Blake, A., and Zisserman, A., 1987, *Visual Reconstruction*, Cambridge, MA: MIT Press.
Chellapa, R., and Jain, A., Eds., 1993, *Markov Random Fields: Theory and Practice*, Boston: Academic Press. ◆
Geman, S., and Geman, D., 1984, Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images, *IEEE Trans. Pattern Analysis Machine Intell.*, 6:721–741.
Grimson, W. E. L., 1982, A computer implementation of a theory of stereo vision, *Philos. Trans. R. Soc. Lond. B Biol. Sci.*, 298.
Kemeny, J. G., and Snell, J. L., 1960, *Finite Markov Chains and Their Applications*, New York: Van Nostrand.
Marroquin, J. L., 1992, Random measure fields and the integration of visual information, *IEEE Trans. Syst. Man Cybern.*, 22:705–716.
Marroquin, J. L., Mitter, S., and Poggio, T., 1987, Probabilistic solution of ill-posed problems in computational vision, *J. Am. Statist. Assoc.*, 82:76–89.
Marroquin, J. L., and Ramirez, A., 1991, Stochastic cellular automata with invariant Gibbsian measures, *IEEE Trans. Inform. Theory*, 37: 541–551.
Ortega, J. M., and Rheinholdt, W. C., 1970, *Iterative Solution of Nonlinear Equations in Several Variables*, New York: Academic Press.
Poggio, T., and Girosi, F., 1990, Regularization algorithms for learning that are equivalent to multilayer networks, *Science*, 247:978–992.
Poggio, T., Yang, W., and Torre, V., 1989, Optical flow: Computational properties and networks, biological and analog, in *The Computing Neuron* (R. Durbin, C. Miall, and G. Mitchison, Eds.), Reading, MA: Addison-Wesley.
Pomerleau, D. A., 1991, Efficient training of artificial neural networks for autonomous navigation, *Neural Computat.*, 3:88–97.
Tikhonov, A. N., and Arsenin, V. Y., 1977, *Solutions of Ill-Posed Problems*, Washington, DC: Winston and Sons.

Reinforcement Learning

Andrew G. Barto

Introduction

The term *reinforcement* comes from studies of animal learning in experimental psychology, where it refers to the occurrence of an event, in the proper relation to a response, that tends to increase the probability that the response will occur again in the same situation. Although not used by psychologists, the term *reinforcement learning* has been widely adopted by theorists in engineering and artificial intelligence to refer to a class of learning tasks and algorithms based on this principle of reinforcement. The simplest reinforcement learning methods are based on the common-sense idea that if an action is followed by a satisfactory state of affairs or an improvement in the state of affairs, then the tendency to produce that action is strengthened, i.e., reinforced.

Reinforcement learning is usually formulated as an *optimization problem* with the objective of finding an action or a strategy for producing actions that is optimal, or best, in some well-defined way. Although in practice it is more important that a reinforcement learning system continue to improve than it is for it to actually achieve optimal behavior, optimality objectives provide a useful categorization of reinforcement learning into three basic types, in order of increasing complexity: *nonassociative*, *associative*, and *sequential*. All these types of reinforcement learning differ from the more commonly studied paradigm of supervised learning, or "learning with a teacher," in significant ways that we discuss in the course of this article. The article REINFORCEMENT LEARNING IN MOTOR CONTROL contains additional information. For more detailed treatments, the reader should consult Barto (1992) and Sutton (1992).

Nonassociative and Associative Reinforcement Learning

Figure 1 shows the basic components of both nonassociative and associative reinforcement learning. The learning system's actions influence the behavior of some process. A *critic* sends the learning system a *reinforcement signal* whose value at any time is a measure of the "goodness" of the current process behavior. Using this information, the learning system updates its action-generation rule, generates another action, and the process repeats. In nonassociative reinforcement learning, the only input to the learning system is the reinforcement signal, whereas in the associative case, the learning system also receives stimulus patterns that provide information about the process and possibly other information as well (the dashed lines in Figure 1). Thus, whereas the objective of nonassociative reinforcement learning is to find the optimal action, the objective in the associative case is to learn an associative mapping that produces the optimal action on any trial as a function of the stimulus pattern present on that trial.

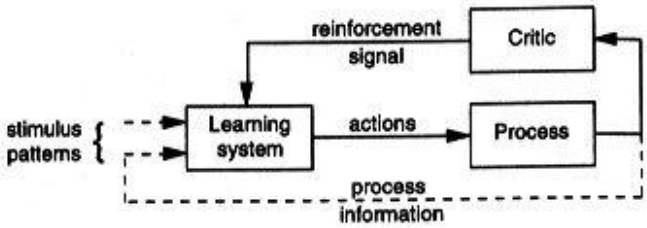

Figure 1. Nonassociative and associative reinforcement learning. A critic evaluates the actions' immediate consequences on the process and sends the learning system a reinforcement signal. In the associative case, in contrast to the nonassociative case, stimulus patterns (dashed lines) are available to the learning system in addition to the reinforcement signal.

An example of a nonassociative reinforcement learning problem has been extensively studied by learning automata theorists (Narendra and Thathachar, 1989). Suppose the learning system has m actions $a_1, a_2, \ldots, a_m$, and that the reinforcement signal simply indicates "success" or "failure." Further, assume that the influence of the learning system's actions on the reinforcement signal can be modeled as a collection of success probabilities $d_1, d_2, \ldots, d_m$, where d_i is the probability of success given that the learning system has generated a_i. The d_i's do not have to sum to one. With no initial knowledge of these values, the learning system's objective is to eventually maximize the probability of receiving "success," which is accomplished when it always performs the action a_j such that $d_j = \max\{d_i | i = 1, \ldots, m\}$.

One class of learning systems for this problem consists of *stochastic learning automata* (Narendra and Thathachar, 1989). Suppose that on each trial, or time step, t, the learning system selects an action $a(t)$ from its set of m actions according to a probability vector $(p_1(t), \ldots, p_m(t))$, where $p_i(t) = Pr\{a(t) = a_i\}$.

A stochastic learning automaton implements a common-sense notion of reinforcement learning: if action a_i is chosen on trial t and the critic's feedback is "success," then $p_i(t)$ is increased and the probabilities of the other actions are decreased; whereas if the critic indicates "failure," then $p_i(t)$ is decreased and the probabilities of the other actions are appropriately adjusted. Many methods for adjusting action probabilities have been studied, and numerous theorems have been proven about how they perform.

One can generalize this nonassociative problem to illustrate an associative reinforcement learning problem. Suppose that on trial t the learning system senses stimulus pattern $x(t)$ and selects an action $a(t) = a_i$ through a process that can depend on $x(t)$. After this action is executed, the critic signals success with probability $d_i(x(t))$ and failure with probability $1 - d_i(x(t))$. The objective of learning is to maximize success probability, achieved when on each trial t the learning system executes the action $a(t) = a_j$, where a_j is the action such that $d_j(x(t)) = \max\{d_i(x(t)) | i = 1, \ldots, m\}$. Unlike supervised learning, examples of optimal actions are not provided during training; they have to be *discovered* through exploration by the learning sytem. Learning tasks like this are related to instrumental, or cued operant, tasks used by animal learning theorists, and the stimulus patterns correspond to discriminative stimuli.

Following are key observations about both nonassociative and associative reinforcement learning:

1. *Uncertainty* plays a key role in reinforcement learning. For example, if the critic in the preceding example evaluated actions deterministically (i.e., $d_i = 1$ or 0 for each i), then the problem would be a much simpler optimization problem.
2. The critic is an abstract model of any process that evaluates the learning system's actions. It need not have direct access to the actions or have any knowledge about the interior workings of the process influenced by those actions.
3. The reinforcement signal can be any signal evaluating the learning system's actions, and not just the success/failure signal described earlier. Often it takes on real values, and the objective of learning is to maximize its expected value. Moreover, the critic can use a variety of criteria in evaluating actions, which it can combine in various ways to form the reinforcement signal.
4. The critic's signal does not directly tell the learning system what action is best; it only evaluates the action taken. The critic also does not directly tell the learning system how to change its actions. These are key features distinguishing reinforcement learning from supervised learning, and we will discuss them further.
5. Reinforcement learning algorithms are *selectional* processes. There must be *variety* in the action-generation process so that the consequences of alternative actions can be compared to select the best. Behavioral variety is called *exploration*; it is often generated through randomness (as in stochastic learning automata), but it need not be.
6. Reinforcement learning involves a conflict between *exploitation* and *exploration*. In deciding which action to take, the learning system has to balance two conflicting objectives: it has to exploit what it has already learned to obtain high evaluations, and it has to behave in new ways—explore—to learn more. Because these needs ordinarily conflict, reinforcement learning systems have to somehow balance them. In control engineering, this is known as the conflict between control and identification (see IDENTIFICATION AND CONTROL). It is absent from supervised and unsupervised learning unless the learning system is also engaged in influencing which training examples it sees (see EXPLORATION IN ACTIVE LEARNING).

Associative Reinforcement Learning Rules

Several associative reinforcement learning rules for neuron-like units have been studied. Consider a neuron-like unit receiving a stimulus pattern as input in addition to the critic's reinforcement signal. Let $x(t)$, $w(t)$, $a(t)$, and $r(t)$, respectively, denote the stimulus vector, weight vector, action, and value of the reinforcement signal at time t. Let $s(t)$ denote the weighted sum of the stimulus components at time t:

$$s(t) = \sum_{i=1}^{n} w_i(t)x_i(t)$$

where $w_i(t)$ and $x_i(t)$, respectively, are the ith components of the weight and stimulus vectors.

Associative Search Unit

One simple associative reinforcement learning rule is an extension of the Hebbian correlation learning rule. This rule, called the *associative search rule*, was motivated by Klopf's (1982) theory of the self-interested neuron. To exhibit variety in its behavior, the unit's output is a random variable depending on the activation level:

$$a(t) = \begin{cases} 1 & \text{with probability } p(t) \\ 0 & \text{with probability } 1 - p(t) \end{cases} \quad (1)$$

where $p(t)$, which must be between 0 and 1, is an increasing function (such as the logistic function) of $s(t)$. Thus, as the weighted sum increases (decreases), the unit becomes more (less) likely to fire (i.e., to produce an output of 1). If the critic takes time τ to evaluate an action, the weights are updated according to the following rule:

$$\Delta w(t) = \eta r(t)a(t - \tau)x(t - \tau) \quad (2)$$

where $r(t)$ is $+1$ (success) or -1 (failure), and $\eta > 0$ is the learning rate parameter.

This is basically the Hebbian correlation rule with the reinforcement signal acting as an additional modulatory factor. Thus, if the unit fires in the presence of an input x, possibly just by chance, and this action is followed by "success," the weights change so that the unit will be more likely to fire in the presence of x and inputs similar to x in the future. A failure signal makes it less likely to fire under these conditions. This rule makes clear the three factors minimally required for associative reinforcement learning: a stimulus signal, x; the action produced in its presence, a; and the consequent evaluation, r.

Selective Bootstrap and Associative Reward-Penalty Units

Widrow, Gupta, and Maitra (1973) extended the Widrow-Hoff, or LMS (least-mean-square), learning rule so that it could be used in associative reinforcement learning problems. They called their extension of LMS the *selective bootstrap* rule. A selective bootstrap unit's output, $a(t)$, is either 0 or 1, computed as the deterministic threshold of the weighted sum, $s(t)$. In supervised learning, an LMS unit receives a training signal, $z(t)$, that directly specifies the desired action at trial t and updates its weights as follows:

$$\Delta w(t) = \eta[z(t) - s(t)]x(t) \quad (3)$$

In contrast, a selective bootstrap unit receives a reinforcement

signal, $r(t)$, and updates its weights according to this rule:

$$\Delta w(t) = \begin{cases} \eta[a(t) - s(t)]x(t) & \text{if } r(t) = \text{success} \\ \eta[1 - a(t) - s(t)]x(t) & \text{if } r(t) = \text{failure} \end{cases}$$

where it is understood that $r(t)$ evaluates $a(t)$. Thus, if $a(t)$ produces "success," the LMS rule is applied with $a(t)$ playing the role of the desired action. Widrow et al. (1973) called this *positive bootstrap adaptation*: weights are updated as if the output actually produced was in fact the desired action. On the other hand, if $a(t)$ leads to "failure," the desired action is $1 - a(t)$, i.e., the action that was *not* produced. This is *negative bootstrap adaptation*. The reinforcement signal switches the unit between positive and negative bootstrap adaptation, motivating the term *selective bootstrap adaptation*.

A closely related unit is the *associative reward-penalty* (A_{R-P}) unit of Barto and Anandan (1985). It differs from the selective bootstrap algorithm in two ways. First, the unit's output is a random variable like that of the associative search unit (Equation 1). Second, its weight-update rule is an *asymmetric* version of the selective bootstrap rule:

$$\Delta w(t) = \begin{cases} \eta[a(t) - s(t)]x(t) & \text{if } r(t) = \text{success} \\ \lambda\eta[1 - a(t) - s(t)]x(t) & \text{if } r(t) = \text{failure} \end{cases}$$

where $0 \leq \lambda \leq 1$ and $\eta > 0$. This rule's asymmetry is important because its asymptotic performance improves as λ approaches zero, but $\lambda = 0$ is, in general, *not* optimal.

We can see from the selective bootstrap and A_{R-P} units that a reinforcement signal is less informative than a signal specifying a desired action. It is also less informative than the error $z(t) - a(t)$ used by the LMS rule. Because this error is a signed quantity, it tells the unit *how*, i.e., in what direction, it should change its action. A reinforcement signal—by itself—does not convey this information. If the learner has only two actions, it is easy to deduce, or estimate, the desired action from the reinforcement signal and the actual action. However, if there are more than two actions, the situation is more difficult because the reinforcement signal does not provide information about actions that were not taken. One way a neuron-like unit with more than two actions can perform associative reinforcement learning is illustrated by the *Stochastic Real-Valued* (SRV) unit of Gullapalli described in REINFORCEMENT LEARNING IN MOTOR CONTROL.

Weight Perturbation

For the units described in the preceding section (except the selective bootstrap unit), behavioral variability is achieved by including random variation in the unit's output. Another approach is to randomly vary the weights. Following Alspector et al. (1993), let δw be a vector of small perturbations, one for each weight, which are independently selected from some probability distribution. Letting $\mathscr{E}$ denote the function evaluating the system's behavior, the weights are updated as follows:

$$\Delta w = -\eta\left[\frac{\mathscr{E}(w + \delta w) - \mathscr{E}(w)}{\delta w}\right] \qquad (4)$$

where $\eta > 0$ is a learning rate parameter. This is a gradient descent learning rule that changes weights according to an estimate of the gradient of $\mathscr{E}$ with respect to the weights. Alspector et al. (1993) say that the method *measures* the gradient instead of *calculating* it as the LMS and error backpropagation algorithms do. This approach has been proposed by several researchers for updating the weights of a unit, or of a network, during supervised learning, where $\mathscr{E}$ gives the error over the training examples. However, $\mathscr{E}$ can be any function evaluating the unit's behavior, including a reinforcement function (in which case, the sign of the learning rule would be changed to make it a gradient *ascent* rule).

Reinforcement Learning Networks

Networks of A_{R-P} units have been used successfully in both supervised and associative reinforcement learning tasks (Barto, 1985; Barto and Jordan, 1987), although only with feedforward connection patterns. For supervised learning, the output units learn just as they do in error backpropagtion, but the hidden units learn according to the A_{R-P} rule. The reinforcement signal, which is defined to increase as the output error decreases, is simply *broadcast* to all the hidden units, which learn simultaneously. If the network as a whole faces an associative reinforcement learning task, all the units are A_{R-P} units, to which the reinforcement signal is uniformly broadcast (Barto, 1985). Another way to use reinforcement learning units in networks is to use them only as output units, with hidden units being trained via backpropagation. Weight changes of the output units determine the quantities that are backpropagated. An example is provided by a network for robot peg-in-hole insertion described in REINFORCEMENT LEARNING IN MOTOR CONTROL.

The error backpropagation algorithm can be used in another way in associative reinforcement learning problems. It is possible to train a multilayer network to form a model of the process by which the critic evaluates actions. After this model is trained sufficiently, it is possible to estimate the gradient of the reinforcement signal with respect to each component of the action vector by analytically differentiating the model's output with respect to its action inputs (which can be done efficiently by backpropagation). This gradient estimate is then used to update the parameters of an action-generation component. Jordan and Jacobs (1990) illustrate this approach.

The weight perturbation approach carries over directly to networks by simply letting w in Equation 4 be the vector consisting all the network's weights. A number of researchers have achieved success using this approach in supervised learning problems. In these cases, one can think of each weight as facing a reinforcement learning task (which is in fact nonassociative), even though the network as a whole faces a supervised learning task. An advantage of this approach is that it applies to networks with arbitrary connection patterns, not just to feedforward networks.

It should be clear from this discussion of reinforcement learning networks that there are many different approaches to solving reinforcement learning problems. Furthermore, although reinforcement learning *tasks* can be clearly distinguished from supervised and unsupervised learning tasks, it is more difficult to precisely define a class of reinforcement learning *algorithms*.

Sequential Reinforcement Learning

Sequential reinforcement requires improving the long-term consequences of an action, or of a strategy for performing actions, in addition to short-term consequences. In these problems, it can make sense to forgo short-term performance in order to achieve better performance over the long term. Tasks having these properties are examples of *optimal control problems*, sometimes called *sequential decision problems* when formulated in discrete time (see ADAPTIVE CONTROL: NEURAL NETWORK APPLICATIONS).

Figure 1, which shows the components of an associative reinforcement learning system, also applies to sequential reinforcement learning, where the box labeled "Process" is a system

being controlled. A sequential reinforcement learning system tries to influence the behavior of the process in order to maximize a measure of the total amount of reinforcement that will be received over time. In the simplest case, this measure is the sum of the future reinforcement values, and the objective is to learn an associative mapping that at each time step t selects, as a function of the stimulus pattern $x(t)$, an action $a(t)$ that maximizes

$$\sum_{k=0}^{\infty} r(t+k)$$

where $r(t+k)$ is the reinforcement signal at step $t+k$. Such an associative mapping is called a *policy*.

Because this sum might be infinite in some problems, and because the learning system usually has control only over its expected value, researchers often consider the following *expected discounted sum* instead:

$$E\{r(t) + \gamma r(t+1) + \gamma^2 r(t+2) + \cdots\} = E\left\{\sum_{k=0}^{\infty} \gamma^k r(t+k)\right\} \tag{5}$$

where E is the expectation over all possible future behavior patterns of the process. The discount factor γ determines the present value of future reinforcement: a reinforcement value received k time steps in the future is worth γ^k times what it would be worth if it were received now. If $0 \leq \gamma < 1$, this infinite discounted sum is finite as long as the reinforcement values are bounded. If $\gamma = 0$, the robot is "myopic" in being only concerned with maximizing immediate reinforcement; this is the associative reinforcement learning problem discussed earlier. As γ approaches 1, the objective explicitly takes future reinforcement into account: the robot becomes more farsighted.

An important special case of this problem occurs when there is no immediate reinforcement until a goal state is reached. This is a *delayed reward* problem in which the learning system has to learn how to make the process enter a goal state. Sometimes the objective is to make it enter a goal state as quickly as possible. A key difficulty in these problems has been called the *temporal credit-assignment problem*: When a goal state is finally reached, which of the decisions made earlier deserve credit for the resulting reinforcement? (See also REINFORCEMENT LEARNING IN MOTOR CONTROL.) A widely studied approach to this problem is to learn an *internal evaluation function* that is more informative than the evaluation function implemented by the external critic. An *adaptive critic* is a system that learns such an internal evaluation function.

Samuel's Checkers Player

Samuel's (1959) checkers-playing program has been a major influence on adaptive critic methods. The checkers player uses an evaluation function to assign a score to each board configuration; and the system makes the move expected to lead to the configuration with the highest score. Samuel used a method to improve the evaluation function through a process that compared the score of the current board position with the score of a board position likely to arise later in the game. As a result of this process of "backing up" board evaluations, the evaluation function improved in its ability to evaluate the long-term consequences of moves. If the evaluation function can be made to score each board configuration according to its true promise of eventually leading to a win, then the best strategy for playing is to myopically select each move so that the next board configuration is the most highly scored. If the evaluation function is optimal in this sense, then it already takes into account all the possible future courses of play. Methods such as Samuel's that attempt to adjust the evaluation function toward this ideal optimal evaluation function are of great utility.

Adaptive Critic Unit and Temporal Difference Methods

An adaptive critic unit is a neuron-like unit that implements a method similar to Samuel's. The unit is a neuron-like unit whose output at time step t is $P(t) = \sum_{i=1}^{n} w_i(t)x_i(t)$, so denoted because it is a *prediction* of the discounted sum of future reinforcement defined by Equation 5. The adaptive critic learning rule rests on noting that correct predictions must satisfy a consistency condition relating predictions at adjacent time steps. Suppose that the predictions at any two successive time steps, say steps t and $t+1$, are correct. This assumption means that

$$P(t) = E\{r(t) + \gamma r(t+1) + \gamma^2 r(t+2) + \cdots\}$$

$$P(t+1) = E\{r(t+1) + \gamma r(t+2) + \gamma^2 r(t+3) + \cdots\}$$

Now notice that we can rewrite $P(t)$ as follows:

$$P(t) = E\{r(t) + \gamma[r(t+1) + \gamma r(t+2) + \cdots]\}$$

But this is exactly the same as

$$P(t) = E\{r(t)\} + \gamma P(t+1)$$

An estimate of the error by which any two adjacent predictions fail to satisfy this consistency condition is called the *temporal difference (TD) error* (Sutton, 1988):

$$r(t) + \gamma P(t+1) - P(t) \tag{6}$$

where $r(t)$ is used as an unbiased estimate of $E\{r(t)\}$. The term *temporal difference* comes from the fact that this error essentially depends on the difference between the critic's predictions at successive time steps.

The adaptive critic unit adjusts its weights according to the following learning rule:

$$\Delta w(t) = \eta[r(t) + \gamma P(t+1) - P(t)]x(t) \tag{7}$$

A subtlety here is that $P(t+1)$ should be computed using the weight vector $w(t)$, not $w(t+1)$. This rule changes the weights to decrease the magnitude of the TD error. Note that if $\gamma = 0$, it is equal to the LMS learning rule (Equation 3). By analogy with the LMS rule, we can think of $r(t) + \gamma P(t+1)$ as the prediction target: it is the quantity that each $P(t)$ should match. The adaptive critic is therefore trying to predict the next reinforcement, $r(t)$, *plus its own next (discounted) prediction*, $\gamma P(t+1)$. It is similar to Samuel's learning method in adjusting weights to make current predictions closer to later predictions.

Although this method is very simple computationally, it actually converges to the correct predictions of the expected discounted sum of future reinforcement if these correct predictions can be computed by a linear unit. This finding is shown by Sutton (1988), who discusses a more general class of methods, called *TD methods*, that include Equation 7 as a special case. It is also possible to learn nonlinear predictions using, for example, multilayer networks trained by back propagating the TD error. Using this approach, Tesauro (1992) produced a system that learned how to play expert-level backgammon.

Actor-Critic Architectures

In an actor-critic architecture, the predictions formed by an adaptive critic act as reinforcement for an associative reinforcement learning component, called the *actor* (Figure 2). To distinguish the adaptive critic's signal from the reinforcement signal supplied by the original, nonadaptive critic, we

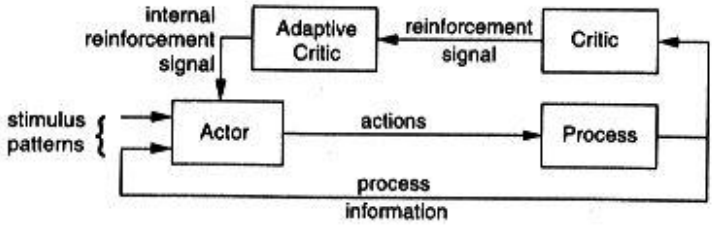

Figure 2. Actor-critic architecture. An adaptive critic provides an internal reinforcement signal to an *actor* which learns a policy for controlling the process.

call it the *internal reinforcement signal.* The actor tries to maximize the *immediate* internal reinforcement signal, while the adaptive critic tries to predict total future reinforcement. To the extent that the adaptive critic's predictions of total future reinforcement are correct given the actor's current policy, the actor actually learns to increase the total amount of future reinforcement.

Barto, Sutton, and Anderson (1983) used this architecture for learning to balance a simulated pole mounted on a cart. The actor had two actions: application of a force of a fixed magnitude to the cart in the plus or minus directions. The nonadaptive critic only provided a signal of failure when the pole fell past a certain angle or the cart hit the end of the track. The stimulus patterns were vectors representing the state of the cart-pole system. The actor was essentially an associative search unit as described above whose weights were modulated by the internal reinforcement signal.

Q-Learning

Another approach to sequential reinforcement learning combines the actor and adaptive critic into a single component that learns separate predictions for each action. At each time step the action with the largest prediction is selected, except for an exploration factor that causes other actions to be selected occasionally. An algorithm for learning predictions of future reinforcement for each action, called the *Q-learning* algorithm, was proposed by Watkins (1989), who proved that it converges to the correct predictions under certain conditions. Although the Q-learning convergence theorem requires lookup-table storage (and therefore finite state and action sets), many researchers have heuristically adapted Q-learning to more general forms of storage, including multilayer neural networks trained by backpropagating the Q-learning error.

Dynamic Programming

Sequential reinforcement learning problems (in fact, all reinforcement learning problems) are examples of stochastic optimal control problems. Among the traditional methods for solving these problems are dynamic programming (DP) algorithms. As applied to optimal control, DP consists of methods for successively approximating optimal evaluation functions and optimal policies. Bertsekas (1987) provides a good treatment of these methods. A basic operation in all DP algorithms is "backing up" evaluations in a manner similar to the operation used in Samuel's method and in the adaptive critic.

Recent reinforcement learning theory exploits connections with DP algorithms while emphasizing important differences. Following is a summary of key observations:

1. Because conventional DP algorithms require multiple exhaustive "sweeps" of the process state set (or a discretized approximation of it), they are not practical for problems with very large finite state sets or high-dimensional continuous state spaces. Sequential reinforcement learning algorithms *approximate* DP algorithms in ways designed to reduce this computational complexity.
2. Instead of requiring exhaustive sweeps, sequential reinforcement learning algorithms operate on states as they occur in actual or simulated experiences in controlling the process. It is appropriate to view them as *Monte Carlo* DP algorithms.
3. Whereas conventional DP algorithms require a complete and accurate model of the process to be controlled, sequential reinforcement learning algorithms do not require such a model. Instead of computing the required quantities (such as state evaluations) from a model, they estimate these quantities from experience. However, reinforcement learning methods can also take advantage of models to improve their efficiency.

It is therefore accurate to view sequential reinforcement learning as a collection of heuristic methods providing computationally feasible approximations of DP solutions to stochastic optimal control problems.

Discussion

The increasing interest in reinforcement learning is due to its applicability to learning by autonomous robotic agents. Although both supervised and unsupervised learning can play essential roles in reinforcement learning systems, these paradigms by themselves are not general enough for learning while acting in a dynamic and uncertain environment. Among the topics being addressed by current reinforcement learning research are these: extending the theory of sequential reinforcement learning to include generalizing function approximation methods; understanding how exploratory behavior is best introduced and controlled; sequential reinforcement learning when the process state cannot be observed; how problem-specific knowledge can be effectively incorporated into reinforcement learning systems; the design of modular and hierarchical architectures; and the relationship to brain reward mechanisms.

Road Map: Learning in Artificial Neural Networks, Deterministic
Background: I.3. Dynamics and Adaptation in Neural Networks
Related Reading: Planning, Connectionist; Problem Solving, Connectionist

References

Alspector, J., Meir, R., Yuhas, B., Jayakumar, A., and Lippe, D., 1993, A parallel gradient descent method for learning in analog VLSI neural networks, in *Advances in Neural Information Processing Systems 5* (S. J. Hanson, J. D. Cowan, and C. L. Giles, Eds.), San Mateo, CA: Morgan Kaufmann, pp. 836–844.

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

Barto, A. G., 1992, Reinforcement learning and adaptive critic methods, in *Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive*

Approaches (D. A. White and D. A. Sofge, Eds.), New York: Van Nostrand Reinhold, pp. 469–491. ◆

Barto, A. G., and Anandan, P., 1985, Pattern recognizing stochastic learning automata, *IEEE Trans. Syst. Man Cybern.*, 15:360–375.

Barto, A. G., and Jordan, M. I., 1987, Gradient following without back-propagation in layered networks, in *Proceedings of the IEEE First Annual Conference on Neural Networks* (M. Caudill and C. Butler, Eds.), San Diego: IEEE, pp. 11629–11636.

Barto, A. G., Sutton, R. S., and Anderson, C. W., 1983, Neuronlike elements that can solve difficult learning control problems, *IEEE Trans. Syst. Man Cybern.*, 13:835–846. Reprinted in *Neurocomputing: Foundations of Research* (J. A. Anderson and E. Rosenfeld, Eds.), Cambridge, MA: MIT Press, 1988.

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

Jordan, M. I., and Jacobs, R. A., 1990, Learning to control an unstable system with forward modeling, in *Advances in Neural Information Processing Systems 2* (D. S. Touretzky, Ed.), San Mateo, CA: Morgan Kaufmann, pp. 324–331.

Jordan, M. I., and Rumelhart, D. E., 1992, Supervised learning with a distal teacher, *Cognit. Sci.*, 16:307–354.

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

Narendra, K., and Thathachar, M. A. L., 1989, *Learning Automata: An Introduction*, Englewood Cliffs, NJ: Prentice Hall. ◆

Samuel, A. L., 1959, Some studies in machine learning using the game of checkers, *IBM J. Res. Develop.*, 3:210–229. Reprinted in *Computers and Thought* (E. A. Feigenbaum and J. Feldman, Eds.), New York: McGraw-Hill, 1963, pp. 71–105.

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

Sutton, R. S., Ed., 1992, *A Special Issue of Machine Learning on Reinforcement Learning, Machine Learn.*, 8. Also published as *Reinforcement Learning*, Boston: Kluwer Academic, 1992. ◆

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

Watkins, C. J. C. H., 1989, Learning from delayed rewards, PhD Thesis, Cambridge University, Cambridge, UK.

Widrow, B., Gupta, N. K., and Maitra, S., 1973, Punish/reward: Learning with a critic in adaptive threshold systems, *IEEE Trans. Syst. Man Cybern.*, 5:455–465.

Reinforcement Learning in Motor Control

Andrew G. Barto

Introduction

How do we learn motor skills such as reaching, walking, swimming, or riding a bicycle? Although there is a large literature on motor skill acquisition which is full of controversies (for a recent introduction to human motor control, see Rosenbaum, 1991), there is general agreement that motor learning requires the learner, human or not, to receive response-produced feedback through various senses providing information about performance. Careful consideration of the nature of the feedback used in learning is important for understanding the role of reinforcement learning in motor control (see REINFORCEMENT LEARNING). One function of feedback is to guide the performance of movements. This is the kind of feedback with which we are familiar from control theory, where it is the basis of servomechanisms, although its role in guiding animal movement is more complex. Another function of feedback is to provide information useful for improving *subsequent* movement. Feedback having this function has been called *learning feedback*. Note that this functional distinction between feedback for control and for learning does not mean that the signals or channels serving these functions need to be different.

Learning Feedback

When motor skills are acquired without the help of an explicit teacher or trainer, learning feedback must consist of information automatically generated by the movement and its consequences on the environment. This has been called *intrinsic feedback* (Schmidt, 1982). The "feel" of a successfully completed movement and the sight of a basketball going through the hoop are examples of intrinsic learning feedback. A teacher or trainer can augment intrinsic feedback by providing *extrinsic feedback* (Schmidt, 1982) consisting of extra information added for training purposes, such as a buzzer indicating that a movement was on target, a word of praise or encouragement, or an indication that a certain kind of error was made.

Most research in the field of artificial neural networks has focused on the learning paradigm called *supervised learning*, which emphasizes the role of training information in the form of desired, or *target*, network responses for a set of training inputs (see PERCEPTRONS, ADALINES, AND BACKPROPAGATION). The aspect of real training that corresponds most closely to the supervised learning paradigm is the trainer's role in telling or showing the learner what to do, or explicitly guiding his or her movements. These activities provide standards of correctness that the learner can try to match as closely as possible by reducing the error between its behavior and the standard. Supervised learning can also be relevant to motor learning when there is no trainer because it can use intrinsic feedback to construct various kinds of *models* that are useful for learning. Barto (1990) and Jordan and Rumelhart (1992) discuss some of the uses of models in learning control.

In contrast to supervised learning, reinforcement learning emphasizes learning feedback that *evaluates* the learner's performance without providing standards of correctness in the form of behavioral targets. Evaluative feedback tells the learner whether, and possibly by how much, its behavior has improved; or it provides a measure of the "goodness" of the behavior; or it just provides an indication of success or failure. Evaluative feedback does not directly tell the learner what it *should* have done, and although it sometimes provides the *magnitude* of an error, it does not include *directional* information telling the learner how to change its behavior, as does the error feedback of supervised learning. Although the most obvious evaluative feedback is extrinsic feedback provided by a trainer, most evaluative feedback is probably intrinsic, being derived by the learner from sensations generated by a movement and its consequences on the environment: the kinesthetic and tactile feel of a successful grasp or the swish of a basketball through the hoop. Instead of trying to match a standard of correctness, a reinforcement learning system tries to maximize the goodness of behavior as indicated by evaluative feedback. To do this, it has to actively try alternatives, compare the resulting evalua-

tions, and use some kind of selection mechanism to guide behavior toward the better alternatives. Although evaluative feedback is often called *reinforcement* feedback, it need not involve pleasure or pain.

Motor learning involves feedback carrying many different kinds of information. Consequently, it is incorrect to view motor learning strictly in terms of either supervised, reinforcement, or any other learning paradigms that have been formulated for theoretical study. Aspects of all of these paradigms play interlocking roles. However, reinforcement learning may be an essential component of motor learning.

Learning from Consequences

The simplest reinforcement learning algorithms are based on the commonsense idea that if an action is followed by a satisfactory state of affairs, or an improvement in the state of affairs, then the tendency to produce that action is strengthened, i.e., reinforced. This basic idea follows Thorndike's classical *Law of Effect* (Thorndike, 1911). Although this principle has generated considerable controversy over the years, it remains influential because its general idea is supported by many experiments and it makes such good intuitive sense (e.g., Glazer, 1971).

To illustrate how this principle applies to motor learning, we first discuss it within the general context of control. Then we describe several special cases related to motor control. Figure 1*A*, is a variation of the classical control system diagram. A controller provides control signals to a controlled system. The behavior of the controlled system is influenced by disturbances, and feedback from the controlled system to the controller provides information on which the control signals can depend. Commands to the controller specify aspects of the control task's objective.

In Figure 1*B*, the control loop is augmented with another feedback loop that provides learning feedback to the controller. In accordance with common practice in reinforcement learning, a *critic* is included that generates evaluative learning feedback on the basis of observing the control signals and their consequences on the behavior of the controlled system. The critic also needs to know the command to the controller because its evaluations must be different depending on what the controller should be trying to do. The critic is an abstraction of whatever process supplies evaluative learning feedback, both intrinsic and extrinsic, to the learning system. It is often said that the critic provides a *reinforcement signal* to the learning system. In most artificial reinforcement learning systems, the critic's output at any time is a number that scores the controller's behavior: the higher the number, the better the behavior. Assume for the moment that the behavior being scored is the immediately preceding behavior. We discuss more complex temporal relationships in later sections. For this process to work, there must be some *variability* in the controller's behavior so that the critic can evaluate many alternatives. A learning mechanism can then adjust the controller's behavior so that it tends toward behavior that is favored by the critic.

A learning rule particularly suited to reinforcement learning control systems implemented as artificial neural networks was developed by Gullapalli (1990) in the form of what he called a Stochastic Real-Valued (SRV) unit. An SRV unit's output is produced by adding a random number to the weighted sum of the components of its input pattern. The random number is drawn from a zero-mean Gaussian distribution. This random component provides the unit with the variability necessary for it to "explore" its activity space. When the reinforcement signal indicates that something good happened just after the unit emitted a particular output value in the presence of some input pattern, the unit's weights are adjusted to move the activation in the direction in which it was perturbed by the random number. This has the effect of increasing the probability that future outputs generated for that input pattern (and similar input patterns) will be closer to the output value just emitted. If the reinforcement signal indicates that something bad happened, the weights are adjusted to move future output values away from the value just emitted. Another part of the SRV learning rule decreases the variance of the Gaussian distribution as learning proceeds. This decreases the variability of the unit's behavior, with the goal of making it eventually stick (i.e., become deterministic) at the best output value for each input pattern. Using this learning rule, an SRV unit learns to produce the best output in response to each input pattern (given appropriate assumptions). Unlike more familiar supervised

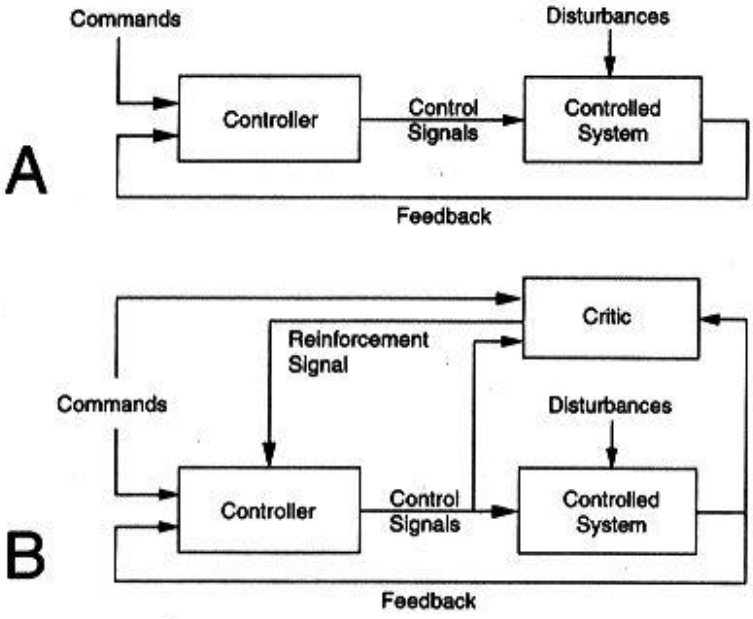

Figure 1. *A*, A basic control loop. A controller provides control signals to a controlled system, whose behavior is influenced by disturbances. Feedback from the controlled system to the controller provides information on which the control signals can depend. Commands to the controller specify aspects of the control task's objective. *B*, A control system with learning feedback. A *critic* provides the controller with a reinforcement signal evaluating its success in achieving the control objectives.

learning units, it is never given target outputs; it has to discover what outputs are best through an active exploration process.

Overcoming the Distal Error Problem

To see how reinforcement learning might work in motor learning, consider the problem of learning the control signals required to move the end of an arm from its initial position to a desired final position. This problem combines aspects of the inverse kinematics and inverse dynamics problems in robotics, where the end of the arm is called the end-effector. Many researchers have used artificial neural networks for these problems, but relatively few have employed reinforcement learning. Lipitkas et al. (1993) provide a simple example of a reinforcement learning approach. To avoid the complexity of transforming each desired arm trajectory into corresponding joint trajectories, and then computing the required time functions of forces, they propose storing prototypical force time functions in memory. The prototypes are modified on playback according to the demands of each movement. Their controller is a network receiving inputs coding the starting location of the end-effector as well as command inputs giving the Cartesian coordinates of the target end-effector position (Figure 2). The six outputs of the network provide parameters to a torque generator which generates time-varying signals for driving the joint actuators of a dynamic arm model. The time-varying signals are parameterized by six numbers determining characteristics of their wavelike shapes (e.g., giving the magnitudes and relative timing of the half-waves). During each movement, the controller operates in open-loop mode, generating the torque time functions without the aid of sensory feedback. The learning problem for the network, then, is to learn a function associating each pair of end-effector starting and target positions to the values of the six parameters that will accomplish the movement.

A straightforward application of supervised learning is not possible here because the required training examples are not available: It is not known what parameters will work for any pair of starting and target positions (except possibly the trivial cases in which the starting position is already the target position, but these are not useful as training examples). This is an instance of what has been called the *distal error problem* (Jordan and Rumelhart, 1992) for supervised learning. This problem is present whenever the standard of correctness required for supervised learning is available in a coordinate system that is different from the one in which the learning system's activity must be specified for learning. In the case of learning how to move an arm from a starting position to a target position, the standard of correctness is the target position, but what must be learned are the control signals to the joint actuators. The resulting end-effector position error is distal to the output of the controller that is to be learned. Although a nonzero distal error vector indicates that the controller made an error, it does not tell the controller how it should change its behavior to reduce the error.

The distal error problem can be solved by using a model of the controller's influence on the arm's movement (possibly learned through supervised learning) to translate distal error vectors into error vectors required for supervised learning (e.g., Jordan and Rumelhart, 1992). Another approach is to learn an inverse model of the controller's influence on the arm's movement (also discussed by Jordan and Rumelhart, 1992). Reinforcement learning offers yet another way to overcome the distal error problem because learning feedback in the form of error vectors is not required. Continuing with the arm movement example, Lipitkas et al. (1993) defined a reinforcement signal that attains a maximum value of 1 if the arm reaches the desired position and stops there. The signal decreases depending on the distance between the end-effector's final position and the target position and on its tangential velocity as it passes the target position. The reinforcement signal could include other criteria of successful movements as well. With inputs coding starting and target end-effector positions, the network employs SRV units to generate six parameter values using its current weights. The torque generator generates a movement using these parameter values. When the movement is completed, it is scored by the reinforcement signal, and the network's weights are changed according to Gullapalli's SRV learning rule. After a few thousand movements with different starting and target end-effector positions, the system could move with reasonable accuracy for new pairs of starting and target positions as well as for the pairs on which it was trained. This amount of practice is required because the system effectively has to search the six-dimensional parameter space for each starting and target position.

Gaining an understanding of the relative advantages and disadvantages of reinforcement learning and model-based approaches to the distal error problem is a topic of current research. It is clear that reinforcement learning approaches are much simpler, but reinforcement learning can be slower in terms of the amount of experience required for learning. This is true because reinforcement learning methods tend to extract less information from each experience than do the model-based approaches. However, in some problems, reinforcement learning can significantly outperform model-based approaches (e.g., Gullapalli, 1991; Markey and Mozer, 1992). This occurs when it is easier to learn the right actions than it is to model their effects on a complicated process. Hybrid learning architectures using both approaches are promising alternatives.

Other examples have been developed in which reinforcement learning is used to address the distal error problem. Gullapalli, Barto, and Grupen (1994) devised a reinforcement learning

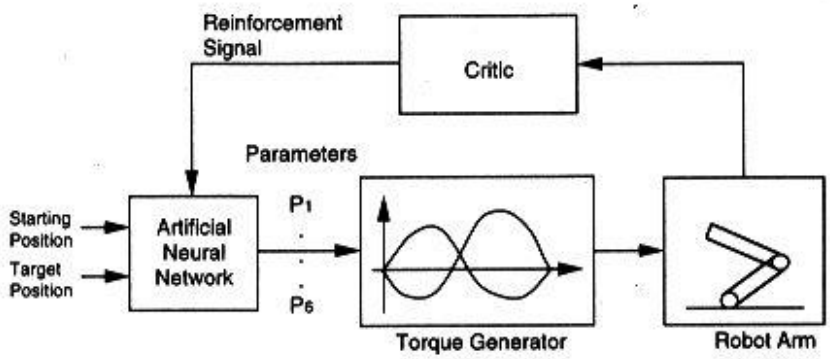

Figure 2. Block diagram of a reinforcement learning controller of an arm (after Figure 1 of Lipitkas et al., 1993). Given inputs coding the starting and target positions of the end effector, the network controller learns to provide correct parameters to a torque generator which generates, in open-loop mode, time-varying torque signals to the arm. The reinforcement signal evaluates the success of each movement after its completion.

network by which a real robot arm learns how to perform a peg-in-hole insertion task. Although this task is important in industrial robotics, it is also suggestive about animal skill acquisition. Unlike the open-loop example of Lipitkas et al. (1993) just described, the peg-in-hole network learns a closed-loop control rule that guides the robot arm using sensory feedback during peg insertion. Learning occurs throughout each peg-insertion attempt, with the reinforcement signal derived from assessments of the progress of the peg toward the desired inserted position and of the forces generated at the robot's wrist. The robot learns to perform insertion tasks even when the clearance between the hole and the peg is many times less than the amount of uncertainty in the robot's sensory feedback.

Another reinforcement learning system was developed by Fagg and Arbib (1992) which models the role of the primate premotor cortex in triggering movements on the basis of visual stimuli. In contrast to the systems just mentioned, this model learns to *select* from previously learned motor behaviors on the basis of sensory cues. The reinforcement signal is +1 when the model selects the correct behavior for a given sensory cue, and −1 otherwise. Although the reinforcement value −1 indicates that an error was made, it does not tell the model which behavior would have been correct or give a clue as to how the model should change its behavior. (The minus sign is not really giving directional information.) This model produces performance curves that are qualitatively similar to those observed in animal experiments.

Credit Assignment

Although reinforcement learning systems do not suffer from the distal error problem, they do suffer from the related *credit assignment problem*. A scalar evaluation of a complex mechanism's behavior does not indicate which of its many action components, both internal and external, were responsible for the evaluation. Thus, it is difficult to determine which of these components deserve the credit (or the blame) for the evaluation. One approach is to assign credit equally to *all* the components so that, through a process of averaging over many variations of the behavior, the components that are key in producing laudable behavior are most strongly reinforced, while inappropriate components are weakened. This approach is commonly taken in reinforcement learning systems, but learning by this process can be very slow in complex systems, such as those involved in motor control. This problem is sometimes referred to as the *structural* credit assignment problem: How is credit assigned to the internal workings of a complex structure? The backpropagation algorithm (e.g., Rumelhart, Hinton, and Williams, 1986) addresses structural credit assignment for artificial neural networks by means of its backpropagation process, and this technique can also be used by reinforcement learning systems (e.g., Gullapalli et al., 1994). However, understanding structural credit assignment mechanisms that are more plausible for biological systems is a frontier of current research.

Reinforcement learning principles lead to a number of alternatives to the backpropagation method for structural assigning credit in complex neural networks. In these methods (see also REINFORCEMENT LEARNING), a single reinforcement signal is uniformly *broadcast* to all the sites of learning, either neurons or individual synapses. Computational studies provide ample evidence that any task that can be learned by error backpropagation can also be learned using this approach, although possibly more slowly. Moreover, these network learning methods are consistent with anatomical and physiological evidence showing the existence of diffusely projecting neural pathways by which neuromodulators (see NEUROMODULATION IN INVERTEBRATE NERVOUS SYSTEMS) can be widely and nonspecifically distributed. It has been suggested that some of these pathways may play a role in reward-mediated learning. A specific hypothesis is that dopamine mediates synaptic enhancement in the corticostriatal pathway in the manner of a broadcast reinforcement signal (Wickens, 1990; see BASAL GANGLIA). Although hypotheses like this are far from being proven, they are much more appealing to many neuroscientists than hypotheses about how error backpropagation might be implemented in the nervous system.

Another aspect of the credit assignment problem occurs when the temporal relationship between a system's behavior and evaluations of that behavior is not as simple as assumed in the previous discussion. How can reinforcement learning work when the learner's behavior is temporally extended and evaluations occur at varying and unpredictable times? Under these more realistic conditions, it is not always clear what events are being evaluated. This has been called the *temporal* credit assignment problem. It is especially relevant in motor control because movements extend over time and evaluative feedback may become available, for example, only after the end of a movement. An approach to this problem that is receiving considerable attention is the use of methods by which the critic itself can learn to provide useful evaluative feedback immediately after the evaluated event. According to this approach, reinforcement learning is not only the process of improving behavior according to given evaluative feedback; it also includes learning how to improve the evaluative feedback itself. These methods have been called *adaptive critic methods* (see REINFORCEMENT LEARNING and Barto, 1992).

Discussion

Motor learning is too complex to view strictly in terms of either supervised learning or reinforcement learning. Feedback used in motor learning ranges from specific standards of correctness to nonspecific evaluative information, and many learning mechanisms with differing characteristics probably interact to produce the motor learning capabilities of animals. However, reinforcement learning principles may be indispensable for motor learning because they seem necessary for improving motor performance beyond the standards of correctness required by supervised learning.

Road Maps: Control Theory and Robotics; Learning in Biological Systems; Biological Motor Control
Background: I.3. Dynamics and Adaptation in Neural Networks
Related Reading: Problem Solving, Connectionist; Sensorimotor Learning

References

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

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

Fagg, A. H., and Arbib, M. A., 1992, A model of primate visual-motor conditional learning, *Adapt. Behav.*, 1:3–37.

Glazer, R., 1971, *The Nature of Reinforcement*, New York: Academic Press.

Gullapalli, V., 1990, A stochastic reinforcement algorithm for learning real-valued functions, *Neural Netw.*, 3:671–692.

Gullapalli, V., 1991, A comparison of supervised and reinforcement learning methods on a reinforcement learning task, in *Proceedings of the 1991 IEEE International Symposium on Intelligent Control*, Los Alamitos, CA: IEEE Computer Society Press, pp. 394–399.

Gullapalli, V., Barto, A. G., and Grupen, R. A., 1994, Learning admittance mappings for force-guided assembly, in *Proceedings of the 1994 International Conference on Robotics and Automation*, Los Alamitos, CA: IEEE Computer Society Press, pp. 2633–2638.

Jordan, M. I., and Rumelhart, D. E., 1992, Supervised learning with a distal teacher, *Cognit. Sci.*, 16:307–354.

Lipitkas, J., D'Eleuterio, G. M. T., Bock, O., and Grodski, J. J., 1993, Reinforcement learning and the parametric motor control hypothesis applied to robotic arm movements, in *Proceedings of the Knowledge-Based Systems and Robotics Workshop*, Gloucester, Ont.: Business Intelligence Systems, pp. 101–106.

Markey, K. L., and Mozer, M. C., 1992, Performance comparison of reinforcement learning algorithms on discrete functions, *Proceedings of the International Joint Conference on Neural Networks*, vol. I, San Diego, CA: IEEE Publishing Services, pp. 853–859.

Rosenbaum, D. A., 1991, *Human Motor Control*, San Diego: Academic Press. ◆

Rumelhart, D. E., Hinton, G. E., and Williams, R. J., 1986, Learning internal representations by error propagation, in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. 1, *Foundations*, (D. E. Rumelhart, J. L. McClelland, and PDP Research Group, Eds.), Cambridge, MA: MIT Press.

Schmidt, R. A., 1982, *Motor Control and Learning*, Champaign, IL: Human Kinetics. ◆

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

Wickens, J., 1990, Striatal dopamine in motor activation and reward-mediated learning: Steps towards a unifying model, *J. Neural Transm.*, 80:9–31.