Russell (eds.), *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 209–217. Morgan Kaufmann, San Francisco.

Estes, W. K. (1950). Toward a statistical theory of learning. *Psycholollgical Review*, 57:94–107.

Farley, B. G., Clark, W. A. (1954). Simulation of self-organizing systems by digital computer. *IRE Transactions on Information Theory*, 4:76–84.

Farries, M. A. and Fairhall, A. L. (2007). Reinforcement learning with modulated spike timingdependent synaptic plasticity. *Journal of neurophysiology*, 98(6):3648–3665.

Feldbaum, A. A. (1965). *Optimal Control Systems*. Academic Press, New York.

Finnsson, H., Björnsson, Y. (2008). Simulation-based approach to general game playing. In *Proceedings of the Association for the Advancement of Artificial Intelligence*, 259–264.

Fiorillo, C. D., Tobler, P. N., and Schultz, W. (2003). Discrete coding of reward probability and uncertainty by dopamine neurons. *Science*, 299(5614):1898–1902.

Fiorillo, C. D., Yun, S. R., and Song, M. R. (2013). Diversity and homogeneity in responses of midbrain dopamine neurons. *The Journal of Neuroscience*, 33(11):4693–4709.

Florian, R. V. (2007). Reinforcement learning through modulation of spike-timing-dependent synaptic plasticity. *Neural Computation*, 19(6):1468–1502.

Fogel, L. J., Owens, A. J., Walsh, M. J. (1966). *Artificial intelligence through simulated evolution*. John Wiley and Sons.

Frey, U. and Morris, R. G. M. (1997). Synaptic tagging and long-term potentiation. *Nature*, 385(6616):533–536.

Friston, K. J., Tononi, G., Reeke, G. N., Sporns, O., Edelman, G. M. (1994). Value-dependent selection in the brain: Simulation in a synthetic neural model. *Neuroscience*, 59:229–243.

Fu, K. S. (1970). Learning control systems—Review and outlook. *IEEE Transactions on Automatic Control*, 15:210–221.

Galanter, E., Gerstenhaber, M. (1956). On thought: The extrinsic theory. *Psychological Review*, 63:218–227.

Gallant, S. I. (1993). *Neural Network Learning and Expert Systems*. MIT Press, Cambridge, MA.

Gallistel, C. R. (2005). Deconstructing the law of effect. *Games and Economic Behavior 52*(2), 410-423.

Gällmo, O., Asplund, L. (1995). Reinforcement learning by construction of hypothetical targets. In J. Alspector, R. Goodman, and T. X. Brown (eds.), *Proceedings of the International Workshop on Applications of Neural Networks to Telecommunications 2*, pp. 300–307. Erlbaum, Hillsdale, NJ.

Gardner, M. (1973). Mathematical games. *Scientific American*, 228(1):108–115.

Gelperin, A., Hopfield, J. J., Tank, D. W. (1985). The logic of *limax* learning. In A. Selverston (ed.), *Model Neural Networks and Behavior*, pp. 247–261. Plenum Press, New York.

Genesereth, M., Thielscher, M. (2014). General game playing. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 8(2), 1–229.

Gershman, S. J., Moustafa, A. A., and Ludvig, E. A. (2013). Time representation in reinforcement learning models of the basal ganglia. *Frontiers in computational neuroscience*,

7.

Gershman, S. J. and Niv, Y. (2010). Learning latent structure: Carving nature at its joints. *Current Opinions in Neurobiology*, 20:251–256.

Gittins, J. C., Jones, D. M. (1974). A dynamic allocation index for the sequential design of experiments. In J. Gani, K. Sarkadi, and I. Vincze (eds.), *Progress in Statistics*, pp. 241–266. North-Holland, Amsterdam–London.

Glimcher, P. W. (2011). Understanding dopamine and reinforcement learning: The dopamine reward prediction error hypothesis. *Proceedings of the National Academy of Sciences*, 108(Supplement 3):15647–15654.

Glimcher, P. W. (2003). *Decisions, uncertainty, and the brain: The science of neuroeconomics*. MIT Press, Cambridge, MA.

Glimcher, P. W. and Fehr, E., editors (2013). *Neuroeconomics: Decision making and the brain, Second Edition*. Academic Press.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, MA.

Goldstein, H. (1957). *Classical Mechanics*. Addison-Wesley, Reading, MA.

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

Gopnik, A., Glymour, C., Sobel, D., Schulz, L. E., Kushnir, T., and Danks, D. (2004). A theory of causal learning in children: Causal maps and Bayes nets. *Psychological Review*, 111(1):3–32.

Gordon, G. J. (1995). Stable function approximation in dynamic programming. In A. Prieditis and S. Russell (eds.), *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 261–268. Morgan Kaufmann, San Francisco. An expanded version was published as Technical Report CMU-CS-95-103. Carnegie Mellon University, Pittsburgh, PA, 1995.

Gordon, G. J. (1996). Chattering in SARSA(λ). CMU learning lab internal report.

Gordon, G. J. (1996). Stable fitted reinforcement learning. In D. S. Touretzky, M. C. Mozer, M. E. Hasselmo (eds.), *Advances in Neural Information Processing Systems: Proceedings of the 1995 Conference*, pp. 1052–1058. MIT Press, Cambridge, MA.

Gordon, G. J. (2001). Reinforcement learning with function approximation converges to a region. *Advances in neural information processing systems*.

Greensmith, E., Bartlett, P. L., Baxter, J. (2001). Variance reduction techniques for gradient estimates in reinforcement learning. In *Advances in Neural Information Processing Systems: Proceedings of the 2000 Conference*, pp. 1507–1514.

Greensmith, E., Bartlett, P. L., Baxter, J. (2004). Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research 5*, 1471-1530.

Greybiel, A. M. (2000). The basal ganglia. *Current Biology*, 10(14):R509–R511.

Griffith, A. K. (1966). A new machine learning technique applied to the game of checkers. Technical Report Project MAC, Artificial Intelligence Memo 94. Massachusetts Institute of Technology, Cambridge, MA.

Griffith, A. K. (1974). A comparison and evaluation of three machine learning procedures as applied to the game of checkers. *Artificial Intelligence*, 5:137–148.

Grossberg, S. (1975). A neural model of attention, reinforcement, and discrimination learning. *International Review of Neurobiology*, 18:263–327.

Gullapalli, V. (1990). A stochastic reinforcement algorithm for learning real-valued functions.

Neural Networks, 3:671–692.

Gurney, K., Prescott, T. J., and Redgrave, P. (2001). A computational model of action selection in the basal ganglia I. A new functional anatomy. *Biological cybernetics*, 84(6):401–410.

Gurvits, L., Lin, L.-J., Hanson, S. J. (1994). Incremental learning of evaluation functions for absorbing Markov chains: New methods and theorems. Preprint.

Hampson, S. E. (1983). *A Neural Model of Adaptive Behavior*. Ph.D. thesis, University of California, Irvine.

Hampson, S. E. (1989). *Connectionist Problem Solving: Computational Aspects of Biological Learning*. Birkhauser, Boston.

Hare, T. A., O'Doherty, J., Camerer, C. F., Schultz, W., and Rangel, A. (2008). Dissociating the role of the orbitofrontal cortex and the striatum in the computation of goal values and prediction errors. *The Journal of Neuroscience*, 28(22):5623–5630.

Hassabis, D. and Maguire, E. A. (2007). Deconstructing episodic memory with construction. *Trends in cognitive sciences*, 11(7):299–306.

Hawkins, R. D., Kandel, E. R. (1984). Is there a cell-biological alphabet for simple forms of learning? *Psychological Review*, 91:375–391.

He, K., Huertas, M., Hong, S. Z., Tie, X., Hell, J. W., Shouval, H., and Kirkwood, A. (2015). Distinct eligibility traces for LTP and LTD in cortical synapses. *Neuron*, 88(3):528–538.

Hebb, D. O. (1949). *The organization of behavior: A neuropsychological theory*. John Wiley and Sons Inc., New York. Reissued by Lawrence Erlbaum Associates Inc., Mahwah NJ, 2002.

Herrnstein, R. J. (1970). On the Law of Effect. *Journal of the Experimental Analysis of Behavior 13*(2), 243-266.

Hersh, R., Griego, R. J. (1969). Brownian motion and potential theory. *Scientific American*, 220:66–74.

Hesterberg, T. C. (1988), *Advances in importance sampling*, Ph.D. Dissertation, Statistics Department, Stanford University.

Hilgard, E. R. (1956). *Theories of Learning, Second Edition*. Appleton-Century-Cofts, Inc., New York.

Hilgard, E. R., Bower, G. H. (1975). *Theories of Learning*. Prentice-Hall, Englewood Cliffs, NJ.

Hinton, G. E. (1984). Distributed representations. Technical Report CMU-CS-84-157. Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA.

Hinton, G. E., Osindero, S., and Teh, Y. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554.

Hochreiter, S., Schmidhuber, J. (1997). LTSM can solve hard time lag problems. In *Advances in Neural Information Processing Systems: Proceedings of the 1996 Conference*, pp. 473–479. MIT Press, Cambridge, MA.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor.

Holland, J. H. (1976). Adaptation. In R. Rosen and F. M. Snell (eds.), *Progress in Theoretical Biology*, vol. 4, pp. 263–293. Academic Press, New York.

Holland, J. H. (1986). Escaping brittleness: The possibility of general-purpose learning algorithms applied to rule-based systems. In R. S. Michalski, J. G. Carbonell, and T. M. Mitchell (eds.), *Machine Learning: An Artificial Intelligence Approach*, vol. 2,

pp. 593–623. Morgan Kaufmann, San Mateo, CA.

Hollerman, J. R. and Schultz, W. (1998). Dopmine neurons report an error in the temporal prediction of reward during learning. *Nature Neuroscience*, 1:304–309.

Houk, J. C., Adams, J. L., Barto, A. G. (1995). A model of how the basal ganglia generates and uses neural signals that predict reinforcement. In J. C. Houk, J. L. Davis, and D. G. Beiser (eds.), *Models of Information Processing in the Basal Ganglia*, pp. 249–270. MIT Press, Cambridge, MA.

Howard, R. (1960). *Dynamic Programming and Markov Processes*. MIT Press, Cambridge, MA.

Hull, C. L. (1932). The goal-gradient hypothesis and maze learning. *Psychological Review*, 39(1):25–43.

Hull, C. L. (1943). *Principles of Behavior*. Appleton-Century, New York.

Hull, C. L. (1952). *A Behavior System*. Wiley, New York.

Izhikevich, E. M. (2007). Solving the distal reward problem through linkage of STDP and dopamine signaling. *Cerebral cortex*, 17(10):2443–2452.

Jaakkola, T., Jordan, M. I., Singh, S. P. (1994). On the convergence of stochastic iterative dynamic programming algorithms. *Neural Computation*, 6:1185–1201.

Jaakkola, T., Singh, S. P., Jordan, M. I. (1995). Reinforcement learning algorithm for partially observable Markov decision problems. In G. Tesauro, D. S. Touretzky, T. Leen (eds.), *Advances in Neural Information Processing Systems: Proceedings of the 1994 Conference*, pp. 345–352. MIT Press, Cambridge, MA.

Joel, D., Niv, Y., and Ruppin, E. (2002). Actor-critic models of the basal ganglia: New anatomical and computational perspectives. *Neural networks*, 15(4):535–547.

Johanson, E. B., Killeen, P. R., Russell, V. A., Tripp, G., Wickens, J. R., Tannock, R., Williams, J., and Sagvolden, T. (2009). Origins of altered reinforcement effects in ADHD. *Behavioral and Brain Functions*, 5(7).

Johnson, A. and Redish, A. D. (2007). Neural ensembles in CA3 transiently encode paths forward of the animal at a decision point. *The Journal of neuroscience*, 27(45):12176–12189.

Kaelbling, L. P. (1993a). Hierarchical learning in stochastic domains: Preliminary results. In *Proceedings of the Tenth International Conference on Machine Learning*, pp. 167–173. Morgan Kaufmann, San Mateo, CA.

Kaelbling, L. P. (1993b). *Learning in Embedded Systems*. MIT Press, Cambridge, MA.

Kaelbling, L. P. (Ed.) (1996). Special triple issue on reinforcement learning, *Machine Learning 22*(1/2/3).

Kaelbling, L. P., Littman, M. L., Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285.

Kahneman, D. and Tversky, A. (1979). Prospect theory: An analysis of decision under risk. *Econometrica: Journal of the Econometric Society*, 47:263–291.

Kakutani, S. (1945). Markov processes and the Dirichlet problem. *Proceedings of the Japan Academy*, 21:227–233.

Kalos, M. H., Whitlock, P. A. (1986). *Monte Carlo Methods*. Wiley, New York.

Kamin, L. J. (1968). "Attention-like" processes in classical conditioning. In Jones, M. R., editor, *Miami Symposium on the Prediction of Behavior, 1967: Aversive Stimulation*, pages 9–31. University of Miami Press, Coral Gables, Florida.

Kamin, L. J. (1969). Predictability, surprise, attention, and conditioning. In Campbell, B. A. and Church, R. M., editors, *Punishment and Aversive Behavior*, pages 279–296. Appleton-Century-Crofts, New York, NY.

Kandel, E. R., Schwartz, J. H., Jessell, T. M., Siegelbaum, S. A., and Hudspeth, A. J., editors (2013). *Principles of Neural Science, Fifth Edition*. McGraw-Hill Companies, Inc.

Kanerva, P. (1988). *Sparse Distributed Memory*. MIT Press, Cambridge, MA.

Kanerva, P. (1993). Sparse distributed memory and related models. In M. H. Hassoun (ed.), *Associative Neural Memories: Theory and Implementation*, pp. 50–76. Oxford University Press, New York.

Karampatziakis, N., and Langford, J. (2010). Online importance weight aware updates. ArXiv:1011.1576.

Kashyap, R. L., Blaydon, C. C., Fu, K. S. (1970). Stochastic approximation. In J. M. Mendel and K. S. Fu (eds.), *Adaptive, Learning, and Pattern Recognition Systems: Theory and Applications*, pp. 329–355. Academic Press, New York.

Kearns, M., Singh, S. (2002). Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49(2-3), 209–232.

Keerthi, S. S., Ravindran, B. (1997). Reinforcement learning. In E. Fiesler and R. Beale (eds.), *Handbook of Neural Computation*, C3. Oxford University Press, New York.

Kehoe, E. J., Schreurs, B. G., and Graham, P. (1987). Temporal primacy overrides prior training in serial compound conditioning of the rabbits nictitating membrane response. *Animal Learning & Behavior*, 15(4):455–464.

Keiflin, R. and Janak, P. H. (2015). Dopamine prediction errors in reward learning and addiction: Ffrom theory to neural circuitry. *Neuron*, 88(2):247– 263.

Kimble, G. A. (1961). *Hilgard and Marquis' Conditioning and Learning*. Appleton-Century-Crofts, New York.

Kimble, G. A. (1967). *Foundations of Conditioning and Learning*. Appleton-Century-Crofts, New York.

Klopf, A. H. (1972). Brain function and adaptive systems—A heterostatic theory. Technical Report AFCRL-72-0164, Air Force Cambridge Research Laboratories, Bedford, MA. A summary appears in *Proceedings of the International Conference on Systems, Man, and Cybernetics*. IEEE Systems, Man, and Cybernetics Society, Dallas, TX, 1974.

Klopf, A. H. (1975). A comparison of natural and artificial intelligence. *SIGART Newsletter*, 53:11–13.

Klopf, A. H. (1982). *The Hedonistic Neuron: A Theory of Memory, Learning, and Intelligence*. Hemisphere, Washington, DC.

Klopf, A. H. (1988). A neuronal model of classical conditioning. *Psychobiology*, 16:85–125.

Kocsis, L., Szepesvári, Cs. (2006). Bandit based Monte-Carlo planning. In *Proceedings of the European Conference on Machine Learning*, 282–293. Springer Berlin Heidelberg.

Kohonen, T. (1977). *Associative Memory: A System Theoretic Approach*. Springer-Verlag, Berlin.

Koller, D., Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

Kolodziejski, C., Porr, B., and Wörgötter, F. (2009). On the asymptotic equivalence between differential Hebbian and temporal difference learning. *Neural computation*, 21(4):1173–1202.

Korf, R. E. (1988). Optimal path finding algorithms. In L. N. Kanal and V. Kumar (eds.), *Search in Artificial Intelligence*, pp. 223–267. Springer Verlag, Berlin.

Koshland, D. E. (1980). *Bacterial Chemotaxis as a Model Bhavioral System*. Raven Press, New York.

Koza, J. R. (1992). *Genetic programming: On the programming of computers by means of natural selection* (Vol. 1). MIT press.

Kraft, L. G., Campagna, D. P. (1990). A summary comparison of CMAC neural network and traditional adaptive control systems. In T. Miller, R. S. Sutton, and P. J. Werbos (eds.), *Neural Networks for Control*, pp. 143–169. MIT Press, Cambridge, MA.

Kraft, L. G., Miller, W. T., Dietz, D. (1992). Development and application of CMAC neural network-based control. In D. A. White and D. A. Sofge (eds.), *Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches*, pp. 215–232. Van Nostrand Reinhold, New York.

Kumar, P. R., Varaiya, P. (1986). *Stochastic Systems: Estimation, Identification, and Adaptive Control*. Prentice-Hall, Englewood Cliffs, NJ.

Kumar, P. R. (1985). A survey of some results in stochastic adaptive control. *SIAM Journal of Control and Optimization*, 23:329–380.

Kumar, V., Kanal, L. N. (1988). The CDP: A unifying formulation for heuristic search, dynamic programming, and branch-and-bound. In L. N. Kanal and V. Kumar (eds.), *Search in Artificial Intelligence*, pp. 1–37. Springer-Verlag, Berlin.

Kushner, H. J., Dupuis, P. (1992). *Numerical Methods for Stochastic Control Problems in Continuous Time*. Springer-Verlag, New York.

Lai, T. L., Robbins, H. (1985). Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22.

Lakshmivarahan, S. and Narendra, K. S. (1982). Learning algorithms for two-person zero-sum stochastic games with incomplete information: A unified approach. *SIAM Journal of Control and Optimization*, 20:541–552.

Lammel, S., Lim, B. K., and Malenka, R. C. (2014). Reward and aversion in a heterogeneous midbrain dopamine system. *Neuropharmacology*, 76:353–359.

Lane, S. H., Handelman, D. A., Gelfand, J. J. (1992). Theory and development of higher-order CMAC neural networks. *IEEE Control Systems 12*(2):23–30.

Lang, K. J., Waibel, A. H., Hinton, G. E. (1990). A time-delay neural network architecture for isolated word recognition. *Neural Networks*, 3:33–43.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Legenstein, R. and andW. Maass, D. P. (2008). A learning theory for reward-modulated spike-timing-dependent plasticity with application to biofeedback. *PLoS Computational Biology*, 4(10).

Levy, W. B. and Steward, D. (1983). Temporal contiguity requirements for long-term associative potentiation/depression in thehippocampus. *Neuroscience*, 8:791–797.

Lewis, F. L., Liu, D. (Eds.). (2013). *Reinforcement Learning and Approximate Dynamic Programming for Feedback Control*. John Wiley and Sons.

Lewis, R. L., Howes, A., and Singh, S. (2014). Computational rationality: Linking mechanism and behavior through utility maximization. *Topics in Cognitive Science*, 6(2):279–311.

Lin, C.-S., Kim, H. (1991). CMAC-based adaptive critic self-learning control. *IEEE*

Transactions on Neural Networks, 2:530–533.

Lin, L.-J. (1992). Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, 8:293–321.

Lin, L.-J., Mitchell, T. (1992). Reinforcement learning with hidden states. In *Proceedings of the Second International Conference on Simulation of Adaptive Behavior: From Animals to Animats*, pp. 271–280. MIT Press, Cambridge, MA.

Littman, M. L. (1994). Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the Eleventh International Conference on Machine Learning*, pp. 157–163. Morgan Kaufmann, San Francisco.

Littman, M. L., Cassandra, A. R., Kaelbling, L. P. (1995). Learning policies for partially observable environments: Scaling up. In A. Prieditis and S. Russell (eds.), *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 362–370. Morgan Kaufmann, San Francisco.

Littman, M. L., Dean, T. L., Kaelbling, L. P. (1995). On the complexity of solving Markov decision problems. In *Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence*, pp. 394–402.

Liu, J. S. (2001). *Monte Carlo strategies in scientific computing*. Berlin, Springer-Verlag.

Ljung, L. (1998). System identification. In Procházka, A., Uhl´*i*î, J., Rayner, P. W. J., and Kingsbury, N. G., editors, *Signal Analysis and Prediction*, pages 163–173. Springer Science + Business Media New York, LLC.

Ljung, L., Söderstrom, T. (1983). *Theory and Practice of Recursive Identification*. MIT Press, Cambridge, MA.

Ljungberg, T., Apicella, P., and Schultz, W. (1992). Responses of monkey dopamine neurons during learning of behavioral reactions. *Journal of Neurophysiology*, 67(1):145–163.

Lovejoy, W. S. (1991). A survey of algorithmic methods for partially observed Markov decision processes. *Annals of Operations Research*, 28:47–66.

Luce, D. (1959). *Individual Choice Behavior*. Wiley, New York.

Ludvig, E. A., Sutton, R. S., and Kehoe, E. J. (2008). Stimulus representation and the timing of reward-prediction errors in models of the dopamine system. *Neural Computation*, 20(12):3034–3054.

Ludvig, E. A., Sutton, R. S., and Kehoe, E. J. (2012). Evaluating the TD model of classical conditioning. *Learning & behavior*, 40(3):305–319.

Mackintosh, N. J. (1975). A theory of attention: Variations in the associability of stimuli with reinforcement. *Psychological Review*, 82(4):276–298.

Maclin, R., and Shavlik, J. W. (1994). Incorporating advice into agents that learn from reinforcements. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pp. 694–699. AAAI Press, Menlo Park, CA.

Mahadevan, S. (1996). Average reward reinforcement learning: Foundations, algorithms, and empirical results. *Machine Learning*, 22:159–196.

Mahadevan, S., and Connell, J. (1992). Automatic programming of behavior-based robots using reinforcement learning. *Artificial Intelligence*, 55:311–365.

Mahmood, A. R., and Sutton, R. S. (2015). Off-policy learning based on weighted importance sampling with linear computational complexity. In *Proceedings of the 31st Conference on Uncertainty in Artificial Intelligence*, Amsterdam, Netherlands.

Mahmood, A. R., Sutton, R. S., Degris, T., and Pilarski, P. M. (2012). Tuning-free step-size adaptation. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE Interna-*

tional Conference on (pp. 2121-2124). IEEE.

Markey, K. L. (1994). Efficient learning of multiple degree-of-freedom control problems with quasi-independent Q-agents. In M. C. Mozer, P. Smolensky, D. S. Touretzky, J. L. Elman, and A. S. Weigend (eds.), *Proceedings of the 1990 Connectionist Models Summer School*. Erlbaum, Hillsdale, NJ.

Markram, H., Lübke, J., Frotscher, M., and Sakmann, B. (1997). Regulation of synaptic efficacy by coincidence of postsynaptic APs and EPSPs. *Science*, 275:213–215.

Mataric, M. J. (1994). Reward functions for accelerated learning. In *Machine Learning: Proceedings of the Eleventh international conference*, pages 181–189.

Matsuda, W., Furuta, T., Nakamura, K. C., Hioki, H., Fujiyama, F., Arai, R., and Kaneko, T. (2009). Single nigrostriatal dopaminergic neurons form widely spread and highly dense axonal arborizations in the neostriatum. *The Journal of Neuroscience*, 29(2):444–453.

Mazur, J. E. (1994). *Learning and Behavior*, 3rd ed. Prentice-Hall, Englewood Cliffs, NJ.

McCallum, A. K. (1993). Overcoming incomplete perception with utile distinction memory. In *Proceedings of the Tenth International Conference on Machine Learning*, pp. 190–196. Morgan Kaufmann, San Mateo, CA.

McCallum, A. K. (1995). *Reinforcement Learning with Selective Perception and Hidden State*. Ph.D. thesis, University of Rochester, Rochester, NY.

Mendel, J. M. (1966). A survey of learning control systems. *ISA Transactions*, 5:297–303.

Mendel, J. M., McLaren, R. W. (1970). Reinforcement learning control and pattern recognition systems. In J. M. Mendel and K. S. Fu (eds.), *Adaptive, Learning and Pattern Recognition Systems: Theory and Applications*, pp. 287–318. Academic Press, New York.

Michie, D. (1961). Trial and error. In S. A. Barnett and A. McLaren (eds.), *Science Survey, Part 2*, pp. 129–145. Penguin, Harmondsworth.

Michie, D. (1963). Experiments on the mechanisation of game learning. 1. characterization of the model and its parameters. *Computer Journal*, 1:232–263.

Michie, D. (1974). *On Machine Intelligence*. Edinburgh University Press, Edinburgh.

Michie, D., Chambers, R. A. (1968). BOXES: An experiment in adaptive control. In E. Dale and D. Michie (eds.), *Machine Intelligence 2*, pp. 137–152. Oliver and Boyd, Edinburgh.

Miller, R. (1981). *Meaning and Purpose in the Intact Brain: A Philosophical, Psychological, and Biological Account of Conscious Process*. Clarendon Press, Oxford.

Miller, W. T., An, E., Glanz, F., Carter, M. (1990). The design of CMAC neural networks for control. *Adaptive and Learning Systems 1*:140–145.

Miller, W. T., Glanz, F. H. (1996). *UNH_CMAC verison 2.1: The University of New Hampshire Implementation of the Cerebellar Model Arithmetic Computer - CMAC*. Robotics Laboratory Technical Report, University of New Hampshire, Durham, New Hampshire.

Miller, S., Williams, R. J. (1992). Learning to control a bioreactor using a neural net Dyna-Q system. In *Proceedings of the Seventh Yale Workshop on Adaptive and Learning Systems*, pp. 167–172. Center for Systems Science, Dunham Laboratory, Yale University, New Haven.

Miller, W. T., Scalera, S. M., Kim, A. (1994). Neural network control of dynamic balance for a biped walking robot. In *Proceedings of the Eighth Yale Workshop on Adaptive and Learning Systems*, pp. 156–161. Center for Systems Science, Dunham Laboratory, Yale University, New Haven.

Minsky, M. L. (1954). *Theory of Neural-Analog Reinforcement Systems and Its Application to the Brain-Model Problem*. Ph.D. thesis, Princeton University.

Minsky, M. L. (1961). Steps toward artificial intelligence. *Proceedings of the Institute of Radio Engineers*, 49:8–30. Reprinted in E. A. Feigenbaum and J. Feldman (eds.), *Computers and Thought*, pp. 406–450. McGraw-Hill, New York, 1963.

Minsky, M. L. (1967). *Computation: Finite and Infinite Machines*. Prentice-Hall, Englewood Cliffs, NJ.

Montague, P. R., Dayan, P., Nowlan, S. J., Pouget, A., and Sejnowski, T. J. (1992). Using aperiodic reinforcement for directed self-organization during development. In *Advances in neural information processing systems 5*, pages 969–976.

Montague, P. R., Dayan, P., Person, C., and Sejnowski, T. J. (1995). Bee foraging in uncertain environments using predictive hebbian learning. *Nature*, 377(6551):725–728.

Montague, P. R., Dayan, P., Sejnowski, T. J. (1996). A framework for mesencephalic dopamine systems based on predictive Hebbian learning. *Journal of Neuroscience*, 16:1936–1947.

Montague, P. R. and Sejnowski, T. J. (1994). The predictive brain: Temporal coincidence and temporal order in synaptic learningmechanisms. *Learning & Memory*, 1:1–33.

Moore, A. W. (1990). *Efficient Memory-Based Learning for Robot Control*. Ph.D. thesis, University of Cambridge.

Moore, A. W. (1994). The parti-game algorithm for variable resolution reinforcement learning in multidimensional spaces. In J. D. Cohen, G. Tesauro and J. Alspector (eds.), *Advances in Neural Information Processing Systems: Proceedings of the 1993 Conference*, pp. 711–718. Morgan Kaufmann, San Francisco.

Moore, A. W., Atkeson, C. G. (1993). Prioritized sweeping: Reinforcement learning with less data and less real time. *Machine Learning*, 13:103–130.

Moore, J. W. and Blazis, D. E. J. (1989). Simulation of a classically conditioned response: A cerebellar implementation of the sutton-barto-desmond model. In Byrne, J. H. and Berry, W. O., editors, *Neural Models of Plasticity*, pages 187–207. Academic Press, San Diego, CA.

Moore, J. W., Choi, J.-S., and Brunzell, D. H. (1998). Predictive timing under temporal uncertainty: The time derivative model of the conditioned response. In Rosenbaum, D. A. and Collyer, C. E., editors, *Timing of Behavior*, pages 3–34. MIT Press, Cambridge, MA.

Moore, J. W., Desmond, J. E., Berthier, N. E., Blazis, E. J., Sutton, R. S., and Barto, A. G. (1986). Simulation of the classically conditioned nictitating membrane response by a neuron-like adaptive element: I. Response topography, neuronal firing, and interstimulus intervals. *Behavioural Brain Research*, 21:143–154.

Moore, J. W., Marks, J. S., Castagna, V. E., and Polewan, R. J. (2001). Parameter stability in the TD model of complex CR topographies. Society for Neuroscience Abstract 642.2.

Moore, J. W. and Schmajuk, N. A. (2008). Kamin blocking. *Scholarpedia*, 3(5):3542.

Moore, J. W. and Stickney, K. J. (1980). Formation of attentional-associative networks in real time:Role of the hippocampus and implications for conditioning. *Physiological Psychology*, 8(2):207–217.

Narendra, K. S., Thathachar, M. A. L. (1974). Learning automata—A survey. *IEEE Transactions on Systems, Man, and Cybernetics*, 4:323–334.

Narendra, K. S., Thathachar, M. A. L. (1989). *Learning Automata: An Introduction*. Prentice-Hall, Englewood Cliffs, NJ.

Narendra, K. S. and Wheeler, R. M. (1983). An n-player sequential stochastic game with

identical payoffs. *IEEE Transactions on Systems, Man, and Cybernetics*, 13:1154–1158.

Narendra, K. S., Wheeler, R. M. (1986). Decentralized learning in finite Markov chains. *IEEE Transactions on Automatic Control*, AC31(6):519–526.

Ng, A. Y. (2003). *Shaping and policy search in reinforcement learning*. PhD thesis, University of California, Berkeley, Berkeley, CA.

Ng, A. Y., Harada, D., and Russell, S. (1999). Policy invariance under reward transformations: Theory and application to reward shaping. In Bratko, I. and Dzeroski, S., editors, *Proceedings of the Sixteenth International Conference on Machine Learning (ICML 1999)*, volume 99, pages 278–287.

Nie, J., Haykin, S. (1996). A dynamic channel assignment policy through Q-learning. CRL Report 334. Communications Research Laboratory, McMaster University, Hamilton, Ontario.

Niv, Y. (2009). Reinforcement learning in the brain. *Journal of Mathematical Psychology*, 53(3):139–154.

Niv, Y., Daw, N. D., and Dayan, P. (2005). How fast to work: Response vigor, motivation and tonic dopamine. In Yeiss, Y., Schölkopft, B., and Platt, J., editors, *Advances in Neural Information Processing Systems 18 (NIPS 2005)*, pages 1019–1026. MIT Press, Cambridge, MA.

Niv, Y., Daw, N. D., Joel, D., and Dayan, P. (2007). Tonic dopamine: opportunity costs and the control of response vigor. *Psychopharmacology*, 191(3):507–520.

Niv, Y., Joel, D., and Dayan, P. (2006). A normative perspective on motivation. *Trends in Cognitive Sciences*, 10(8):375–381.

Nowé, A., Vrancx, P., De Hauwere, Y. M. (2012). Game theory and multi-agent reinforcement learning. In *Reinforcement Learning* (pp. 441-470). Springer Berlin Heidelberg.

Nutt, D. J., Lingford-Hughes, A., Erritzoe, D., and Stokes, P. R. A. (2015). The dopamine theory of addiction: 40 years of highs and lows. *Nature Reviews Neuroscience*, 16:305–312.

O'Doherty, J. P., Dayan, P., Friston, K., Critchley, H., and Dolan, R. J. (2003). Temporal difference models and reward-related learning in the human brain. *Neuron*, 38(2):329–337.

O'Doherty, J. P., Dayan, P., Schultz, J., Deichmann, R., Friston, K., and Dolan, R. J. (2004). Dissociable roles of ventral and dorsal striatum in instrumental conditioning. *Science*, 304(5669):452–454.

Ólafsdóttir, H. F., Barry, C., Saleem, A. B., Hassabis, D., and Spiers, H. J. (2015). Hippocampal place cells construct reward related sequences through unexplored space. *Elife*, 4:e06063.

Olds, J. and Milner, P. (1954). Positive reinforcement produced by electrical stimulation of the septal area and other regions of rat brain. *Journal of Comparative and Physiological Psychology*, 47(6):419–427.

O'Reilly, R. C. and Frank, M. J. (2006). Making working memory work: A computational model of learning in the prefrontal cortex and basal ganglia. *Neural Computation*, 18(2):283–328.

O'Reilly, R. C., Frank, M. J., Hazy, T. E., and Watz, B. (2007). PVLV: the primary value and learned value Pavlovian learning algorithm. *Behavioral neuroscience*, 121(1):31–49.

Oudeyer, P.-Y. and Kaplan, F. (2007). What is intrinsic motivation? A typology of computational approaches. *Frontiers in Neurorobotics*, 1.

Oudeyer, P.-Y., Kaplan, F., and Hafner, V. V. (2007). Intrinsic motivation systems for autonomous mental development. *IEEE Transactions on Evolutionary Computation*, 11(2):265–286.

Page, C. V. (1977). Heuristics for signature table analysis as a pattern recognition technique. *IEEE Transactions on Systems, Man, and Cybernetics*, 7:77–86.

Pan, W.-X., Schmidt, R., Wickens, J. R., and Hyland, B. I. (2005). Dopamine cells respond to predicted events during classical conditioning: Evidence for eligibility traces in the reward-learning network. *The Journal of Neuroscience*, 25(26):6235–6242.

Parks, P. C., Militzer, J. (1991). Improved allocation of weights for associative memory storage in learning control systems. *IFAC Design Methods of Control Systems*, Zurich, Switzerland, 507–512.

Parr, R., Russell, S. (1995). Approximating optimal policies for partially observable stochastic domains. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pp. 1088–1094. Morgan Kaufmann.

Pavlov, P. I. (1927). *Conditioned Reflexes*. Oxford University Press, London.

Pawlak, V. and Kerr, J. N. D. (2008). Dopamine receptor activation is required for corticostriatal spike-timing-dependent plasticity. *The Journal of Neuroscience*, 28(10):2435–2446.

Pawlak, V., Wickens, J. R., Kirkwood, A., and Kerr, J. N. D. (2010). Timing is not everything: neuromodulation opens the STDP gate. *Frontiers in synaptic neuroscience*, 2.

Pearce, J. M. and Hall, G. (1980). A model for Pavlovian learning: Variation in the effectiveness of conditioning but not unconditioned stimuli. *Psychological Review*, 87(6):532–552.

Pearl, J. (1984). *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley, Reading, MA.

Pearl, J. (1995). Causal diagrams for empirical research. *Biometrika*, 82(4), 669-688.

Pecevski, D., Maass, W., and Legenstein, R. A. (2007). Theoretical analysis of learning with reward-modulated spike-timing-dependent plasticity. In *Advances in Neural Information Processing Systems*, pages 881–888.

Peng, J. (1993). *Efficient Dynamic Programming-Based Learning for Control*. Ph.D. thesis, Northeastern University, Boston.

Peng, J., Williams, R. J. (1993). Efficient learning and planning within the Dyna framework. *Adaptive Behavior*, 1(4):437–454.

Peng, J., Williams, R. J. (1994). Incremental multi-step Q-learning. In W. W. Cohen and H. Hirsh (eds.), *Proceedings of the Eleventh International Conference on Machine Learning*, pp. 226–232. Morgan Kaufmann, San Francisco.

Peng, J., Williams, R. J. (1996). Incremental multi-step Q-learning. *Machine Learning*, 22:283–290.

Peters, J. and Büchel, C. (2010). Neural representations of subjective reward value. *Behavioral brain research*, 213(2):135–141.

Peterson, G. B. (2004). A day of great illumination: B.F. Skinner's discovery of shaping. *Journal of the Experimental Analysis of Behavior*, 82(3):317–28.

Pezzulo, G., van der Meer, M. A. A., Lansink, C. S., and Pennartz, C. M. A. (2014). Internally generated sequences in learning and executing goal-directed behavior. *Trends in Cognitive Science*, 18(12):647–657.

Phansalkar, V. V., Thathachar, M. A. L. (1995). Local and global optimization algorithms

for generalized learning automata. *Neural Computation*, 7:950–973.

Poggio, T., Girosi, F. (1989). A theory of networks for approximation and learning. A.I. Memo 1140. Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA.

Poggio, T., Girosi, F. (1990). Regularization algorithms for learning that are equivalent to multilayer networks. *Science*, 247:978–982.

Powell, M. J. D. (1987). Radial basis functions for multivariate interpolation: A review. In J. C. Mason and M. G. Cox (eds.), *Algorithms for Approximation*, pp. 143–167. Clarendon Press, Oxford.

Powell, W. B. (2011). *Approximate Dynamic Programming: Solving the Curses of Dimensionality*, Second edition. John Wiley and Sons.

Powers, W. T. (1973). *Behavior: The Control of Perception*. Aldine de Gruyter, Chicago. 2nd expanded edition 2005.

Precup, D., Sutton, R. S., Dasgupta, S. (2001). Off-policy temporal-difference learning with function approximation. In *Proceedings of the 18th International Conference on Machine Learning*.

Precup, D., Sutton, R. S., Singh, S. (2000). Eligibility traces for off-policy policy evaluation. In *Proceedings of the 17th International Conference on Machine Learning*, pp. 759–766. Morgan Kaufmann.

Puterman, M. L. (1994). *Markov Decision Problems*. Wiley, New York.

Puterman, M. L., Shin, M. C. (1978). Modified policy iteration algorithms for discounted Markov decision problems. *Management Science*, 24:1127–1137.

Quartz, S., Dayan, P., Montague, P. R., and Sejnowski, T. J. (1992). Expectation learning in the brain using diffuse ascending connections. In *Society for Neuroscience Abstracts*, volume 18, page 1210.

Randløv, J. and Alstrøm, P. (1998). Learning to drive a bicycle using reinforcement learning and shaping. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 463–471.

Rangel, A., Camerer, C., and Montague, P. R. (2008). A framework for studying the neurobiology of value-based decision making. *Nature Reviews Neuroscience*, 9(7):545–556.

Rangel, A. and Hare, T. (2010). Neural computations associated with goal-directed choice. *Current opinion in neurobiology*, 20(2):262–270.

Redgrave, P. and Gurney, K. (2006). The short-latency dopamine signal: a role in discovering novel actions? *Nature Reviews Neuroscience*, 7:967–975.

Redish, D. A. (2004). Addiction as a computational process gone awry. *Science*, 306(5703):1944–1947.

Reetz, D. (1977). Approximate solutions of a discounted Markovian decision process. *Bonner Mathematische Schriften*, 98:77–92.

Rescorla, R. A. and Wagner, A. R. (1972). A theory of Pavlovian conditioning: Variations in the effectiveness of reinforcement and nonreinforcement. In Black, A. H. and Prokasy, W. F., editors, *Classical Conditioning II*, pages 64–99. Appleton-Century-Crofts, New York.

Revusky, S. and Garcia, J. (1970). Learned associations over long delays. In Bower, G., editor, *The psychology of learning and motivation*, volume 4, pages 1–84. Academic Press, Inc., New York.

Reynolds, J. N. J. and Wickens, J. R. (2002). Dopamine-dependent plasticity of corticostriatal synapses. *Neural Networks*, 15(4):507–521.

Ring, M. B. (1994). *Continual Learning in Reinforcement Environments*. Ph.D. thesis, University of Texas, Austin.

Rivest, R. L., Schapire, R. E. (1987). Diversity-based inference of finite automata. In *Proceedings of the Twenty-Eighth Annual Symposium on Foundations of Computer Science*, pp. 78–87. Computer Society Press of the IEEE, Washington, DC.

Robbins, H. (1952). Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 58:527–535.

Robertie, B. (1992). Carbon versus silicon: Matching wits with TD-Gammon. *Inside Backgammon*, 2:14–22.

Roesch, M. R., Calu, D. J., and Schoenbaum, G. (2007). Dopamine neurons encode the better option in rats deciding between differently delayed or sized rewards. *Nature Neuroscience*, 10(12):1615–1624.

Romo, R. and Schultz, W. (1990). Dopamine neurons of the monkey midbrain: Contingencies of responses to active touch during self-initiated arm movements. *Journal of Neurophysiology*, 63(3):592–624.

Rosenblatt, F. (1962). *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Spartan Books, Washington, DC.

Ross, S. (1983). *Introduction to Stochastic Dynamic Programming*. Academic Press, New York.

Ross, T. (1933). Machines that think. *Scientific American*, pages 206–208.

Rubinstein, R. Y. (1981). *Simulation and the Monte Carlo Method*. Wiley, New York.

Rumelhart, D. E., Hinton, G. E., Williams, R. J. (1986). Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland (eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. I, *Foundations*. Bradford/MIT Press, Cambridge, MA.

Rummery, G. A. (1995). *Problem Solving with Reinforcement Learning*. Ph.D. thesis, Cambridge University.

Rummery, G. A., Niranjan, M. (1994). On-line Q-learning using connectionist systems. Technical Report CUED/F-INFENG/TR 166. Engineering Department, Cambridge University.

Russell, S., Norvig, P. (2009). *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ.

Rust, J. (1996). Numerical dynamic programming in economics. In H. Amman, D. Kendrick, and J. Rust (eds.), *Handbook of Computational Economics*, pp. 614–722. Elsevier, Amsterdam.

Ryan, R. M. and Deci, E. L. (2000). Intrinsic and extrinsic motivations: Classic definitions and new directions. *Contemporary Educational Psychology*, 25(1):54–67.

Saddoris, M. P., Cacciapaglia, F., Wightmman, R. M., and Carelli, R. M. (2015). Differential dopamine release dynamics in the nucleus accumbens core and shell reveal complementary signals for error prediction and incentive motivation. *The Journal of Neuroscience*, 35(33):11572–11582.

Saksida, L. M., Raymond, S. M., and Touretzky, D. S. (1997). Shaping robot behavior using principles from instrumental conditioning. *Robotics and Autonomous Systems*, 22(3):231–249.

Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal on Research and Development*, 3:211–229. Reprinted in E. A. Feigenbaum and J. Feldman (eds.), *Computers and Thought*, pp. 71–105. McGraw-Hill, New York, 1963.

Samuel, A. L. (1967). Some studies in machine learning using the game of checkers. II—Recent progress. *IBM Journal on Research and Development*, 11:601–617.

Schmajuk, N. A. (2008). Computational models of classical conditioning. *Scholarpedia*, 3(3):1664.

Schmidhuber, J. (1991a). Adaptive confidence and adaptive curiosity. Technical Report FKI-149-91, Institut für Informatik, Technische Universität München, Arcisstr. 21, 800 München 2, Germany.

Schmidhuber, J. (1991b). A possibility for implementing curiosity and boredom in model-building neural controllers. In *From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior*, pages 222–227, Cambridge, MA. MIT Press.

Schmidhuber, J. (2009). Driven by compression progress: A simple principle explains essential aspects of subjective beauty, novelty, surprise, interestingness, attention, curiosity, creativity, art, science, music, jokes. In Pezzulo, G., Butz, M. V., Sigaud, O., and Baldassarre, G., editors, *Anticipatory Behavior in Adaptive Learning Systems. From Psychological Theories to Artificial Cognitive Systems*, pages 48–76. Springer, Berlin.

Schmidhuber, J., Storck, J., and Hochreiter, S. (1994). Reinforcement driven information acquisition in nondeterministic environments. Technical report, Fakultät für Informatik, Technische Universität München, München, Germany.

Schultz, D. G., Melsa, J. L. (1967). *State Functions and Linear Control Systems.* McGraw-Hill, New York.

Schultz, W. (1998). Predictive reward signal of dopamine neurons. *Journal of Neurophysiology*, 80:1–27.

Schultz, W., Dayan, P., Montague, P. R. (1997). A neural substrate of prediction and reward. *Science*, 275:1593–1598.

Schultz, W. and Romo, R. (1990). Dopamine neurons of the monkey midbrain: contingencies of responses to stimuli eliciting immediate behavioral reactions. *Journal of Neurophysiology*, 63(3):607–624.

Schwartz, A. (1993). A reinforcement learning method for maximizing undiscounted rewards. In *Proceedings of the Tenth International Conference on Machine Learning*, pp. 298–305. Morgan Kaufmann, San Mateo, CA.

Schweitzer, P. J., Seidmann, A. (1985). Generalized polynomial approximations in Markovian decision processes. *Journal of Mathematical Analysis and Applications*, 110:568–582.

Selfridge, O. G. (1978). Tracking and trailing: Adaptation in movement strategies. Technical report, Bolt Beranek and Newman, Inc. Unpublished report.

Selfridge, O. G. (1984). Some themes and primitives in ill-defined systems. In Selfridge, O. G., Rissland, E. L., and Arbib, M. A., editors, *Adaptive Control of Ill-Defined Systems*, pages 21–26. Plenum Press, NY. Proceedings of the NATO Advanced Research Institute on Adaptive Control of Ill-defined Systems, NATO Conference Series II, Systems Science, Vol. 16.

Selfridge, O. J., Sutton, R. S., Barto, A. G. (1985). Training and tracking in robotics. In A. Joshi (ed.), *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pp. 670–672. Morgan Kaufmann, San Mateo, CA.

Seo, H., Barraclough, D., and Lee, D. (2007). Dynamic signals related to choices and outcomes in the dorsolateral prefrontal cortex. *Cerebral Cortex*, 17(suppl 1):110–117.

Seung, H. S. (2003). Learning in spiking neural networks by reinforcement of stochastic synaptic transmission. *Neuron*, 40(6):1063–1073.

Shah, A. (2012). Psychological and neuroscientific connections with reinforcement learning. In Wiering, M. and van Otterlo, M., editors, *Reinforcement Learning: State of the Art*, pages 507–537. Springer-Verlag, Berlin.

Shannon, C. E. (1950). Programming a computer for playing chess. *Philosophical Magazine*, 41:256–275.

Shannon, C. E. (1951). Presentation of a maze-solving machine. In Forester, H. V., editor, *Cybernetics. Transactions of the Eighth Conference*, pages 173–180. Josiah Macy Jr. Foundation.

Shannon, C. E. (1952). "Theseus" maze-solving mouse. http://cyberneticzoo.com/mazesolvers/1952--theseus-maze-solving-mouse--claude-shannon-american/.

Shelton, C. R. (2001). *Importance Sampling for Reinforcement Learning with Multiple Objectives*. PhD thesis, Massachusetts Institute of Technology.

Sherman, J., Morrison, W. J. (1949). Adjustment of an inverse matrix corresponding to changes in the elements of a given column or a given row of the original matrix (abstract). *Annals of Mathematical Statistics 20*:621.

Shewchuk, J., Dean, T. (1990). Towards learning time-varying functions with high input dimensionality. In *Proceedings of the Fifth IEEE International Symposium on Intelligent Control*, pp. 383–388. IEEE Computer Society Press, Los Alamitos, CA.

Shimansky, Y. P. (2009). Biologically plausible learning in neural networks: a lesson from bacterial chemotaxis. *Biological Cybernetics*, 101(5-6):379–385.

Si, J., Barto, A., Powell, W., Wunsch, D. (Eds.). (2004). *Handbook of learning and approximate dynamic programming*. John Wiley and Sons.

Silver, D. (2009). *Reinforcement learning and simulation based search in the game of Go*. University of Alberta Doctoral dissertation.

Singh, S. P. (1992a). Reinforcement learning with a hierarchy of abstract models. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pp. 202–207. AAAI/MIT Press, Menlo Park, CA.

Singh, S. P. (1992b). Scaling reinforcement learning algorithms by learning variable temporal resolution models. In *Proceedings of the Ninth International Machine Learning Conference*, pp. 406–415. Morgan Kaufmann, San Mateo, CA.

Singh, S. P. (1993). *Learning to Solve Markovian Decision Processes*. Ph.D. thesis, University of Massachusetts, Amherst. Appeared as CMPSCI Technical Report 93-77.

Singh, S. P. (Ed.) (2002). Special double issue on reinforcement learning, *Machine Learning 49*(2/3).

Singh, S., Barto, A. G., and Chentanez, N. (2005). Intrinsically motivated reinforcement learning. In *Advances in Neural Information Processing Systems 17: Proceedings of the 2004 Conference*, pages 1281–1288, Cambridge MA. MIT Press.

Singh, S. P., Bertsekas, D. (1997). Reinforcement learning for dynamic channel allocation in cellular telephone systems. In *Advances in Neural Information Processing Systems: Proceedings of the 1996 Conference*, pp. 974–980. MIT Press, Cambridge, MA.

Singh, S. P., Jaakkola, T., Jordan, M. I. (1994). Learning without state-estimation in partially observable Markovian decision problems. In W. W. Cohen and H. Hirsch (eds.),

Proceedings of the Eleventh International Conference on Machine Learning, pp. 284–292. Morgan Kaufmann, San Francisco.

Singh, S. P., Jaakkola, T., Jordan, M. I. (1995). Reinforcement learing with soft state aggregation. In G. Tesauro, D. S. Touretzky, T. Leen (eds.), *Advances in Neural Information Processing Systems: Proceedings of the 1994 Conference*, pp. 359–368. MIT Press, Cambridge, MA.

Singh, S., Lewis, R. L., and Barto, A. G. (2009). Where do rewards come from? In Taatgen, N. and van Rijn, H., editors, *Proceedings of the 31st Annual Conference of the Cognitive Science Society*, pages 2601–2606. Cognitive Science Society.

Singh, S., Lewis, R. L., Barto, A. G., and Sorg, J. (2010). Intrinsically motivated reinforcement learning: An evolutionary perspective. *IEEE Transactions on Autonomous Mental Development*, 2(2):7082. Special issue on Active Learning and Intrinsically Motivated Exploration in Robots: Advances and Challenges.

Singh, S. P., Sutton, R. S. (1996). Reinforcement learning with replacing eligibility traces. *Machine Learning*, 22:123–158.

Sivarajan, K. N., McEliece, R. J., Ketchum, J. W. (1990). Dynamic channel assignment in cellular radio. In *Proceedings of the 40th Vehicular Technology Conference*, pp. 631–637.

Skinner, B. F. (1938). *The Behavior of Organisms: An Experimental Analysis*. Appleton-Century, New York.

Skinner, B. F. (1958). Reinforcement today. *American Psychologist*, 13(3):94–99.

Skinner, B. F. (1981). Selection by consequences. *Science 213*(4507):501–504.

Smith, K. S. and Greybiel, A. M. (2013). A dual operator view of habitual behavior reflecting cortical and striatal dynamics. *Neuron*, 79(2):361–374.

Sofge, D. A., White, D. A. (1992). Applied learning: Optimal control for manufacturing. In D. A. White and D. A. Sofge (eds.), *Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches*, pp. 259–281. Van Nostrand Reinhold, New York.

Sorg, J., Singh, S., and Lewis, R. (2010). Internal rewards mitigate agent boundedness. In *Proceedings of the 27th International Conference on Machine Learning (ICML)*, pages 1007–1014.

Sorg, J. D. (2011). *The Optimal Reward Problem:Designing Effective Reward for Bounded Agents*. PhD thesis, Computer Science and Engineering, The University of Michigan.

Spence, K. W. (1947). The role of secondary reinforcement in delayed reward learning. *Psychological Review*, 54(1):1–8.

Spong, M. W. (1994). Swing up control of the acrobot. In *Proceedings of the 1994 IEEE Conference on Robotics and Automation*, pp. 2356-2361. IEEE Computer Society Press, Los Alamitos, CA.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Staddon, J. E. R. (1983). *Adaptive Behavior and Learning*. Cambridge University Press, Cambridge.

Steinberg, E. E., Keiflin, R., Boivin, J. R., Witten, I. B., Deisseroth, K., and Janak, P. H. (2013). A causal link between prediction errors, dopamine neurons and learning. *Nature Neuroscience*, 16(7):966–973.

Sterling, P. and Laughlin, S. (2015). *Principles of Neural Design*. MIT Press, Cambridge, MA.

Storck, J., Hochreiter, S., and Schmidhuber, J. (1995). Reinforcement-driven information acquisition in non-deterministic environments. In *Proceedings of ICANN'95, Paris, France*, volume 2, pages 159–164.

Sugiyama, M., Hachiya, H., Morimura, T. (2013). *Statistical Reinforcement Learning: Modern Machine Learning Approaches*. Chapman & Hall/CRC.

Suri, R. E., Bargas, J., and Arbib, M. A. (2001). Modeling functions of striatal dopamine modulation in learning and planning. *Neuroscience*, 103(1):65–85.

Suri, R. E. and Schultz, W. (1998). Learning of sequential movements by neural network model with dopamine-like reinforcement signal. *Experimental Brain Research*, 121(3):350–354.

Suri, R. E. and Schultz, W. (1999). A neural network model with dopamine-like reinforcement signal that learns a spatial delayed response task. *Neuroscience*, 91(3):871–890.

Sutton, R. S. (1978a). Learning theory support for a single channel theory of the brain. Unpublished report.

Sutton, R. S. (1978b). Single channel theory: A neuronal theory of learning. *Brain Theory Newsletter*, 4:72–75. Center for Systems Neuroscience, University of Massachusetts, Amherst, MA.

Sutton, R. S. (1978c). A unified theory of expectation in classical and instrumental conditioning. Bachelors thesis, Stanford University.

Sutton, R. S. (1984). *Temporal Credit Assignment in Reinforcement Learning*. Ph.D. thesis, University of Massachusetts, Amherst.

Sutton, R. S. (1988). Learning to predict by the method of temporal differences. *Machine Learning*, 3:9–44.

Sutton, R. S. (1990). Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the Seventh International Conference on Machine Learning*, pp. 216–224. Morgan Kaufmann, San Mateo, CA.

Sutton, R. S. (1991a). Dyna, an integrated architecture for learning, planning, and reacting. *SIGART Bulletin*, 2:160–163. ACM Press.

Sutton, R. S. (1991b). Planning by incremental dynamic programming. In L. A. Birnbaum and G. C. Collins (eds.), *Proceedings of the Eighth International Workshop on Machine Learning*, pp. 353–357. Morgan Kaufmann, San Mateo, CA.

Sutton, R. S. (Ed.) (1992). *Reinforcement Learning*. Kluwer Academic Press. Reprinting of a special double issue on reinforcement learning, *Machine Learning 8*(3/4).

Sutton, R. S. (1995). TD models: Modeling the world at a mixture of time scales. In A. Prieditis and S. Russell (eds.), *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 531–539. Morgan Kaufmann, San Francisco.

Sutton, R. S. (1996). Generalization in reinforcement learning: Successful examples using sparse coarse coding. In D. S. Touretzky, M. C. Mozer and M. E. Hasselmo (eds.), *Advances in Neural Information Processing Systems: Proceedings of the 1995 Conference*, pp. 1038–1044. MIT Press, Cambridge, MA.

Sutton, R. S. (2009). The grand challenge of predictive empirical abstract knowledge. *Working Notes of the IJCAI-09 Workshop on Grand Challenges for Reasoning from Experiences*.

Sutton, R. S., Modayil, J., Delp, M., Degris, T., Pilarski, P. M., White, A., Precup, D. (2011). Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *Proceedings of the Tenth International Conference on*

Autonomous Agents and Multiagent Systems, pp. 761–768, Taipei, Taiwan.

Sutton, R. S., Barto, A. G. (1981a). Toward a modern theory of adaptive networks: Expectation and prediction. *Psychological Review*, 88:135–170.

Sutton, R. S., Barto, A. G. (1981b). An adaptive network that constructs and uses an internal model of its world. *Cognition and Brain Theory*, 3:217–246.

Sutton, R. S., Barto, A. G. (1987). A temporal-difference model of classical conditioning. In *Proceedings of the Ninth Annual Conference of the Cognitive Science Society*, pp. 355-378. Erlbaum, Hillsdale, NJ.

Sutton, R. S., Barto, A. G. (1990). Time-derivative models of Pavlovian reinforcement. In M. Gabriel and J. Moore (eds.), *Learning and Computational Neuroscience: Foundations of Adaptive Networks*, pp. 497–537. MIT Press, Cambridge, MA.

Sutton, R. S., Mahmood, A. R., Precup, D., van Hasselt, H. (2014). A new Q(λ) with interim forward view and Monte Carlo equivalence. In *Proceedings of the 31st International Conference on Machine Learning*, Beijing, China.

Sutton, R. S., Pinette, B. (1985). The learning of world models by connectionist networks. In *Proceedings of the Seventh Annual Conference of the Cognitive Science Society*, pp. 54–64.

Sutton, R. S., Singh, S. (1994). On bias and step size in temporal-difference learning. In *Proceedings of the Eighth Yale Workshop on Adaptive and Learning Systems*, pp. 91–96. Center for Systems Science, Dunham Laboratory, Yale University, New Haven.

Sutton, R. S., Whitehead, D. S. (1993). Online learning with random representations. In *Proceedings of the Tenth International Machine Learning Conference*, pp. 314–321. Morgan Kaufmann, San Mateo, CA.

Szepesvári, C. (2010). Algorithms for reinforcement learning. Synthesis Lectures on Artificial Intelligence and Machine Learning 4(1), 1–103.

Szita, I. (2012). Reinforcement learning in games. In *Reinforcement Learning* (pp. 539-577). Springer Berlin Heidelberg.

Tadepalli, P., Ok, D. (1994). H-learning: A reinforcement learning method to optimize undiscounted average reward. Technical Report 94-30-01. Oregon State University, Computer Science Department, Corvallis.

Takahashi, Y., Schoenbaum, G., and Niv, Y. (2008). Silencing the critics: understanding the effects of cocaine sensitization on dorsolateral and ventral striatum in the context of an actor/critic model. *Frontiers in Neuroscience*, 2(1):86–99.

Tan, M. (1991). Learning a cost-sensitive internal representation for reinforcement learning. In L. A. Birnbaum and G. C. Collins (eds.), *Proceedings of the Eighth International Workshop on Machine Learning*, pp. 358–362. Morgan Kaufmann, San Mateo, CA.

Tan, M. (1993). Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the Tenth International Conference on Machine Learning*, pp. 330–337. Morgan Kaufmann, San Mateo, CA.

Tesauro, G. J. (1986). Simple neural models of classical conditioning. *Biological Cybernetics*, 55:187–200.

Tesauro, G. J. (1992). Practical issues in temporal difference learning. *Machine Learning*, 8:257–277.

Tesauro, G. J. (1994). TD-Gammon, a self-teaching backgammon program, achieves master-level play. *Neural Computation*, 6(2):215–219.

Tesauro, G. J. (1995). Temporal difference learning and TD-Gammon. *Communications of*

the ACM, 38:58–68.

Tesauro, G. J., Galperin, G. R. (1997). On-line policy improvement using Monte-Carlo search. In *Advances in Neural Information Processing Systems: Proceedings of the 1996 Conference*, pp. 1068–1074. MIT Press, Cambridge, MA.

Tham, C. K. (1994). *Modular On-Line Function Approximation for Scaling up Reinforcement Learning*. PhD thesis, Cambridge University.

Thathachar, M. A. L. and Sastry, P. S. (1985). A new approach to the design of reinforcement schemes for learning automata. *IEEE Transactions on Systems, Man, and Cybernetics*, 15:168–175.

Thathachar, M. and Sastry, P. S. (2002). Varieties of learning automata: an overview. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 36(6):711–722.

Thathachar, M. and Sastry, P. S. (2011). *Networks of learning automata: Techniques for online stochastic optimization*. Springer Science & Business Media.

Thistlethwaite, D. (1951). A critical review of latent learning and related experiments. *Psychological Bulletin*, 48(2):97–129.

Thompson, W. R. (1933). On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25:285–294.

Thompson, W. R. (1934). On the theory of apportionment. *American Journal of Mathematics*, 57:450–457.

Thorndike, E. L. (1898). Animal intelligence: An experimental study of the associative processes in animals. *The Psychological Review, Series of Monograph Supplements*, II(4).

Thorndike, E. L. (1911). *Animal Intelligence*. Hafner, Darien, CT.

Thorp, E. O. (1966). *Beat the Dealer: A Winning Strategy for the Game of Twenty-One*. Random House, New York.

Tobler, P. N., Fiorillo, C. D., and Schultz, W. (2005). Adaptive coding of reward value by dopamine neurons. *Science*, 307(5715):1642–1645.

Tolman, E. C. (1932). *Purposive Behavior in Animals and Men*. Century, New York.

Tolman, E. C. (1948). Cognitive maps in rats and men. *Psychological Review*, 55(4):189–208.

Tsai, H.-S., Zhang, F., Adamantidis, A., Stuber, G. D., Bonci, A., de Lecea, L., and Deisseroth, K. (2009). Phasic firing in dopaminergic neurons is sufficient for behavioral conditioning. *Science*, 324(5930):1080–1084.

Tsetlin, M. L. (1973). *Automaton Theory and Modeling of Biological Systems*. Academic Press, New York.

Tsitsiklis, J. N. (1994). Asynchronous stochastic approximation and Q-learning. *Machine Learning*, 16:185–202.

Tsitsiklis, J. N. (2002). On the convergence of optimistic policy iteration. *Journal of Machine Learning Research*, 3:59–72.

Tsitsiklis, J. N. and Van Roy, B. (1996). Feature-based methods for large scale dynamic programming. *Machine Learning*, 22:59–94.

Tsitsiklis, J. N., Van Roy, B. (1997). An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, 42:674–690.

Tsitsiklis, J. N., Van Roy, B. (1999). Average cost temporal-difference learning. *Automatica*, 35:1799–1808.

Turing, A. M. (1950). Computing machinery and intelligence. *Mind* 433–460.

Turing, A. M. (1948). Intelligent Machinery, A Heretical Theory. *The Turing Test: Verbal Behavior as the Hallmark of Intelligence*, 105.

Ungar, L. H. (1990). A bioreactor benchmark for adaptive network-based process control. In W. T. Miller, R. S. Sutton, and P. J. Werbos (eds.), *Neural Networks for Control*, pp. 387–402. MIT Press, Cambridge, MA.

Urbanczik, R. and Senn, W. (2009). Reinforcement learning in populations of spiking neurons. *Nature neuroscience*, 12(3):250–252.

Urbanowicz, R. J., Moore, J. H. (2009). Learning classifier systems: A complete introduction, review, and roadmap. *Journal of Artificial Evolution and Applications*.

Valentin, V. V., Dickinson, A., and O'Doherty, J. P. (2007). Determining the neural substrates of goal-directed learning in the human brain. *The Journal of Neuroscience*, 27(15):4019–4026.

van Hasselt, H. (2010). Double Q-learning. In *Advances in Neural Information Processing Systems*, pp. 2613–2621.

van Hasselt, H. (2011). *Insights in Reinforcement Learning: Formal Analysis and Empircal Evaluation of Temporal-difference Learning*. SIKS dissertation series number 2011-04.

van Hasselt, H., Sutton, R. S. (in prep.). Learning to predict independent of span.

Van Roy, B., Bertsekas, D. P., Lee, Y., Tsitsiklis, J. N. (1997). A neuro-dynamic programming approach to retailer inventory management. In *Proceedings of the 36th IEEE Conference on Decision and Control*, Vol. 4, pp. 4052–4057.

van Seijen, H., Van Hasselt, H., Whiteson, S., Wiering, M. (2009). A theoretical and empirical analysis of Expected Sarsa. In *IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, pp. 177-184.

van Seijen, H., Sutton, R. S. (2014). True online TD(λ). In *Proceedings of the 31st International Conference on Machine Learning*. JMLR W&CP 32(1):692–700.

van Seijin, H., Mahmood, A. R., Pilarski, P. M., Machado, M. C., Sutton, R. S. (in prep.). True online temporal-difference learning.

Vasilaki, E., Frémaux, N., Urbanczik, R., Senn, W., and Gerstner, W. (2009). Spike-based reinforcement learning in continuous state and action space: when policy gradient methods fail. *PLoS Computational Biology*, 5(12).

Viswanathan, R. and Narendra, K. S. (1974). Games of stochastic automata. *IEEE Transactions on Systems, Man, and Cybernetics*, 4:131–135.

Walter, W. G. (1950). An imitation of life. *Scientific American*, pages 42–45.

Walter, W. G. (1951). A machine that learns. *Scientific American*, 185(2):60–63.

Waltz, M. D., Fu, K. S. (1965). A heuristic approach to reinforcement learning control systems. *IEEE Transactions on Automatic Control*, 10:390–398.

Watkins, C. J. C. H. (1989). *Learning from Delayed Rewards*. Ph.D. thesis, Cambridge University.

Watkins, C. J. C. H., Dayan, P. (1992). Q-learning. *Machine Learning*, 8:279–292.

Wiering, M., Van Otterlo, M. (2012). *Reinforcement Learning*. Springer Berlin Heidelberg.

Werbos, P. J. (1977). Advanced forecasting methods for global crisis warning and models of intelligence. *General Systems Yearbook*, 22:25–38.

Werbos, P. J. (1982). Applications of advances in nonlinear sensitivity analysis. In R. F. Drenick and F. Kozin (eds.), *System Modeling and Optimization*, pp. 762–770. Springer-Verlag, Berlin.

Werbos, P. J. (1987). Building and understanding adaptive systems: A statistical/numerical approach to factory automation and brain research. *IEEE Transactions on Systems, Man, and Cybernetics*, 17:7–20.

Werbos, P. J. (1988). Generalization of back propagation with applications to a recurrent gas market model. *Neural Networks*, 1:339–356.

Werbos, P. J. (1989). Neural networks for control and system identification. In *Proceedings of the 28th Conference on Decision and Control*, pp. 260–265. IEEE Control Systems Society.

Werbos, P. J. (1990). Consistency of HDP applied to a simple reinforcement learning problem. *Neural Networks*, 3:179–189.

Werbos, P. J. (1992). Approximate dynamic programming for real-time control and neural modeling. In D. A. White and D. A. Sofge (eds.), *Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches*, pp. 493–525. Van Nostrand Reinhold, New York.

White, D. J. (1969). *Dynamic Programming*. Holden-Day, San Francisco.

White, D. J. (1985). Real applications of Markov decision processes. *Interfaces*, 15:73–83.

White, D. J. (1988). Further real applications of Markov decision processes. *Interfaces*, 18:55–61.

White, D. J. (1993). A survey of applications of Markov decision processes. *Journal of the Operational Research Society*, 44:1073–1096.

Whitehead, S. D., Ballard, D. H. (1991). Learning to perceive and act by trial and error. *Machine Learning*, 7:45–83.

Whitt, W. (1978). Approximations of dynamic programs I. *Mathematics of Operations Research*, 3:231–243.

Whittle, P. (1982). *Optimization over Time*, vol. 1. Wiley, New York.

Whittle, P. (1983). *Optimization over Time*, vol. 2. Wiley, New York.

Wickens, J. and Kötter, R. (1995). Cellular models of reinforcement. In Houk, J. C., Davis, J. L., and Beiser, D. G., editors, *Models of Information Processing in the Basal Ganglia*, pages 187–214. MIT Press, Cambridge, MA.

Widrow, B., Gupta, N. K., Maitra, S. (1973). Punish/reward: Learning with a critic in adaptive threshold systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 3:455–465.

Widrow, B., Hoff, M. E. (1960). Adaptive switching circuits. In *1960 WESCON Convention Record Part IV*, pp. 96–104. Institute of Radio Engineers, New York. Reprinted in J. A. Anderson and E. Rosenfeld, *Neurocomputing: Foundations of Research*, pp. 126–134. MIT Press, Cambridge, MA, 1988.

Widrow, B., Smith, F. W. (1964). Pattern-recognizing control systems. In J. T. Tou and R. H. Wilcox (eds.), *Computer and Information Sciences*, pp. 288–317. Spartan, Washington, DC.

Widrow, B., Stearns, S. D. (1985). *Adaptive Signal Processing*. Prentice-Hall, Englewood Cliffs, NJ.

Williams, R. J. (1986). Reinforcement learning in connectionist networks: A mathematical analysis. Technical Report ICS 8605. Institute for Cognitive Science, University of California at San Diego, La Jolla.

Williams, R. J. (1987). Reinforcement-learning connectionist systems. Technical Report NU-CCS-87-3. College of Computer Science, Northeastern University, Boston.

Williams, R. J. (1988). On the use of backpropagation in associative reinforcement learning. In *Proceedings of the IEEE International Conference on Neural Networks*, pp. I263–I270. IEEE San Diego section and IEEE TAB Neural Network Committee.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256.

Williams, R. J., Baird, L. C. (1990). A mathematical analysis of actor-critic architectures for learning optimal controls through incremental dynamic programming. In *Proceedings of the Sixth Yale Workshop on Adaptive and Learning Systems*, pp. 96–101. Center for Systems Science, Dunham Laboratory, Yale University, New Haven.

Wilson, R. C., Takahashi, Y. K., Schoenbaum, G., and Niv, Y. (2014). Orbitofrontal cortex as a cognitive map of task space. *Neuron*, 81(2):267–279.

Wilson, S. W. (1994). ZCS: A zeroth order classifier system. *Evolutionary Computation*, 2:1–18.

Wise, R. A. (2004). Dopamine, learning, and motivation. *Nature Reviews Neuroscience*, 5(6):1–12.

Witten, I. H. (1976). The apparent conflict between estimation and control—A survey of the two-armed problem. *Journal of the Franklin Institute*, 301:161–189.

Witten, I. H. (1977). An adaptive optimal controller for discrete-time Markov environments. *Information and Control*, 34:286–295.

Witten, I. H., Corbin, M. J. (1973). Human operators and automatic adaptive controllers: A comparative study on a particular control task. *International Journal of Man–Machine Studies*, 5:75–104.

Woodworth, R. S., Schlosberg, H. (1938). *Experimental psychology*. New York: Henry Holt and Company.

Xie, X. and Seung, H. S. (2004). Learning in neural networks by reinforcement of irregular spiking. *Physical Review E*, 69(4).

Yagishita, S., Hayashi-Takagi, A., Ellis-Davies, G. C. R., Urakubo, H., Ishii, S., and Kasai, H. (2014). A critical time window for dopamine actions on the structural plasticity of dendritic spines. *Science*, 345(6204):1616–1619.

Yee, R. C., Saxena, S., Utgoff, P. E., Barto, A. G. (1990). Explaining temporal differences to create useful concepts for evaluating states. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pp. 882–888. AAAI Press, Menlo Park, CA.

Yin, H. H. and Knowlton, B. J. (2006). The role of the basal ganglia in habit formation. *Nature Reviews Neuroscience*, 7(6):464–476.

Young, P. (1984). *Recursive Estimation and Time-Series Analysis*. Springer-Verlag, Berlin.

Yu, H. (2012). Least squares temporal difference methods: An analysis under general conditions. *SIAM Journal on Control and Optimization*, 50(6), 3310–3343.

Zhang, M., Yum, T. P. (1989). Comparisons of channel-assignment strategies in cellular mobile telephone systems. *IEEE Transactions on Vehicular Technology*, 38:211-215.

Zhang, W. (1996). *Reinforcement Learning for Job-shop Scheduling*. Ph.D. thesis, Oregon State University. Technical Report CS-96-30-1.

Zhang, W., Dietterich, T. G. (1995). A reinforcement learning approach to job-shop scheduling. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pp. 1114–1120. Morgan Kaufmann.

Zhang, W., Dietterich, T. G. (1996). High-performance job-shop scheduling with a time–delay TD(λ) network. In D. S. Touretzky, M. C. Mozer, M. E. Hasselmo (eds.), *Ad-*

vances in Neural Information Processing Systems: Proceedings of the 1995 Conference, pp. 1024–1030. MIT Press, Cambridge, MA.

Zweben, M., Daun, B., Deale, M. (1994). Scheduling and rescheduling with iterative repair. In M. Zweben and M. S. Fox (eds.), *Intelligent Scheduling*, pp. 241–255. Morgan Kaufmann, San Francisco.

Index