C

Markov Decision Problems (MDPs)

We briefly review the main ingredients of Markov Decision Problems or MDPs, which, as we discuss in Chapter 6, can be viewed as single-agent stochastic games. The literature on MDPs is rich, and the reader is referred to the many textbooks on the subject for further reading.

C.1 The model

An MDP is a model for decision making in an uncertain, dynamic world. The (single) agent starts out in some state, takes an action, and receives some immediate rewards. The state then transitions probabilistically to some other state and the process repeats. Formally speaking, an MDP is a tuple (S, A, p, R). S is a set of states and A is a set of actions. The function $p : S \times A \times S \mapsto \mathbb{R}$ specifies the transition probability among states: $p(s, a, s')$ is the probability of ending in state s' when taking action a in state s. Finally, the function $R : S \times A \mapsto \mathbb{R}$ returns the reward for each state-action pair.

The individual rewards are aggregated in one of two ways.

The limit-average reward: $\lim_{T=1}^{\infty} \frac{\sum_{t=1}^{T} R_t}{T}$, where r_t is the reward at time t.

The future-discounted reward: $\sum_{t=1}^{\infty} \beta R_t$, where $0 < \beta < 1$ is the discount factor.

(The reader will notice that both of these definitions must be refined to account for cases in which (in the first case) the limit is ill defined, and in which (in the second case) the sum is infinite. We do not discuss these subtleties here.)

A (stationary, deterministic) policy $\Pi : S \mapsto A$ maps each state to an action.

For concreteness, in the next section we focus on the future-discounted reward case.

C.2 Solving known MDPs via value iteration

optimal policy

Every policy yields a reward under either of the reward-aggregation schemes. A policy that maximizes the total reward is called an *optimal policy*. The primary computational challenge associated with MDPs is to find an optimal policy for a given MDP.

value iteration

It is possible to use linear programming techniques (see Appendix B) to calculate an optimal policy for a known MDP in polynomial time. However, we will focus on an older, dynamic-programming-style method called *value iteration*. We do so for two reasons. First, in typical real-world cases of interest, the LP-formulation of the MDP is too large to solve. Value iteration provides the basis for a number of more practical solutions, such as those providing approximate solutions to very large MDPs. Second, value iteration is relevant to the discussion of learning in MDPs, discussed in Chapter 7.

Value iteration defines a value function $V^\pi : S \mapsto \mathbb{R}$, which specifies the value of following policy π starting in state s. Similarly, we can define a state-action value function $Q^\pi : S \times A \mapsto \mathbb{R}$ as a function that captures the value of starting in state s, taking action a, and then continuing according to policy π. These two functions are related to each other by the following pair of equations.

$$Q^\pi(s,a) = r(s,a) + \beta \sum_{\hat{s}} p(s,a,\hat{s}) V^\pi(\hat{s})$$

$$V^\pi(s) = Q^\pi(s, \pi(s))$$

Bellman equations

For the optimal policy π^*, the second equation becomes $V^{\pi^*}(s) = \max_a Q^{\pi^*}(s,a)$ and the set of equations is referred to as the *Bellman equations*. Note that the optimal policy is easily recovered from the solution to the Bellman equations, specifically from the Q function; the optimal action in state s is $\arg\max_a Q^{\pi^*}(s,a)$.

The Bellman equations are interesting not only because they characterize the optimal policy, but also—indeed, primarily—because they give rise to a procedure for calculating the Q and V values of the optimal policy, and hence the optimal policy itself. Consider the following two assignment versions of the Bellman equations.

$$Q_{t+1}(s,a) \leftarrow r(s,a) + \beta \sum_{\hat{s}} p(s,a,\hat{s}) V_t(\hat{s})$$

$$V_t(s) \leftarrow \max_a Q_t(s,a)$$

Given an MDP, and starting with arbitrary initial Q values, we can repeatedly iterate these two sets of assignment operators ("sets," since each choice of s and a produces a different instance). It is well known that any "fair" order of iteration (by which we mean that each instance of the rules is updated after a finite amount of time) converges on the Q and V values of an optimal policy.