

Perhaps the simplest advantage that policy parameterization may have over action-value parameterization is that the policy may be a simpler function to approximate. Problems vary in the complexity of their policies and action-value functions. For some, the action-value function is simpler and thus easier to approximate. For others, the policy is simpler. In the latter case a policy-based method will typically learn faster and yield a superior asymptotic policy (as in Tetris; see Şimşek, Algorta, and Kothiyal, 2016).

Finally, we note that the choice of policy parameterization is sometimes a good way of injecting prior knowledge about the desired form of the policy into the reinforcement learning system. This is often the most important reason for using a policy-based learning method.

*Exercise 13.1* Use your knowledge of the gridworld and its dynamics to determine an *exact* symbolic expression for the optimal probability of selecting the **right** action in Example 13.1.  $\square$

## 13.2 The Policy Gradient Theorem

In addition to the practical advantages of policy parameterization over  $\varepsilon$ -greedy action selection, there is also an important theoretical advantage. With continuous policy parameterization the action probabilities change smoothly as a function of the learned parameter, whereas in  $\varepsilon$ -greedy selection the action probabilities may change dramatically for an arbitrarily small change in the estimated action values, if that change results in a different action having the maximal value. Largely because of this, stronger convergence guarantees are available for policy-gradient methods than for action-value methods. In particular, it is the continuity of the policy dependence on the parameters that enables policy-gradient methods to approximate gradient ascent (13.1).

The episodic and continuing cases define the performance measure,  $J(\boldsymbol{\theta})$ , differently and thus have to be treated separately to some extent. Nevertheless, we will try to present both cases uniformly, and we develop a notation so that the major theoretical results can be described with a single set of equations.

In this section we treat the episodic case, for which we define the performance measure as the value of the start state of the episode. We can simplify the notation without losing any meaningful generality by assuming that every episode starts in some particular (non-random) state  $s_0$ . Then, in the episodic case we define performance as

$$J(\boldsymbol{\theta}) \doteq v_{\pi_{\boldsymbol{\theta}}}(s_0), \quad (13.4)$$

where  $v_{\pi_{\boldsymbol{\theta}}}$  is the true value function for  $\pi_{\boldsymbol{\theta}}$ , the policy determined by  $\boldsymbol{\theta}$ . From here on in our discussion we will assume no discounting ( $\gamma = 1$ ) for the episodic case, although for completeness we do include the possibility of discounting in the boxed algorithms.

With function approximation it may seem challenging to change the policy parameter in a way that ensures improvement. The problem is that performance depends on both the action selections and the distribution of states in which those selections are made, and that both of these are affected by the policy parameter. Given a state, the effect of the policy parameter on the actions, and thus on reward, can be computed in a relatively straightforward way from knowledge of the parameterization. But the effect of the policy

### Proof of the Policy Gradient Theorem (episodic case)

With just elementary calculus and re-arranging of terms, we can prove the policy gradient theorem from first principles. To keep the notation simple, we leave it implicit in all cases that  $\pi$  is a function of  $\theta$ , and all gradients are also implicitly with respect to  $\theta$ . First note that the gradient of the state-value function can be written in terms of the action-value function as

$$\begin{aligned}
 \nabla v_\pi(s) &= \nabla \left[ \sum_a \pi(a|s) q_\pi(s, a) \right], \quad \text{for all } s \in \mathcal{S} && \text{(Exercise 3.18)} \\
 &= \sum_a \left[ \nabla \pi(a|s) q_\pi(s, a) + \pi(a|s) \nabla q_\pi(s, a) \right] && \text{(product rule of calculus)} \\
 &= \sum_a \left[ \nabla \pi(a|s) q_\pi(s, a) + \pi(a|s) \nabla \sum_{s', r} p(s', r | s, a) (r + v_\pi(s')) \right] \\
 &&& \text{(Exercise 3.19 and Equation 3.2)} \\
 &= \sum_a \left[ \nabla \pi(a|s) q_\pi(s, a) + \pi(a|s) \sum_{s'} p(s' | s, a) \nabla v_\pi(s') \right] && \text{(Eq. 3.4)} \\
 &= \sum_a \left[ \nabla \pi(a|s) q_\pi(s, a) + \pi(a|s) \sum_{s'} p(s' | s, a) \right. \\
 &\quad \left. \sum_{a'} [\nabla \pi(a' | s') q_\pi(s', a') + \pi(a' | s') \sum_{s''} p(s'' | s', a') \nabla v_\pi(s'')] \right] && \text{(unrolling)} \\
 &= \sum_{x \in \mathcal{S}} \sum_{k=0}^{\infty} \Pr(s \rightarrow x, k, \pi) \sum_a \nabla \pi(a|x) q_\pi(x, a),
 \end{aligned}$$

after repeated unrolling, where  $\Pr(s \rightarrow x, k, \pi)$  is the probability of transitioning from state  $s$  to state  $x$  in  $k$  steps under policy  $\pi$ . It is then immediate that

$$\begin{aligned}
 \nabla J(\theta) &= \nabla v_\pi(s_0) \\
 &= \sum_s \left( \sum_{k=0}^{\infty} \Pr(s_0 \rightarrow s, k, \pi) \right) \sum_a \nabla \pi(a|s) q_\pi(s, a) \\
 &= \sum_s \eta(s) \sum_a \nabla \pi(a|s) q_\pi(s, a) && \text{(box page 199)} \\
 &= \sum_{s'} \eta(s') \sum_s \frac{\eta(s)}{\sum_{s'} \eta(s')} \sum_a \nabla \pi(a|s) q_\pi(s, a) \\
 &= \sum_{s'} \eta(s') \sum_s \mu(s) \sum_a \nabla \pi(a|s) q_\pi(s, a) && \text{(Eq. 9.3)} \\
 &\propto \sum_s \mu(s) \sum_a \nabla \pi(a|s) q_\pi(s, a) && \text{(Q.E.D.)}
 \end{aligned}$$

on the state distribution is a function of the environment and is typically unknown. How can we estimate the performance gradient with respect to the policy parameter when the gradient depends on the unknown effect of policy changes on the state distribution?

Fortunately, there is an excellent theoretical answer to this challenge in the form of the *policy gradient theorem*, which provides an analytic expression for the gradient of performance with respect to the policy parameter (which is what we need to approximate for gradient ascent (13.1)) that does *not* involve the derivative of the state distribution. The policy gradient theorem for the episodic case establishes that

$$\nabla J(\boldsymbol{\theta}) \propto \sum_s \mu(s) \sum_a q_\pi(s, a) \nabla \pi(a|s, \boldsymbol{\theta}), \quad (13.5)$$

where the gradients are column vectors of partial derivatives with respect to the components of  $\boldsymbol{\theta}$ , and  $\pi$  denotes the policy corresponding to parameter vector  $\boldsymbol{\theta}$ . The symbol  $\propto$  here means “proportional to”. In the episodic case, the constant of proportionality is the average length of an episode, and in the continuing case it is 1, so that the relationship is actually an equality. The distribution  $\mu$  here (as in Chapters 9 and 10) is the on-policy distribution under  $\pi$  (see page 199). The policy gradient theorem is proved for the episodic case in the box on the previous page.

### 13.3 REINFORCE: Monte Carlo Policy Gradient

We are now ready to derive our first policy-gradient learning algorithm. Recall our overall strategy of stochastic gradient ascent (13.1), which requires a way to obtain samples such that the expectation of the sample gradient is proportional to the actual gradient of the performance measure as a function of the parameter. The sample gradients need only be proportional to the gradient because any constant of proportionality can be absorbed into the step size  $\alpha$ , which is otherwise arbitrary. The policy gradient theorem gives an exact expression proportional to the gradient; all that is needed is some way of sampling whose expectation equals or approximates this expression. Notice that the right-hand side of the policy gradient theorem is a sum over states weighted by how often the states occur under the target policy  $\pi$ ; if  $\pi$  is followed, then states will be encountered in these proportions. Thus

$$\begin{aligned} \nabla J(\boldsymbol{\theta}) &\propto \sum_s \mu(s) \sum_a q_\pi(s, a) \nabla \pi(a|s, \boldsymbol{\theta}) \\ &= \mathbb{E}_\pi \left[ \sum_a q_\pi(S_t, a) \nabla \pi(a|S_t, \boldsymbol{\theta}) \right]. \end{aligned} \quad (13.6)$$

We could stop here and instantiate our stochastic gradient-ascent algorithm (13.1) as

$$\boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t + \alpha \sum_a \hat{q}(S_t, a, \mathbf{w}) \nabla \pi(a|S_t, \boldsymbol{\theta}), \quad (13.7)$$

where  $\hat{q}$  is some learned approximation to  $q_\pi$ . This algorithm, which has been called an *all-actions* method because its update involves all of the actions, is promising and