5.6 Off-Policy Monte Carlo Control

We are now ready to present an example of the second class of learning control methods we consider in this book: off-policy methods. Recall that the distinguishing feature of on-policy methods is that they estimate the value of a policy while using it for control. In off-policy methods these two functions are separated. The policy used to generate behavior, called the *behavior* policy, may in fact be unrelated to the policy that is evaluated and improved, called the *estimation* policy. An advantage of this separation is that the estimation policy may be deterministic (e.g., greedy), while the behavior policy can continue to sample all possible actions.

Off-policy Monte Carlo control methods use the technique presented in the preceding section for estimating the value function for one policy while following another. They follow the behavior policy while learning about and improving the estimation policy. This technique requires that the behavior policy have a nonzero probability of selecting all actions that might be selected by the estimation policy. To explore all possibilities, we require that the behavior policy be soft.

Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
$Q(s,a) \leftarrow$ arbitrary
$N(s,a) \leftarrow 0$; Numerator and
$D(s,a) \leftarrow 0$; Denominator of $Q(s,a)$
$\pi \leftarrow$ an arbitrary deterministic policy

Repeat forever:
(a) Select a policy π' and use it to generate an episode:
$s_0, a_0, r_1, s_1, a_1, r_2, \ldots, s_{T-1}, a_{T-1}, r_T, s_T$
(b) $\tau \leftarrow$ latest time at which $a_\tau \neq \pi(s_\tau)$
(c) For each pair s, a appearing in the episode at time τ or later:
$t \leftarrow$ the time of first occurrence of s, a such that $t \geq \tau$
$w \leftarrow \prod_{k=t+1}^{T-1} \frac{1}{\pi'(s_k,a_k)}$
$N(s,a) \leftarrow N(s,a) + wR_t$
$D(s,a) \leftarrow D(s,a) + w$
$Q(s,a) \leftarrow \frac{N(s,a)}{D(s,a)}$
(d) For each $s \in \mathcal{S}$:
$\pi(s) \leftarrow \arg\max_a Q(s,a)$

Figure 5.7: An off-policy Monte Carlo control algorithm.

Figure 5.7 shows an off-policy Monte Carlo method, based on GPI, for computing Q^*. The behavior policy π' is maintained as an arbitrary soft policy. The estimation policy π is the greedy policy with respect to Q, an estimate of Q^π. The behavior policy chosen in (a) can be anything, but in order to assure convergence of π to the optimal policy, an infinite number of returns suitable for use in (c) must be obtained for each pair of state and action. This can be assured by careful choice of the behavior policy. For example, any ε-soft behavior policy will suffice.

A potential problem is that this method learns only from the *tails* of episodes, after the last nongreedy action. If nongreedy actions are frequent, then learning will be slow, particularly for states appearing in the early portions of long episodes. Potentially, this could greatly slow learning. There has been insufficient experience with off-policy Monte Carlo methods to assess how serious this problem is.

Exercise 5.4: Racetrack (programming) Consider driving a race car around a turn like those shown in Figure 5.8. You want to go as fast as possible, but not so fast as to run off the track. In our simplified racetrack, the car is at one of a discrete set of

grid positions, the cells in the diagram. The velocity is also discrete, a number of grid cells moved horizontally and vertically per time step. The actions are increments to the velocity components. Each may be changed by $+1$, -1, or 0 in one step, for a total of nine actions. Both velocity components are restricted to be nonnegative and less than 5, and they cannot both be zero. Each episode begins in one of the randomly selected start states and ends when the car crosses the finish line. The rewards are -1 for each step that stays on the track, and -5 if the agent tries to drive off the track. Actually leaving the track is not allowed, but the position is always advanced by at least one cell along either the horizontal or vertical axes. With these restrictions and considering only right turns, such as shown in the figure, all episodes are guaranteed to terminate, yet the optimal policy is unlikely to be excluded. To make the task more challenging, we assume that on half of the time steps the position is displaced forward or to the right by one additional cell beyond that specified by the velocity. Apply the on-policy Monte Carlo control method to this task to compute the optimal policy from each starting state. Exhibit several trajectories following the optimal policy.

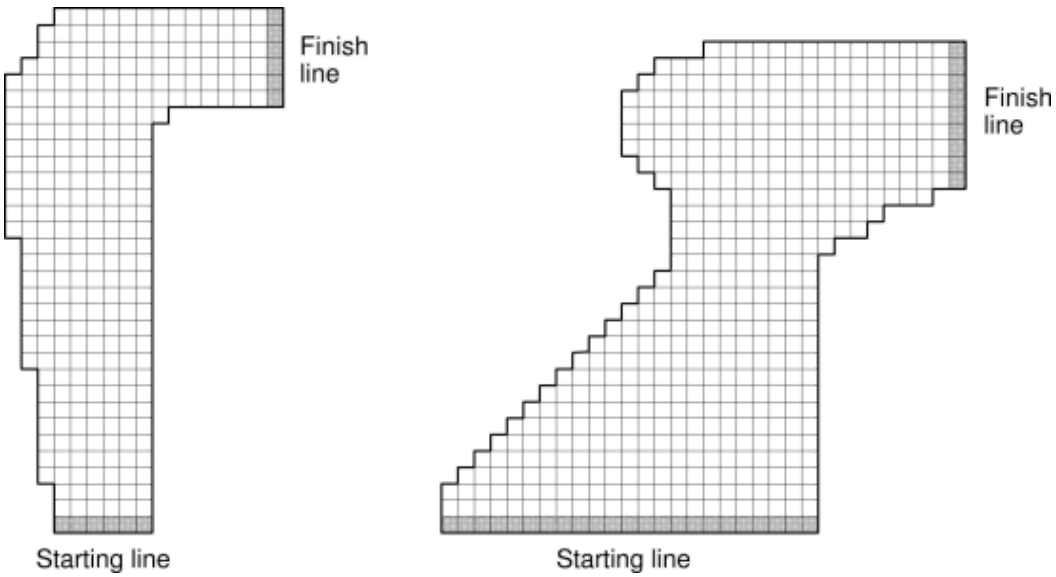

Figure 5.8: A couple of right turns for the racetrack task.

Mark Lee 2005-01-04