9 Classical Conditioning and Reinforcement Learning

9.1 Introduction

The ability of animals to learn appropriate actions in response to particular stimuli on the basis of associated rewards or punishments is a focus of behavioral psychology. The field is traditionally separated into classical (or Pavlovian) and instrumental (or operant) conditioning. In classical conditioning, the reinforcers (i.e., the rewards or punishments) are delivered independently of any actions taken by the animal. In instrumental conditioning, the actions of the animal determine what reinforcement is provided. Learning about stimuli or actions solely on the basis of the rewards and punishments associated with them is called reinforcement learning. Reinforcement learning is minimally supervised because animals are not told explicitly what actions to take in particular situations, but must work this out for themselves on the basis of the reinforcement they receive.

classical and instrumental conditioning

reinforcement learning

We begin this chapter with a discussion of aspects of classical conditioning and the models that have been developed to account for them. We first discuss various pairings of one or more stimuli with presentation or denial of a reward, and present a simple learning algorithm that summarizes the results. We then present an algorithm, called temporal difference learning, that leads to predictions of both the presence and the timing of rewards delivered after a delay following stimulus presentation. Two neural systems, the cerebellum and the midbrain dopamine system, have been particularly well studied from the perspective of conditioning. The cerebellum has been studied in association with eyeblink conditioning, a paradigm in which animals learn to shut their eyes just in advance of disturbances, such as puffs of air, that are signaled by cues. The midbrain dopaminergic system has been studied in association with reward learning. We focus on the latter, together with a small fraction of the extensive behavioral data on conditioning.

There are two broad classes of instrumental conditioning tasks. In the first class, which we illustrate with an example of foraging by bees, the reinforcement is delivered immediately after the action is taken. This makes learning relatively easy. In the second class, the reward or punishment depends on an entire sequence of actions and is partly or wholly delayed

delayed rewards until the sequence is completed. Thus, learning the appropriate action at each step in the sequence must be based on future expectation, rather than immediate receipt, of reward. This makes learning more difficult. Despite the differences between classical and instrumental conditioning, we show how to use the temporal difference model we discuss for classical conditioning as the heart of a model of instrumental conditioning when rewards are delayed.

For consistency with the literature on reinforcement learning, throughout this chapter, the letter r is used to represent a reward rather than a firing rate. Also, for convenience, we consider discrete actions such as a choice between two alternatives, rather than a continuous range of actions. We also consider trials that consist of a number of discrete events and use an integer time variable $t = 0, 1, 2, \ldots$ to indicate steps during a trial. We therefore also use discrete weight update rules (like those we discussed for supervised learning in chapter 8) rather than learning rules described by differential equations.

9.2 Classical Conditioning

Classical conditioning involves a wide range of different training and testing procedures and a rich set of behavioral phenomena. The basic procedures and results we discuss are summarized in table 9.1. Rather than going through the entries in the table at this point, we introduce a learning algorithm that serves to summarize and structure these results.

unconditioned stimulus and response

conditioned stimulus and response

In the classic Pavlovian experiment, dogs are repeatedly fed just after a bell is rung. Subsequently, the dogs salivate whenever the bell sounds, as if they expect food to arrive. The food is called the unconditioned stimulus. Dogs naturally salivate when they receive food, and salivation is thus called the unconditioned response. The bell is called the conditioned stimulus because it elicits salivation only under the condition that there has been prior learning. The learned salivary response to the bell is called the conditioned response. We do not use this terminology in the following discussion. Instead, we treat those aspects of the conditioned responses that mark the animal's expectation of the delivery of reward, and build models of how these expectations are learned. We therefore refer to stimuli, rewards, and expectation of reward.

Predicting Reward: The Rescorla-Wagner Rule

The Rescorla-Wagner rule (Rescorla and Wagner, 1972), which is a version of the delta rule of chapter 8, provides a concise account of certain aspects of classical conditioning. The rule is based on a simple linear prediction of the reward associated with a stimulus. We use a binary variable u to represent the presence or absence of the stimulus ($u = 1$ if the stimulus

Paradigm	Pre-Train	Train		Result	
Pavlovian		$s \to r$		$s \to 'r'$	
Extinction	$s \to r$	$s \to \cdot$		$s \to '\cdot'$	
Partial		$s \to r$	$s \to \cdot$	$s \to \alpha\,'r'$	
Blocking	$s_1 \to r$	$s_1+s_2 \to r$		$s_1 \to 'r'$	$s_2 \to '\cdot'$
Inhibitory		$s_1+s_2 \to \cdot$	$s_1 \to r$	$s_1 \to 'r'$	$s_2 \to -'r'$
Overshadow		$s_1+s_2 \to r$		$s_1 \to \alpha_1'r'$	$s_2 \to \alpha_2'r'$
Secondary	$s_1 \to r$	$s_2 \to s_1$		$s_2 \to 'r'$	

Table 9.1 Classical conditioning paradigms. The columns indicate the training procedures and results, with some paradigms requiring a pre-training as well as a training period. Both training and pre-training periods consist of a moderate number of training trials. The arrows represent an association between one or two stimuli (s, or s_1 and s_2) and either a reward (r) or the absence of a reward ($\cdot$). In Partial and Inhibitory conditioning, the two types of training trials that are indicated are alternated. In the Result column, the arrows represent an association between a stimulus and the expectation of a reward ('r') or no reward ('$\cdot$'). The factors of α denote a partial or weakened expectation, and the minus sign indicates the suppression of an expectation of reward.

is present, $u = 0$ if it is absent). The expected reward, denoted by v, is expressed as this stimulus variable multiplied by a weight w,

stimulus u

expected reward v

weight w

$$v = wu \, . \tag{9.1}$$

The value of the weight, w, is established by a learning rule designed to minimize the expected squared error between the actual reward r and the prediction v, $\langle (r - v)^2 \rangle$. The angle brackets indicate an average over the presentations of the stimulus and reward, either or both of which may be stochastic. As we saw in chapter 8, stochastic gradient descent in the form of the delta rule is one way of minimizing this error. This results in the trial-by-trial learning rule known as the Rescorla-Wagner rule,

Rescorla-Wagner rule

$$w \to w + \epsilon \delta u \quad \text{with} \quad \delta = r - v \, . \tag{9.2}$$

Here ϵ is the learning rate, which can be interpreted in psychological terms as the associability of the stimulus with the reward. The crucial term in this learning rule is the prediction error, δ. In a later section, we interpret the activity of dopaminergic cells in the ventral tegmental area (VTA) as encoding a form of this prediction error. If ϵ is sufficiently small and $u = 1$ on every trial (the stimulus is always presented), the rule ultimately makes w fluctuate about the equilibrium value $w = \langle r \rangle$, at which point the average value of δ is 0.

associability

The filled circles in figure 9.1 show how learning progresses according to the Rescorla-Wagner rule during the acquisition and extinction phases of Pavlovian conditioning. In this example, the stimulus and reward were both initially presented on each trial, but later the reward was removed. The weight approaches the asymptotic limit $w = r$ exponentially during the rewarded phase of training (conditioning), and exponentially decays to $w = 0$ during the unrewarded phase (extinction). Experimental learning
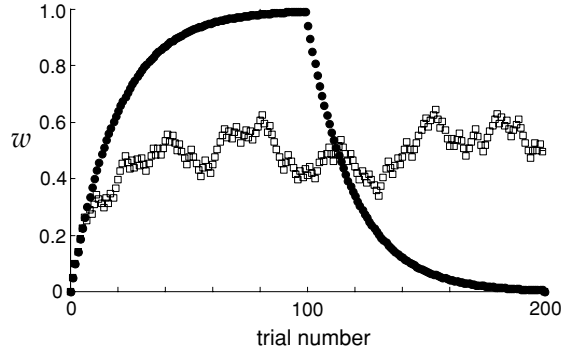
acquisition

extinction

Figure 9.1 Acquisition and extinction curves for Pavlovian conditioning and partial reinforcement as predicted by the Rescorla-Wagner model. The filled circles show the time evolution of the weight w over 200 trials. In the first 100 trials, a reward of $r = 1$ was paired with the stimulus, while in trials 100-200 no reward was paired ($r = 0$). Open squares show the evolution of the weights when a reward of $r = 1$ was paired with the stimulus randomly on 50% of the trials. In both cases, $\epsilon = 0.05$.

curves are generally more sigmoidal in shape. There are various ways to account for this discrepancy, the simplest of which is to assume a nonlinear relationship between the expectation v and the behavior of the animal.

partial reinforcement

The Rescorla-Wagner rule also accounts for aspects of the phenomenon of partial reinforcement, in which a reward is associated with a stimulus only on a random fraction of trials (table 9.1). Behavioral measures of the ultimate association of the reward with the stimulus in these cases indicate that it is weaker than when the reward is always presented. This is expected from the delta rule, because the ultimate steady-state average value of $w = \langle r \rangle$ is smaller than r in this case. The open squares in figure 9.1 show what happens to the weight when the reward is associated with the stimulus 50% of the time. After an initial rise from 0, the weight varies randomly around an average value of 0.5.

stimulus vector **u**
weight vector **w**

To account for experiments in which more than one stimulus is used in association with a reward, the Rescorla-Wagner rule must be extended to include multiple stimuli. This is done by introducing a vector of binary variables **u**, with each of its components representing the presence or absence of a given stimulus, together with a vector of weights **w**. The expected reward is then the sum of each stimulus parameter multiplied by its corresponding weight, written compactly as a dot product,

$$v = \mathbf{w} \cdot \mathbf{u} . \tag{9.3}$$

delta rule

Minimizing the prediction error by stochastic gradient descent in this case gives the delta learning rule,

$$\mathbf{w} \rightarrow \mathbf{w} + \epsilon \delta \mathbf{u} \quad \text{with} \quad \delta = r - v . \tag{9.4}$$

blocking

Various classical conditioning experiments probe the way that predictions are shared between multiple stimuli (see table 9.1). Blocking is the

paradigm that first led to the suggestion of the delta rule in connection with classical conditioning. In blocking, two stimuli are presented together with the reward, but only after an association has already developed for one stimulus by itself. In other words, during the pre-training period, a stimulus is associated with a reward, as in Pavlovian conditioning. Then, during the training period, a second stimulus is present along with the first, in association with the same reward. In this case, the pre-existing association of the first stimulus with the reward blocks an association from forming between the second stimulus and the reward. Thus, after training, a conditioned response is evoked only by the first stimulus, not by the second. This follows from the vector form of the delta rule, because training with the first stimulus makes $w_1 = r$. When the second stimulus is presented along with the first, its weight starts out at $w_2 = 0$, but the prediction of reward $v = w_1 u_1 + w_2 u_2$ is still equal to r. This makes $\delta = 0$, so no further weight modification occurs.

A standard way to induce inhibitory conditioning is to use trials in which one stimulus is shown in conjunction with the reward in alternation with trials in which that stimulus and an additional stimulus are presented in the absence of reward. In this case, the second stimulus becomes a conditioned inhibitor, predicting the absence of reward. This can be demonstrated by presenting a third stimulus that also predicts reward, in conjunction with the inhibitory stimulus, and showing that the net prediction of reward is reduced. It can also be demonstrated by showing that subsequent learning of a positive association between the inhibitory stimulus and reward is slowed. Inhibition emerges naturally from the delta rule. Trials in which the first stimulus is associated with a reward result in a positive value of w_1. Over trials in which both stimuli are presented together, the net prediction $v = w_1 + w_2$ comes to be 0, so w_2 is forced to be negative. *(inhibitory conditioning)*

A further example of the interaction between stimuli is overshadowing. If two stimuli are presented together during training, the prediction of reward is shared between them. After application of the delta rule, $v = w_1 + w_2 = r$. However, the prediction is often shared unequally, as if one stimulus is more salient than the other. Overshadowing can be encompassed by generalizing the delta rule so that the two stimuli have different learning rates (different values of ϵ), reflecting unequal associabilities. Weight modification stops when $\langle \delta \rangle = 0$, at which point the faster growing weight will be larger than the slower growing weight. Various, more subtle effects come from having not only different but also modifiable learning rates, but these lie beyond the scope of our account. *(overshadowing)*

The Rescorla-Wagner rule, binary stimulus parameters, and linear reward prediction are obviously gross simplifications of animal learning behavior. Yet they summarize and unify an impressive amount of classical conditioning data and are useful, provided their shortcomings are fully appreciated. As a reminder of this, we point out one experiment, secondary conditioning, that cannot be encompassed within this scheme.

secondary conditioning

Secondary conditioning involves the association of one stimulus with a reward, followed by an association of a second stimulus with the first stimulus (table 9.1). This causes the second stimulus to evoke expectation of a reward with which it has never been paired (although if pairings of the two stimuli without the reward are repeated too many times, the result is extinction of the association of both stimuli with the reward). The delta rule cannot account for the positive expectation associated with the second stimulus. Indeed, because the reward does not appear when the second stimulus is presented, the delta rule would cause w_2 to become negative. In other words, in this case, the delta rule would predict inhibitory, not excitatory, secondary conditioning. Secondary conditioning is related to the problem of delayed rewards in instrumental conditioning that we discuss later in this chapter.

Secondary conditioning raises the important issue of keeping track of the time within a trial in which stimuli and rewards are present. This is evident because a positive association with the second stimulus is reliably established only if it precedes the first stimulus in the trials in which they are paired. If the two stimuli are presented simultaneously, the result may be inhibitory rather than secondary conditioning.

Predicting Future Reward: Temporal Difference Learning

We measure time within a trial using a discrete time variable t, which falls in the range $0 \leq t \leq T$. The stimulus $u(t)$, the prediction $v(t)$, and the reward $r(t)$ are all expressed as functions of t.

In addition to associating stimuli with rewards and punishments, animals can learn to predict the future time within a trial at which reinforcement will be delivered. We might therefore be tempted to interpret $v(t)$ as the reward predicted to be delivered at time step t. However, Sutton and Barto (1990) suggested an alternative interpretation of $v(t)$ that provides a better match to psychological and neurobiological data, and suggests how animals might use their predictions to optimize behavior when rewards are delayed. The suggestion is that the variable $v(t)$ should be interpreted as a prediction of the total future reward expected from time t onward to the

total future reward   end of the trial, namely

$$\left\langle \sum_{\tau=0}^{T-t} r(t+\tau) \right\rangle . \tag{9.5}$$

The brackets denote an average over trials. This quantity is useful for optimization, because it summarizes the total expected worth of the current state. To approximate $v(t)$, we generalize the linear relationship used for classical conditioning, equation 9.3. For the case of a single time-dependent stimulus $u(t)$, we write

$$v(t) = \sum_{\tau=0}^{t} w(\tau) u(t-\tau) . \tag{9.6}$$

This is just a discrete time version of the sort of linear filter used in chapters 1 and 2.

Arranging for $v(t)$ to predict the total future reward would appear to require a simple modification of the delta rule we have discussed previously,

$$w(\tau) \rightarrow w(\tau) + \epsilon\delta(t)u(t - \tau), \tag{9.7}$$

with $\delta(t)$ being the difference between the actual and predicted total future reward, $\delta(t) = \sum_\tau r(t + \tau) - v(t)$. However, there is a problem with applying this rule in a stochastic gradient descent algorithm. Computation of $\delta(t)$ requires knowledge of the total future reward on a given trial. Although $r(t)$ is known at time t, the succeeding $r(t+1), r(t+2)\ldots$ have yet to be experienced, making it impossible to calculate $\delta(t)$. A possible solution is suggested by the recursive formula

$$\sum_{\tau=0}^{T-t} r(t + \tau) = r(t) + \sum_{\tau=0}^{T-t-1} r(t+1+\tau). \tag{9.8}$$

The temporal difference model of prediction is based on the observation that $v(t+1)$ provides an approximation of the average value (across trials) of the last term in equation 9.8, so we can write

$$\sum_{\tau=0}^{T-t} r(t + \tau) \approx r(t) + v(t+1). \tag{9.9}$$

Replacing the sum in the equation $\delta(t) = \sum_\tau r(t + \tau) - v(t)$ by this approximation gives the temporal difference learning rule,

temporal difference rule

$$w(\tau) \rightarrow w(\tau) + \epsilon\delta(t)u(t-\tau)   \text{with}   \delta(t) = r(t) + v(t+1) - v(t). \tag{9.10}$$

The name of the rule comes from the term $v(t+1) - v(t)$, which is the difference between two successive estimates. $\delta(t)$ is usually called the temporal difference error. Under a variety of circumstances, this rule is likely to converge to make the correct predictions.

Figure 9.2 shows what happens when the temporal difference rule is applied during a training period in which a stimulus appears at time $t = 100$, and a reward is given for a short interval around $t = 200$. Initially, $w(\tau) = 0$ for all τ. Figure 9.2A shows that the temporal difference error starts off being nonzero only at the time of the reward, $t = 200$, and then, over trials, moves backward in time, eventually stabilizing around the time of the stimulus, where it takes the value 2. This is equal to the (integrated) total reward provided over the course of each trial. Figure 9.2B shows the behavior during a trial of a number of variables before and after learning. After learning, the prediction $v(t)$ is 2 from the time the stimulus is first presented ($t = 100$) until the time the reward starts to be delivered. Thus, the temporal difference prediction error (δ) has a spike at $t = 99$. This spike persists, because $u(t) = 0$ for $t < 100$. The temporal difference term (Δv) is negative around $t = 200$, exactly compensating for the delivery of reward, and thus making $\delta = 0$.
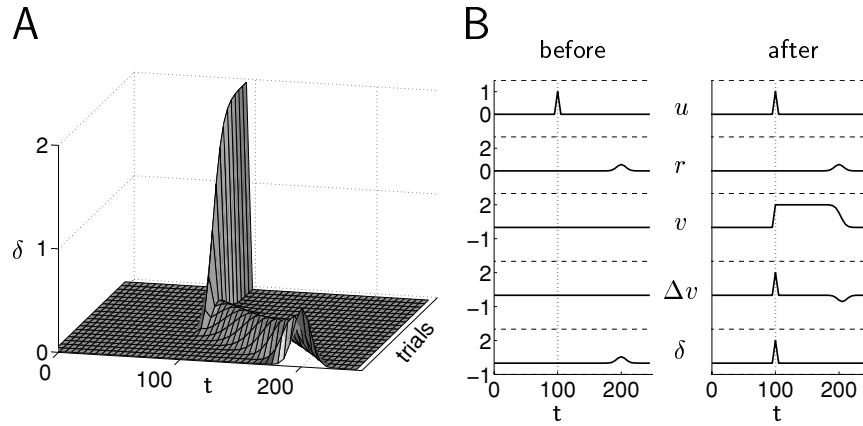
Figure 9.2 Learning to predict a reward. (A) The surface plot shows the prediction error $\delta(t)$ as a function of time within a trial, across trials. In the early trials, the peak error occurs at the time of the reward ($t = 200$), while in later trials it occurs at the time of the stimulus ($t = 100$). (B) The rows show the stimulus $u(t)$, the reward $r(t)$, the prediction $v(t)$, the temporal difference between predictions $\Delta v(t-1) = v(t) - v(t-1)$, and the full temporal difference error $\delta(t-1) = r(t-1) + \Delta v(t-1)$. The reward is presented over a short interval, and the prediction v sums the total reward. The left column shows the behavior before training, and the right column, after training. $\Delta v(t-1)$ and $\delta(t-1)$ are plotted instead of $\Delta v(t)$ and $\delta(t)$ because the latter quantities cannot be computed until time $t+1$, when $v(t+1)$ is available.

As the peak in δ moves backward from the time of the reward to the time of the stimulus, weights $w(\tau)$ for $\tau = 100, 99, \ldots$ successively grow. This gradually extends the prediction of future reward, $v(t)$, from an initial transient at the time of the stimulus to a broad plateau extending from the time of the stimulus to the time of the reward. Eventually, v predicts the correct total future reward from the time of the stimulus onward, and predicts the time of the reward delivery by dropping to 0 at the time when the reward is delivered. The exact shape of the ridge of activity that moves from $t = 200$ to $t = 100$ over the course of trials in figure 9.2A is sensitive to a number of factors, including the learning rate, and the form of the linear filter of equation 9.6.

Unlike the delta rule, the temporal difference rule provides an account of secondary conditioning. Suppose an association between stimulus s_1 and a future reward has been established, as in figure 9.2. When, as indicated in table 9.1, a second stimulus, s_2, is introduced before the first stimulus, the positive spike in $\delta(t)$ at the time that s_1 is presented drives an increase in the value of the weight associated with s_2, and thus establishes a positive association between the second stimulus and the reward. This exactly mirrors the primary learning process for s_1 described above. Of course, because the reward is not presented in these trials, there is a negative spike in $\delta(t)$ at the time of the reward itself, and ultimately the association between both s_1 and s_2 and the reward extinguishes.
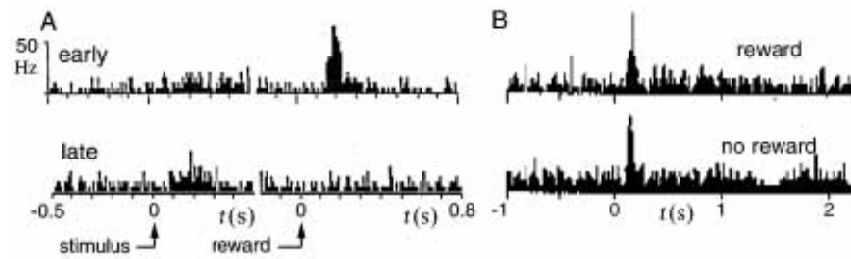
Figure 9.3 Activity of dopaminergic neurons in the VTA for a monkey performing reaction time tasks. (A) Activity of a dopamine cell accumulated over 20 trials showing the spikes time-locked to a stimulus (left panels) or to the reward (right panels) at the times marked 0. The top row is for early trials before the behavior is fully established. The bottom row is for late trials, when the monkey expects the reward on the basis of the stimulus. (B) Activity of a dopamine neuron with and without an expected reward delivery in a similar task. The top row shows the normal behavior of the cell when the reward is delivered. The bottom row shows the result of not delivering an expected reward. The basal firing rate of dopamine cells is rather low, but the inhibition at the time the reward would have been given is evident. (A adapted from Mirenowicz & Schultz, 1994; B adapted from Schultz, 1998.)

Dopamine and Predictions of Reward

The prediction error δ plays an essential role in both the Rescorla-Wagner and temporal difference learning rules, and we might hope to find a neural signal that represents this quantity. One suggestion is that the activity of dopaminergic neurons in the ventral tegmental area (VTA) in the midbrain plays this role.

ventral tegmental area VTA

There is substantial evidence that dopamine is involved in reward learning. Drugs of addiction, such as cocaine and amphetamines, act partly by increasing the longevity of the dopamine that is released onto target structures such as the nucleus accumbens. Other drugs, such as morphine and heroin, also affect the dopamine system. Further, dopamine delivery is important in self-stimulation experiments. Rats will compulsively press levers that cause current to be delivered through electrodes into various areas of their brains. One of the most effective self-stimulation sites is the medial forebrain ascending bundle, which is an axonal pathway. Stimulating this pathway is likely to cause increased delivery of dopamine to the nucleus accumbens and other areas of the brain, because the bundle contains many fibers from dopaminergic cells in the VTA projecting to the nucleus accumbens.

dopamine

In a series of studies by Schultz and his colleagues (Schultz, 1998), monkeys were trained through instrumental conditioning to respond to stimuli such as lights and sounds in order to obtain food and drink rewards. The activities of cells in the VTA were recorded while the monkeys learned these tasks. Figure 9.3A shows the activity of a dopamine cell at two times during learning. The figure is based on a reaction time task in which the monkey keeps a finger resting on a key until a sound is presented. The

monkey then has to release the resting key and press another one to get a fruit juice reward. The reward is delivered a short time after the second key is pressed. The upper plot of figure 9.3A shows the response of a cell in early trials. The cell responds vigorously to the reward, but only fires a little above baseline in response to the sound. The lower plot shows the response after a moderate amount of training. Now the cell responds to the sound, but not to the reward. The responses show a distinct similarity to the plots of $\delta(t)$ in figure 9.2.

The similarity between the responses of the dopaminergic neurons and the quantity $\delta(t)$ suggests that their activity provides a prediction error for reward, i.e. an ongoing difference between the amount of reward that is delivered and the amount that is expected. Figure 9.3B provides further evidence for this interpretation. It shows the activity of a dopamine cell in a task similar to that of figure 9.3A. The top row of this figure shows normal performance, and is just like the bottom row of figure 9.3A. The bottom row shows what happens when the monkey is expecting reward but it is not delivered. In this case, the cell's activity is inhibited below baseline at just the time it would have been activated by the reward in the original trials. This is in agreement with the prediction error interpretation of this activity.

Something similar to the temporal difference learning rule could be realized in a neural system if the dopamine signal representing δ acts to gate and regulate the plasticity associated with learning. We discuss this possibility further in a later section.

9.3   Static Action Choice

policy

In classical conditioning experiments, rewards are directly associated with stimuli. In more natural settings, rewards and punishments are associated with the actions an animal takes. Animals develop policies, or plans of action, that increase reward. In studying how this might be done, we consider two different cases. In static action choice, the reward or punishment immediately follows the action taken. In sequential action choice, rewards may be delayed until several actions are completed.

foraging

As an example of static action choice, we consider bees foraging among flowers in search of nectar. We model an experiment in which single bees forage under controlled conditions among blue and yellow artificial flowers (small dishes of sugar water sitting on colored cards). In actual experiments, the bees learn within a single session (involving visits to 40 artificial flowers) about the reward characteristics of the blue and yellow flowers. All else being equal, they preferentially land on the color of flower that delivers more reward. This preference is maintained over multiple sessions. However, if the reward characteristics of the flowers are interchanged, the bees quickly swap their preferences.

We treat a simplified version of the problem, ignoring the spatial aspects of sampling, and assuming that a model bee is faced with repeated choices between two different flowers. If the bee chooses the blue flower on a trial, it receives a quantity of nectar r_b drawn from a probability density $p[r_b]$. If it chooses the yellow flower, it receives a quantity r_y, drawn from a probability density $p[r_y]$. The task of choosing between the flowers is a form of stochastic two-armed bandit problem (named after slot machines), and is formally equivalent to many instrumental conditioning tasks.

two-armed bandit

The model bee has a stochastic policy, which means that it chooses blue and yellow flowers with probabilities that we write as $P[b]$ and $P[y]$, respectively. A convenient way to parameterize these probabilities is to use the softmax distribution

stochastic policy

softmax

$$P[b] = \frac{\exp(\beta m_b)}{\exp(\beta m_b) + \exp(\beta m_y)} \quad P[y] = \frac{\exp(\beta m_y)}{\exp(\beta m_b) + \exp(\beta m_y)} . \quad (9.11)$$

Here, m_b and m_y are parameters, known as action values, that are adjusted by one of the learning processes described below. Note that $P[b] + P[y] = 1$, corresponding to the fact that the model bee invariably makes one of the two choices. Also note that $P[b] = \sigma(\beta(m_b - m_y))$, where $\sigma(m) = 1/(1 + \exp(-m))$ is the standard sigmoid function, which grows monotonically from 0 to 1 as m varies from $-\infty$ to ∞. $P[y]$ is similarly a sigmoid function of $\beta(m_y - m_b)$. The parameters m_b and m_y determine the frequency at which blue and yellow flowers are visited. Their values must be adjusted during the learning process on the basis of the reward provided.

action values m

The parameter β determines the variability of the bee's actions and exerts a strong influence on exploration. For large β, the probability of an action rises rapidly to 1, or falls rapidly to 0, as the difference between the action values increases or decreases. This makes the bee's action choice almost a deterministic function of the m variables. If β is small, the softmax probability approaches 1 or 0 more slowly, and the bee's actions are more variable and random. Thus, β controls the balance between exploration (small β) and exploitation (large β). The choice of whether to explore to determine if the current policy can be improved, or to exploit the available resources on the basis of the current policy, is known as the exploration-exploitation dilemma. Exploration is clearly critical, because the bee must sample from the two colors of flowers to determine which is better, and keep sampling to make sure that the reward conditions have not changed. But exploration is costly, because the bee has to sample flowers it believes to be less beneficial, to check if this is really the case. Some algorithms adjust β over trials, but we do not consider this possibility.

*exploration-
exploitation
dilemma*

There are only two possible actions in the example we study, but the extension to multiple actions, $a = 1, 2, \ldots, N_a$, is straightforward. In this case, a vector **m** of parameters controls the decision process, and the probability $P[a]$ of choosing action a is

*action value
vector* **m**

$$P[a] = \frac{\exp(\beta m_a)}{\sum_{a'=1}^{N_a} \exp(\beta m_{a'})} . \quad (9.12)$$

We consider two simple methods of solving the bee foraging task. In the first method, called the indirect actor, the bee learns to estimate the expected nectar volumes provided by each flower by using a delta rule. It then bases its action choice on these estimates. In the second method, called the direct actor, the choice of actions is based directly on maximizing the expected average reward.

The Indirect Actor

indirect actor

One course for the bee to follow is to learn the average nectar volumes provided by each type of flower and base its action choice on these. This is called an indirect actor scheme, because the policy is mediated indirectly by the expected volumes. Here, this means setting the action values to

$$m_b = \langle r_b \rangle \quad \text{and} \quad m_y = \langle r_y \rangle \, . \tag{9.13}$$

In our discussion of classical conditioning, we saw that the Rescorla-Wagner or delta rule develops weights that approximate the average value of a reward, just as required for equation 9.13. Thus, if the bee chooses a blue flower on a trial and receives nectar volume r_b, it should update m_b according to the prediction error by

$$m_b \to m_b + \epsilon \delta \quad \text{with} \quad \delta = r_b - m_b \, , \tag{9.14}$$

and leave m_y unchanged. If it lands on a yellow flower, m_y is changed to $m_y + \epsilon \delta$ with $\delta = r_y - m_y$, and m_b is unchanged. If the probability densities of reward $p[r_b]$ and $p[r_y]$ change slowly relative to the learning rate, m_b and m_y will track $\langle r_b \rangle$ and $\langle r_y \rangle$, respectively.

Figure 9.4 shows the performance of the indirect actor on the two-flower foraging task. Figure 9.4A shows the course of weight change due to the delta rule in one example run. Figures 9.4B-D indicate the quality of the action choice by showing cumulative sums of the number of visits to blue and yellow flowers in three different runs. For ideal performance in this task, the dashed line would have slope 1 until trial 100 and 0 thereafter, and the solid line would show the reverse behavior, close to what is seen in figure 9.4C. This reflects the consistent choice of the optimal flower in both halves of the trial. A value of $\beta = 1$ (figure 9.4B) allows for continuous exploration, but at the cost of slow learning. When $\beta = 50$ (figure 9.4C & D), the tendency to exploit sometimes leads to good performance (figure 9.4C), but other times, the associated reluctance to explore causes the policy to perform poorly (figure 9.4D).

Figure 9.5A shows action choices of real bumblebees in a foraging experiment. This experiment was designed to test risk aversion in the bees, so the blue and yellow flowers differed in the reliability rather than the quantity of their nectar delivery. For the first 15 trials (each involving 40 visits to flowers), blue flowers always provided 2 μl of nectar, whereas $\frac{1}{3}$ of the yellow flowers provided 6 μl, and $\frac{2}{3}$ provided nothing (note that the mean
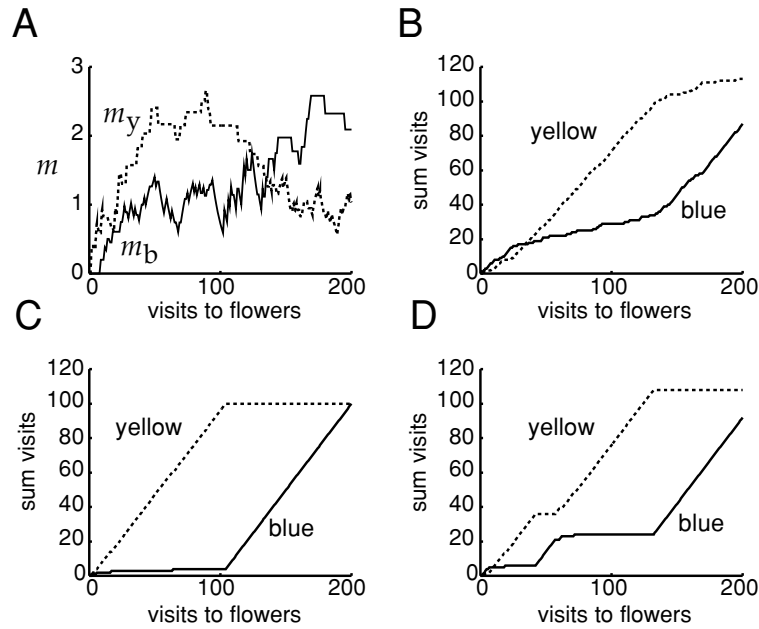
A

B

C

D

Figure 9.4 The indirect actor. Rewards were $\langle r_b \rangle = 1$, $\langle r_y \rangle = 2$ for the first 100 flower visits, and $\langle r_b \rangle = 2$, $\langle r_y \rangle = 1$ for the second 100 flower visits. Nectar was delivered stochastically on half the flowers of each type. (A) Values of m_b (solid) and m_y (dashed) as a function of visits for $\beta = 1$. Because a fixed value of $\epsilon = 0.1$ was used, the weights do not converge perfectly to the corresponding average reward, but they fluctuate around these values. (B-D) Cumulative visits to blue (solid) and yellow (dashed) flowers. (B) When $\beta = 1$, learning is slow, but ultimately the change to the optimal flower color is made reliably. (C, D) When $\beta = 50$, sometimes the bee performs well (C), and other times it performs poorly (D).

reward is the same for the two flower types). Between trials 15 and 16, the delivery characteristics of the flowers were swapped. Figure 9.5A shows the average performance of five bees on this task in terms of their percentage visits to the blue flowers across trials. They exhibit a strong preference for the constant flower type and switch this preference within only a few visits to the flowers when the contingencies change.

To apply the foraging model we have been discussing to the experiment shown in figure 9.5A, we need to model the risk avoidance exhibited by the bees, that is, their reluctance to choose the unreliable flower. One way to do this is to assume that the bees base their policy on the subjective utility function of the nectar volume shown in figure 9.5B, rather than on *subjective utility* the nectar volume itself. Because the function is concave, the mean utility of the unreliable flowers is less than that of the reliable flowers. Figure 9.5C shows that the choices of the model bee match those of the real bees quite well. The model bee is less variable than the actual bees (even more than it appears, because the curve in 9.5A is averaged over five bees), perhaps because the model bees are not sampling from a two-dimensional array of flowers.
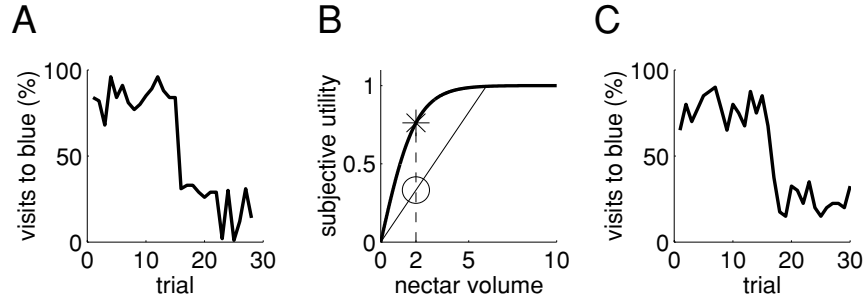
A

B

C

Figure 9.5 Foraging in bumblebees. (A) The mean preference of five real bumblebees for blue flowers over 30 trials involving 40 flower visits. There is a rapid switch of flower preference following the interchange of characteristics after trial 15. (B) Concave subjective utility function mapping nectar volume (in μl) to the subjective utility. The circle shows the average utility of the variable flowers, and the star shows the utility of the constant flowers. (C) The preference of a single model bee on the same task as the bumblebees. Here, $\epsilon = 3/10$ and $\beta = 23/8$. (Data in A from Real, 1991; B and C adapted from Montague et al., 1995.)

The Direct Actor

direct actor

An alternative to basing action choice on average rewards is to choose action values directly to maximize the average expected reward. The expected reward per trial is given in terms of the action probabilities and average rewards per flower by

$$\langle r \rangle = P[\text{b}]\langle r_{\text{b}} \rangle + P[\text{y}]\langle r_{\text{y}} \rangle \,. \tag{9.15}$$

This can be maximized by stochastic gradient ascent. To see how this is done, we take the derivative of $\langle r \rangle$ with respect to m_{b},

$$\frac{\partial \langle r \rangle}{\partial m_{\text{b}}} = \beta \left(P[\text{b}]P[\text{y}]\langle r_{\text{b}} \rangle - P[\text{y}]P[\text{b}]\langle r_{\text{y}} \rangle \right) \,. \tag{9.16}$$

In deriving this result, we have used the fact that, for the softmax distribution of equation 9.11,

$$\frac{\partial P[\text{b}]}{\partial m_{\text{b}}} = \beta P[\text{b}]P[\text{y}] \quad \text{and} \quad \frac{\partial P[\text{y}]}{\partial m_{\text{b}}} = -\beta P[\text{y}]P[\text{b}] \,. \tag{9.17}$$

Using the relation $P[\text{y}] = 1 - P[\text{b}]$, we can rewrite equation 9.16 in a form convenient for later use,

$$\frac{\partial \langle r \rangle}{\partial m_{\text{b}}} = \beta P[\text{b}](1 - P[\text{b}])\langle r_{\text{b}} \rangle - \beta P[\text{y}]P[\text{b}]\langle r_{\text{y}} \rangle \,. \tag{9.18}$$

Furthermore, we can include an arbitrary parameter $\bar{r}$ in both these terms, because it cancels out. Thus,

$$\frac{\partial \langle r \rangle}{\partial m_{\text{b}}} = \beta P[\text{b}](1 - P[\text{b}]) \left(\langle r_{\text{b}} \rangle - \bar{r} \right) - \beta P[\text{y}]P[\text{b}] \left(\langle r_{\text{y}} \rangle - \bar{r} \right) \,. \tag{9.19}$$

A similar expression applies to $\partial\langle r\rangle/\partial m_y$, except that the blue and yellow labels are interchanged.

In stochastic gradient ascent, the changes in the parameter m_b are determined such that, averaged over trials, they end up proportional to $\partial\langle r\rangle/\partial m_b$. We can derive a stochastic gradient ascent rule for m_b from equation 9.19 in two steps. First, we interpret the two terms on the right side as changes associated with the choice of blue and yellow flowers. This accounts for the factors $P[b]$ and $P[y]$, respectively. Second, we note that over trials in which blue is selected, $r_b - \bar{r}$ averages to $\langle r_b\rangle - \bar{r}$, and over trials in which yellow is selected, $r_y - \bar{r}$ averages to $\langle r_y\rangle - \bar{r}$. Thus, if we change m_b according to

$$m_b \to m_b + \epsilon(1 - P[b])(r_b - \bar{r}) \quad \text{if b is selected}$$
$$m_b \to m_b - \epsilon P[b] \ (r_y - \bar{r}) \qquad \text{if y is selected,}$$

the average change in m_b is proportional to $\partial\langle r\rangle/\partial m_b$. Note that m_b is changed even when the bee chooses the yellow flower. We can summarize this learning rule as

$$m_b \to m_b + \epsilon(\delta_{ab} - P[b])(r_a - \bar{r}), \tag{9.20}$$

where a is the action selected (either b or y) and δ_{ab} is the Kronecker delta, $\delta_{ab} = 1$ if $a = b$ and $\delta_{ab} = 0$ if $a = y$. Similarly, the rule for m_y is

$$m_y \to m_y + \epsilon(\delta_{ay} - P[y])(r_a - \bar{r}). \tag{9.21}$$

The learning rule of equations 9.20 and 9.21 performs stochastic gradient ascent on the average reward, whatever the value of $\bar{r}$. Different values of $\bar{r}$ lead to different variances of the stochastic gradient terms, and thus different speeds of learning. A reasonable value for $\bar{r}$ is the mean reward under the specified policy or some estimate of this quantity.

Figure 9.6 shows the consequences of using the direct actor in the same stochastic foraging task as in figure 9.4. Two sample sessions are shown with widely differing levels of performance. Compared to the indirect actor, initial learning is quite slow, and the behavior after the reward characteristics of the flowers are interchanged can be poor. Explicit control of the trade-off between exploration and exploitation is difficult, because the action values can scale up to compensate for different values of β. Despite its comparatively poor performance in this task, the direct actor will be useful later as a model for how action choice can be separated from action evaluation.

The direct actor learning rule can be extended to multiple actions, $a = 1, 2, \ldots, N_a$, by using the multidimensional form of the softmax distribution (equation 9.12). In this case, when action a is taken, $m_{a'}$ for all values of a' is updated according to

$$m_{a'} \to m_{a'} + \epsilon\left(\delta_{aa'} - P[a']\right)(r_a - \bar{r}). \tag{9.22}$$
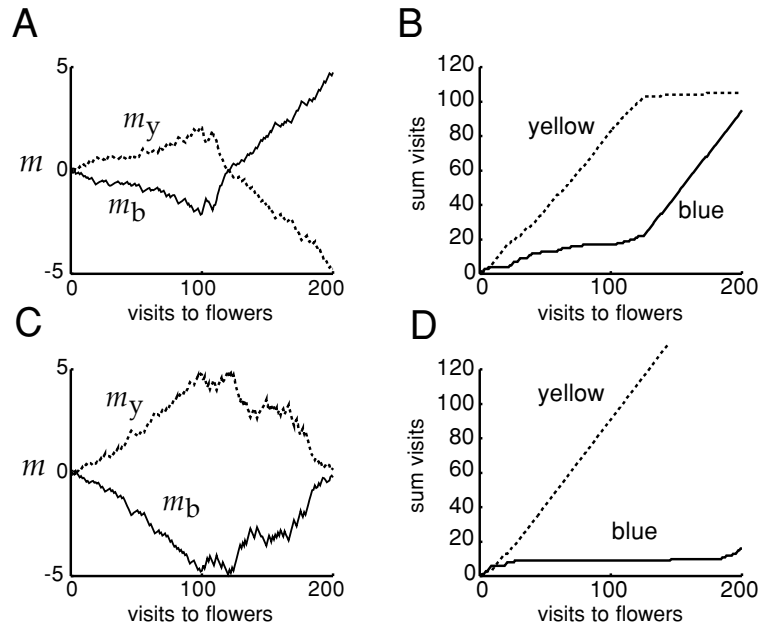
A

B

C

D

Figure 9.6 The direct actor. The statistics of the delivery of reward are the same as in figure 9.4, and $\epsilon = 0.1$, $\bar{r} = 1.5$, $\beta = 1$. The evolution of the weights and cumulative choices of flower type (with yellow dashed and blue solid) are shown for two sample sessions, one with good performance (A and B) and one with poor performance (C and D).

9.4   Sequential Action Choice

In the previous section, we considered ways that animals might learn to choose actions on the basis of immediate information about the consequences of those actions. A significant complication that arises when reward is based on a sequence of actions is illustrated by the maze task shown in figure 9.7. In this example, a hungry rat has to move through a maze, starting from point A, without retracing its steps. When it reaches one of the shaded boxes, it receives the associated number of food pellets and is removed from the maze. The rat then starts again at A. The task is to optimize the total reward, which in this case entails moving left at A and right at B. It is assumed that the animal starts knowing nothing about the structure of the maze or about the rewards.

If the rat started from point B or point C, it could learn to move right or left (respectively), using the methods of the previous section, because it experiences an immediate consequence of its actions in the delivery or nondelivery of food. The difficulty arises because neither action at the actual starting point, A, leads directly to a reward. For example, if the rat goes left at A and also goes left at B, it has to figure out that the former choice was good but the latter was bad. This is a typical problem in tasks that involve delayed rewards. The reward for going left at A is delayed
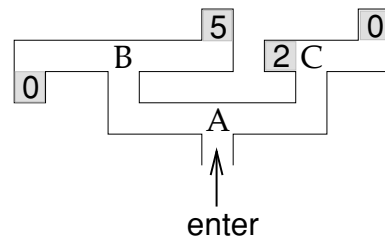
Figure 9.7 The maze task. The rat enters the maze from the bottom and has to move forward. Upon reaching one of the end points (the shaded boxes), it receives the number of food pellets indicated and the trial ends. Decision points are A, B, and C.

until after the rat also goes right at B.

There is an extensive body of theory in engineering, called dynamic programming, as to how systems of any sort can come to select appropriate actions in optimizing control problems similar to (and substantially more complicated than) the maze task. An important method on which we focus is called policy iteration. Our reinforcement learning version of policy iteration maintains and improves a stochastic policy, which determines the actions at each decision point (i.e., left or right turns at A, B, or C) through action values and the softmax distribution of equation 9.12. Policy iteration involves two elements. One, called the critic, uses temporal difference learning to estimate the total future reward that is expected when starting from A, B, or C, when the current policy is followed. The other element, called the actor, maintains and improves the policy. Adjustment of the action values at point A is based on predictions of the expected future rewards associated with points B and C that are provided by the critic. In effect, the rat learns the appropriate action at A, using the same methods of static action choice that allow it to learn the appropriate actions at B and C. However, rather than using an immediate reward as the reinforcement signal, it uses the expectations about future reward that are provided by the critic.

dynamic programming

policy iteration

critic

actor

The Maze Task

As we mentioned when discussing the direct actor, a stochastic policy is a way of assigning a probability distribution over actions (in this case choosing to turn either left or right) to each location (A, B, or C). The location is specified by a variable u that takes the values A, B, or C, and a two-component action value vector $\mathbf{m}(u)$ is associated with each location. The components of the action vector $\mathbf{m}(u)$ control the probability of taking a left or a right turn at u through the softmax distribution of equation 9.12.

The immediate reward provided when action a is taken at location u is written as $r_a(u)$. For the maze of figure 9.7, this takes the values 0, 2, or 5, depending on the values of u and a. The predicted future reward expected

at location u is given by $v(u) = w(u)$. This is an estimate of the total award that the rat expects to receive, on average, if it starts at the point u and follows its current policy through to the end of the maze. The average is taken over the stochastic choices of actions specified by the policy. In this case, the expected reward is simply equal to the weight. The learning procedure consists of two separate steps: policy evaluation, in which $w(u)$ is adjusted to improve the predictions of future reward, and policy improvement, in which $\mathbf{m}(u)$ is adjusted to increase the total reward.

Policy Evaluation

In policy evaluation, the rat keeps its policy fixed (i.e., keeps all the $\mathbf{m}(u)$ fixed) and uses temporal difference learning to determine the expected total future reward starting from each location. Suppose that, initially, the rat has no preference for turning left or right, that is, $\mathbf{m}(u) = 0$ for all u, so the probability of left and right turns is $1/2$ at all locations. By inspection of the possible places the rat can go, we find that the values of the states are

$$v(\text{B}) = \frac{1}{2}(0+5) = 2.5\,, \quad v(\text{C}) = \frac{1}{2}(0+2) = 1\,, \quad \text{and}$$
$$v(\text{A}) = \frac{1}{2}(v(\text{B}) + v(\text{C})) = 1.75\,. \tag{9.23}$$

These values are the average total future rewards that will be received during exploration of the maze when actions are chosen using the random policy. The temporal difference learning rule of equation 9.10 can be used to learn them. If the rat chooses action a at location u and ends up at *critic learning rule*   location u', the temporal difference rule modifies the weight $w(u)$ by

$$w(u) \to w(u) + \epsilon\delta \quad \text{with} \quad \delta = r_a(u) + v(u') - v(u)\,. \tag{9.24}$$

Figure 9.8 shows the result of applying the temporal difference rule to the maze task of figure 9.7. After a fairly short adjustment period, the weights $w(u)$ (and thus the predictions $v(u)$) fluctuate around the correct values for this policy, as given by equation 9.23. The size of the fluctuations could be reduced by making ϵ smaller, but at the expense of increasing the learning time.

In our earlier description of temporal difference learning, we included the possibility that the reward delivery might be stochastic. Here, that stochasticity is the result of a policy that makes use of the information provided by the critic. In the appendix, we discuss a Monte Carlo interpretation of the terms in the temporal difference learning rule that justifies its use.
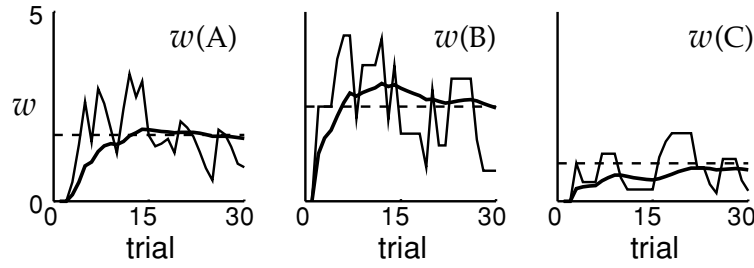
Figure 9.8 Policy evaluation. The thin lines show the course of learning of the weights w(A), w(B), and w(C) over trials through the maze in figure 9.7, using a random unbiased policy ($\mathbf{m}(u) = 0$). Here $\epsilon = 0.5$, so learning is fast but noisy. The dashed lines show the correct weight values from equation 9.23. The thick lines are running averages of the weight values.

Policy Improvement

In policy improvement, the expected total future rewards at the different locations are used as surrogate immediate rewards. Suppose the rat is about to take action a at location u and move to location u'. The expected worth to the rat of that action is the sum of the actual reward received and the rewards that are expected to follow, which is $r_a(u) + v(u')$. For simplicity, we assume that the rat receives the reward for location u at the same time it decides to move on to location u'. The direct actor scheme of equation 9.22 uses the difference $r_a - \bar{r}$ between a sample of the worth of the action (r_a) and a reinforcement comparison term ($\bar{r}$), which might be the average value over all the actions that can be taken. Policy improvement uses $r_a(u) + v(u')$ as the equivalent of the sampled worth of the action, and $v(u)$ as the average value across all actions that can be taken at u. The difference between these is $\delta = r_a(u) + v(u') - v(u)$, which is exactly the same term as in policy evaluation (equation 9.24). The policy improvement or actor learning rule is then   *actor learning rule*

$$m_{a'}(u) \rightarrow m_{a'}(u) + \epsilon \left(\delta_{aa'} - P[a'; u] \right) \delta \qquad (9.25)$$

for all a', where $P[a'; u]$ is the probability of taking action a' at location u given by the softmax distribution of equation 9.12 with action value $m_{a'}(u)$.

To look at this more concretely, consider the temporal difference error starting from location $u = A$, using the true values of the locations given by equation 9.23 (i.e., assuming that policy evaluation is perfect). Depending on the action, δ takes the two values

$$\delta = 0 + v(B) - v(A) = \quad\ \ 0.75 \quad \text{for a left turn}$$
$$\delta = 0 + v(C) - v(A) = -0.75 \quad \text{for a right turn.}$$

The learning rule of equation 9.25 increases the probability that the action with $\delta > 0$ is taken and decreases the probability that the action with $\delta < 0$
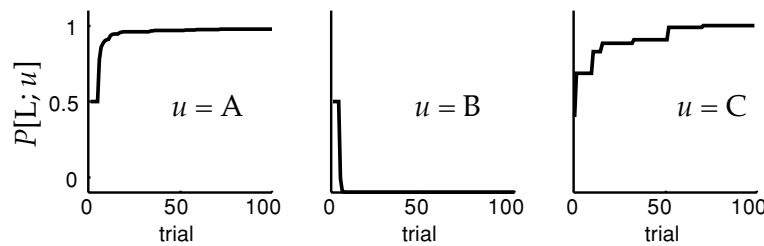
Figure 9.9 Actor-critic learning. The three curves show $P[L; u]$ for the three starting locations u = A, B, and C in the maze of figure 9.7. These rapidly converge to their optimal values, representing left turns at A and C and a right turn at B. Here, $\epsilon = 0.5$ and $\beta = 1$.

is taken. This increases the chance that the rat makes the correct turn (left) at A in the maze of figure 9.7.

As the policy changes, the values, and therefore the temporal difference terms, change as well. However, because the values of all locations can increase only if we choose better actions at those locations, this form of policy improvement inevitably leads to higher values and better actions. This monotonic improvement (or at least not worsening) of the expected future rewards at all locations is proved formally in the dynamic programming *Markov decision* theory of policy iteration for a class of problems called Markov decision *problems* problems (which includes the maze task), as discussed in the appendix.

Strictly speaking, policy evaluation should be complete before a policy is improved. It is also most straightforward to improve the policy completely before it is re-evaluated. A convenient (though not provably correct) alternative is to interleave partial policy evaluation and policy im-*actor-critic* provement steps. This is called the actor-critic algorithm. Figure 9.9 shows *algorithm* the result of applying this algorithm to the maze task. The plots show the development over trials of the probability of choosing to go left, $P[L; u]$, for all three locations. The model rat quickly learns to go left at location A and right at B. Learning at location C is slow because the rat quickly learns that it is not worth going to C at all, so it rarely gets to try the actions there. Thus the algorithm makes an implicit choice of exploration strategy.

Generalizations of Actor-Critic Learning

The full actor-critic model for solving sequential action tasks includes three generalizations of the maze learner that we have presented. The first involves additional information that may be available at the different locations. If, for example, sensory information is available at a location *state vector* **u** u, we associate a state vector $\mathbf{u}(u)$ with that location. The vector $\mathbf{u}(u)$ parameterizes whatever information is available at location u that might help the animal decide which action to take. For example, the state vector might represent a faint scent of food that the rat might detect in the maze task. When a state vector is available, $v(u)$, which is the value at location

u, can depend on it. The simplest dependence is provided by the linear form $v(u) = \mathbf{w} \cdot \mathbf{u}(u)$, similar to the input-output relationship used in linear feedforward network models. The learning rule for the critic (equation 9.24) is then generalized to include the information provided by the state vector,

$$\mathbf{w} \rightarrow \mathbf{w} + \epsilon \delta \mathbf{u}(u), \qquad (9.26)$$

with δ given as in equation 9.24. The maze task we discussed could be formulated in this way by using what is called a unary representation, $\mathbf{u}(A) = (1, 0, 0)$, $\mathbf{u}(B) = (0, 1, 0)$, and $\mathbf{u}(C) = (0, 0, 1)$.

unary representation

We must also modify the actor learning rule to make use of the information provided by the state vector. This is done by generalizing the action value vector $\mathbf{m}$ to a matrix $\mathbf{M}$, called an action matrix. $\mathbf{M}$ has as many columns as there are components of $\mathbf{u}$ and as many rows as there are actions. Given input $\mathbf{u}$, action a is chosen at location u with the softmax probability of equation 9.12, but using component a of the action value vector,

action matrix $\mathbf{M}$

$$\mathbf{m} = \mathbf{M} \cdot \mathbf{u}(u) \quad \text{or} \quad m_a = \sum_b M_{ab} u_b(u). \qquad (9.27)$$

In this case, the learning rule 9.25 must be generalized to specify how to change elements of the action matrix when action a is chosen at location u with state vector $\mathbf{u}(u)$, leading to location u'. A rule similar to equation 9.25 is appropriate, except that the change in $\mathbf{M}$ depends on the state vector $\mathbf{u}$,

three-term covariance rule

$$M_{a'b} \rightarrow M_{a'b} + \epsilon \left(\delta_{aa'} - P[a'; u] \right) \delta u_b(u) \qquad (9.28)$$

for all a', with δ given again as in equation 9.24. This is called a three-term covariance learning rule.

We can speculate about the biophysical significance of the three-term covariance rule by interpreting $\delta_{aa'}$ as the output of cell a' when action a is chosen (which has mean value $P[a'; u]$) and interpreting $\mathbf{u}$ as the input to that cell. Compared with the Hebbian covariance rules studied in chapter 8, learning is gated by a third term, the reinforcement signal δ. It has been suggested that the dorsal striatum, which is part of the basal ganglia, is involved in the selection and sequencing of actions. Terminals of axons projecting from the substantia nigra pars compacta release dopamine onto synapses within the striatum, suggesting that they might play such a gating role. The activity of these dopamine neurons is similar to that of the VTA neurons discussed previously as a possible substrate for δ.

dorsal striatum
basal ganglia

The second generalization is to the case that rewards and punishments received soon after an action are more important than rewards and punishments received later. One natural way to accommodate this is a technique called exponential discounting. In computing the expected future reward, this amounts to multiplying a reward that will be received τ time steps after a given action by a factor γ^τ, where $0 \leq \gamma \leq 1$ is the discounting factor. The smaller γ, the stronger the effect (i.e., the less important temporally

discounting

distant rewards are). Discounting has a major influence on the optimal be-
havior in problems for which there are many steps to a goal. Exponential
discounting can be accommodated within the temporal difference frame-
work by changing the prediction error δ to

$$\delta = r_a(u) + \gamma v(u') - v(u), \tag{9.29}$$

which is then used in the learning rules of equations 9.26 and 9.28.

In computing the amount to change a weight or action value, we defined
the worth of an action as the sum of the immediate reward delivered and
the estimate of the future reward arising from the next state. A final gen-
eralization of actor-critic learning comes from basing the learning rules on
the sum of the next two, three, or more immediate rewards delivered and
basing the estimate of the future reward on more temporally distant times
within a trial. As in discounting, we can use a factor λ to weight how
strongly the expected future rewards from temporally distant points affect
learning. Suppose that $\mathbf{u}(t) = \mathbf{u}(u(t))$ is the state vector used at time step
t of a trial. Such generalized temporal difference learning can be achieved
by computing new state vectors, defined by the recursive relation

$$\tilde{\mathbf{u}}(t) = \tilde{\mathbf{u}}(t-1) + (1-\lambda)(\mathbf{u}(t) - \tilde{\mathbf{u}}(t-1)), \tag{9.30}$$

stimulus traces

TD(λ) rule

and using them instead of the original state vectors $\mathbf{u}$ in equations 9.26
and 9.28. These new state vectors $\tilde{\mathbf{u}}(t)$ are called stimulus traces, and the
resulting learning rule is called the TD(λ) rule. Use of this rule with an
appropriate value of λ can significantly speed up learning.

Learning the Water Maze

As an example of generalized reinforcement learning, we consider the wa-
ter maze task. This is a navigation problem in which rats are placed in a
large pool of milky water and have to swim around until they find a small
platform that is submerged slightly below the surface of the water. The
opaqueness of the water prevents them from seeing the platform directly,
and their natural aversion to water (although they are competent swim-
mers) motivates them to find the platform. After several trials, the rats
learn the location of the platform and swim directly to it when placed in
the water.

Figure 9.10A shows the structure of the model, with the state vector $\mathbf{u}$
providing input to the critic and a collection of eight possible actions for
the actor, which are expressed as compass directions. The components of
$\mathbf{u}$ represent the activity of hippocampal place cells (which are discussed in
chapters 1 and 8). Figure 9.10B shows the activation of one of the input
units as a function of spatial position in the pool. The activity, like that of
a place cell, is spatially restricted.

During training, each trial consists of starting the model rat from a ran-
dom location at the outside of the maze and letting it run until it finds
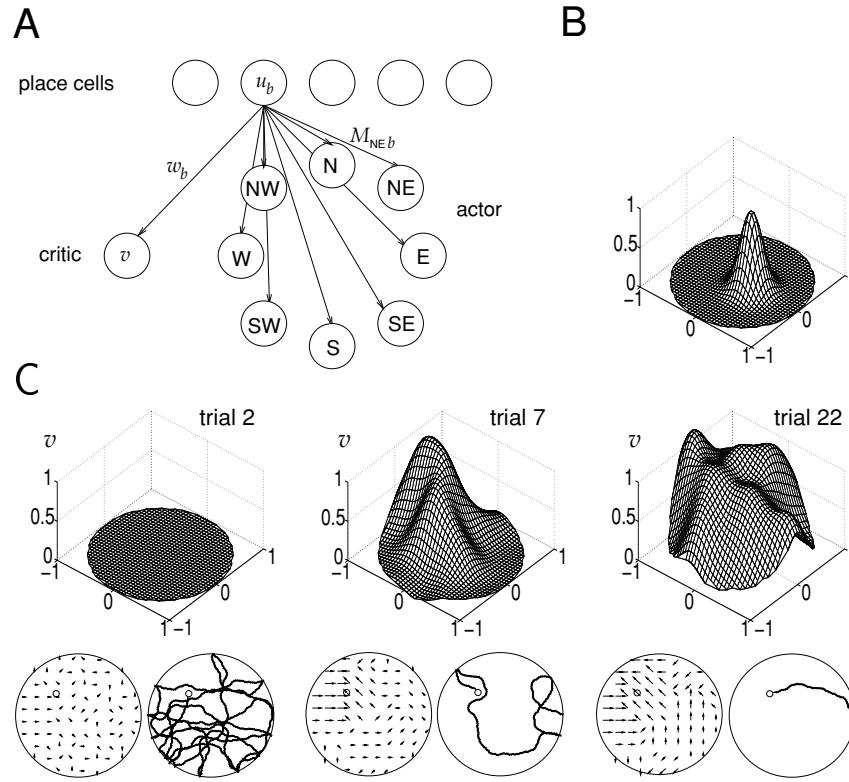
A

B

C

Figure 9.10 Reinforcement learning model of a rat solving a simple water maze task in a 2 m diameter circular pool. (A) There are 493 place cell inputs and 8 actions. The rat moves at 0.3 m/s and reflects off the walls of the maze if it hits them. (B) Gaussian place field for a single input cell with width 0.16 m. The centers of the place fields for different cells are uniformly distributed across the pool. (C) Upper: The development of the value function v as a function of the location in the pool over the first 22 trials, starting from $v = 0$ everywhere . Lower arrow plots: The action with the highest probability for each location in the maze. Lower path plots: Actual paths taken by the model rat from random starting points to the platform, indicated by a small circle. A slight modification of the actor learning rule was used to enforce generalization between spatially similar actions. (Adapted from Foster et al., 2000.)

the platform indicated by a small circle in the lower part of figure 9.10C. At that point a reward of 1 is provided. The reward is discounted with $\gamma = 0.9975$ to model the incentive for the rat to find the goal as quickly as possible. Figure 9.10C indicates the course of learning (trials 2, 7, and 22) of the expected future reward as a function of location (upper figures) and the policy (lower figures with arrows). The lower figures also show sample paths taken by the rat (lower figures with wiggly lines). The final value function (at trial 22) is rather inaccurate, but nevertheless the policy learned is broadly correct, and the paths to the platform are quite short and direct.

Judged by measures such as path length, initial learning proceeds in the
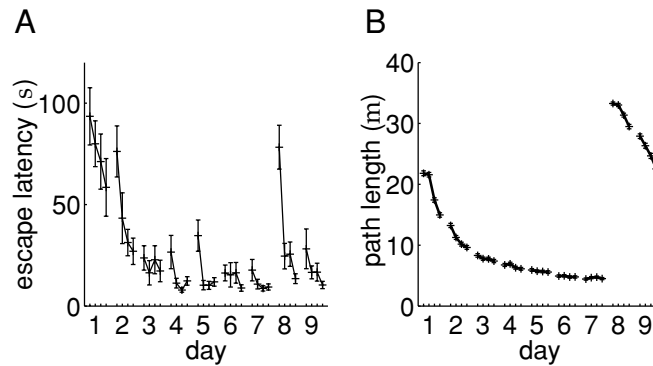
A B

Figure 9.11 Comparison of rats and the model in the water maze task. (A) Average latencies of 12 rats in getting to a fixed platform in the water maze, using four trials per day. On the eighth day, the platform was moved to a new location, which is called reversal. (B) Average path length from 1000 simulations of the model performing the same task. Initial learning matches that of the rats, but performance is worse following reversal. (Adapted from Foster et al., 2000.)

model in a manner comparable to that of actual rats. Figure 9.11A shows the average performance of 12 real rats performing four trials per day in the water maze. The rats swam to a platform at a fixed location, starting from randomly chosen initial locations. The performance of the rats rapidly improves and levels off by about the sixth day. When the platform is moved on the eighth day, in what is called reversal training, the initial latency is long, because the rats search near the old platform position. However, they rapidly learn the new location. Figure 9.11B shows the performance of the model on the same task (though judged by path lengths rather than latencies). Initial learning is equally quick, with near perfect paths by the sixth day. However, performance during reversal training is poor, because the model has trouble forgetting the previous location of the platform. The rats are clearly better at handling this transition. Nevertheless the model shows something of the power of a primitive, but general, learning method.

9.5 Chapter Summary

We discussed reinforcement learning models for classical and instrumental conditioning, interpreting the former in terms of learning predictions about total future rewards and the latter in terms of optimization of those rewards. We introduced the Rescorla-Wagner or delta learning rule for classical conditioning, together with its temporal difference extension, and indirect and direct actor rules for instrumental conditioning given immediate rewards. Finally, we presented the actor-critic version of the dynamic programming technique of policy iteration, evaluating policies using temporal difference learning and improving them using the direct actor learning rule, based on surrogate immediate rewards from the evaluation step.

In the appendix, we show more precisely how temporal difference learning can be seen as a Monte Carlo technique for performing policy iteration.

9.6   Appendix

Markov Decision Problems

Markov decision problems offer a simple formalism for describing tasks such as the maze. A Markov decision problem is comprised of states, actions, transitions, and rewards. The states, labeled by u, are what we called locations in the maze task, and the actions, labeled by a, are the analogs of the choices of which direction to run. In the maze, each action taken at state u led uniquely and deterministically to a new state u'. Markov decision problems generalize this to include the possibility that the transitions from u due to action a may be stochastic, leading to state u' with a transition probability $P[u'|u; a]$. $\sum_{u'} P[u'|u; a] = 1$ for all u and a, because the animal has to end up somewhere. There can be absorbing states (like the shaded boxes in figure 9.7), which are u for which $P[u|u; a] = 1$ for all actions a (i.e., there is no escape for the animal from these locations). Finally, the rewards r can depend both on the state u and the action executed a, and they might be stochastic. We write $\langle r_a(u) \rangle$ for the mean reward in this case. The crucial Markov property is that, given the state at the current time step, the distribution over future states and rewards is independent of the past states. This defines the state sequence as the output of a controlled Markov chain. For convenience, we consider only Markov decision problems that are finite (finite numbers of actions and states) and absorbing (the animal always ends up in one of the absorbing states), and for which the rewards are bounded. We also require that $\langle r_a(u) \rangle = 0$ for all actions a at all absorbing states.

absorbing state

Markov property

The Bellman Equation

The task for a system or animal facing a Markov decision problem, starting in state u at time 0, is to choose a policy, denoted by $\mathbf{M}$, that maximizes the expected total future reward,

$$v^*(u) = \max_{\mathbf{M}} \left\langle \sum_{t=0}^{\infty} r_{a(t)}(u(t)) \right\rangle_{u,\mathbf{M}} , \qquad (9.31)$$

where $u(0) = u$, actions $a(t)$ are determined (either deterministically or stochastically) on the basis of the state $u(t)$ according to policy $\mathbf{M}$, and the notation $\langle \rangle_{u,\mathbf{M}}$ implies taking an expectation over the actions and the states to which they lead, starting at state u and using policy $\mathbf{M}$.

The trouble with the sum in equation 9.31 is that the action $a(0)$ at time 0 affects not only $\langle r_{a(0)}(u(0)) \rangle$, but, by influencing the state of the sys-

tem, also the subsequent rewards. It would seem that the animal would have to consider optimizing whole sequences of actions, the number of which grows exponentially with time. Bellman's (1957) insight was that the Markov property effectively solves this problem. He rewrote equation 9.31 to separate the first and subsequent terms, and used a recursive principle for the latter. The Bellman equation is

$$v^*(u) = \max_a \left\{ \langle r_a(u) \rangle + \sum_{u'} P[u'|u; a] v^*(u') \right\} . \tag{9.32}$$

This says that maximizing reward at u requires choosing the action a that maximizes the sum of the mean immediate reward $\langle r_a(u) \rangle$ and the average of the largest possible values of all the states u' to which a can lead the system, weighted by their probabilities.

Policy Iteration

The actor-critic algorithm is a form of a dynamic programming technique called policy iteration. Policy iteration involves interleaved steps of policy evaluation (updating the critic) and policy improvement (updating the actor). Evaluation of policy **M** requires working out the values for all states u. We call these values $v^M(u)$, to reflect explicitly their dependence on the policy. Each value is analogous to the quantity in 9.5. Using the same argument that led to the Bellman equation, we can derive the recursive formula

$$v^M(u) = \sum_a P_M[a; u] \left\{ \langle r_a(u) \rangle + \sum_{u'} P[u'|u; a] v^M(u') \right\} . \tag{9.33}$$

Equation 9.33, for all states u, is a set of linear equations for $v^M(u)$ that can be solved by matrix inversion. Reinforcement learning has been interpreted as a stochastic Monte Carlo method for performing this operation.

Monte Carlo method

Temporal difference learning uses an approximate Monte Carlo method to evaluate the right side of equation 9.33, and uses the difference between this approximation and the estimate of $v^M(u)$ as the prediction error. The first idea underlying the method is that $r_a(u) + v^M(u')$ is a sample whose mean is exactly the right side of equation 9.33. The second idea is bootstrapping, using the current estimate $v(u')$ in place of $v^M(u')$ in this sample. Thus $r_a(u) + v(u')$ is used as a sampled approximation to $v^M(u)$, and

$$\delta(t) = r_a(u) + v(u') - v(u) \tag{9.34}$$

is used as a sampled approximation to the discrepancy $v^M(u) - v(u)$, which is an appropriate error measure for training $v(u)$ to equal $v^M(u)$. Evaluating and improving policies from such samples without learning $P[u'|u; a]$ and $\langle r_a(u) \rangle$ directly is called an asynchronous, model-free approach to policy evaluation. It is possible to guarantee the convergence of

the estimate v to its true value $v^{\mathbf{M}}$ under a set of conditions discussed in the texts mentioned in the annotated bibliography.

The other half of policy iteration is policy improvement. This normally works by finding an action a^* that maximizes the expression in the curly brackets in equation 9.33 and making the new $P_{\mathbf{M}}[a^*; u] = 1$. One can show that the new policy will be uniformly better than the old policy, making the expected long-term reward at every state no smaller than the old policy, or equally large, if it is already optimal. Further, because the number of different policies is finite, policy iteration is bound to converge.

Performing policy improvement like this requires knowledge of the transition probabilities and mean rewards. Reinforcement learning again uses an asynchronous, model-free approach to policy improvement, using Monte Carlo samples. First, note that any policy $\mathbf{M}'$ that improves the average value

$$\sum_a P_{\mathbf{M}'}[u; a] \left\{ \langle r_a(u) \rangle + \sum_{u'} P[u'|u; a] v^{\mathbf{M}}(u') \right\} \qquad (9.35)$$

for every state u is guaranteed to be a better policy. The idea for a single state u is to treat equation 9.35 rather like equation 9.15, except replacing the average immediate reward $\langle r_a \rangle$ there by an effective average immediate reward $\langle r_a(u) \rangle + \sum_{u'} P[u'|u; a] v^{\mathbf{M}}(u')$ to take long-term as well as current rewards into account. By the same reasoning as above, $r_a(u) + v(u')$ is used as an approximate Monte Carlo sample of the effective immediate reward, and $v(u)$ as the equivalent of the reinforcement comparison term $\bar{r}$. This leads directly to the actor learning rule of equation 9.25.

Note that there is an interaction between the stochasticity in the reinforcement learning versions of policy evaluation and policy improvement. This means that it is not known whether the two together are guaranteed to converge. One could perform temporal difference policy evaluation (which can be proven to converge) until convergence before attempting policy improvement, and this would be sure to work.

9.7 Annotated Bibliography

Dickinson (1980), **Mackintosh (1983)**, and **Shanks (1995)** review animal and human conditioning behavior, including alternatives to Rescorla & Wagner's (1972) rule. **Gallistel (1990)** and Gallistel & Gibbon (2000) discuss aspects of conditioning, in particular to do with timing, that we have omitted.

Our description of the temporal difference model of classical conditioning in this chapter is based on **Sutton (1988)** and **Sutton & Barto (1990)**. The treatment of static action choice comes from **Narendra & Thatachar (1989)** and **Williams (1992)**, and that of action choice with delayed rewards and