Chapter 2

Markov Decision Processes

For now, we'll assume that the world is known. This involves the state transitions and the reward.

Unknown systems are similar to complex systems. In both, once we don't access the world everywhere, we need to actually *learn* about the world around us.

2.1 Optimality

Theorem 2.1.1: Value Iteration

Initialize:

$$V^0 \sim \|V^0\|_\infty \in [0, 1/1-\gamma]$$

Iterate until convergence:

$$V^{t+1} \leftarrow \mathcal{J}(V^t)$$

Analysis

This algorithm runs in $O(|\mathcal{S}|^3)$ time since we need to perform a matrix inversion.

Theorem 2.1.2: Exact Policy Evaluation

Represent the reward from each state-action pair as a vector

$$R^\pi \in \mathbb{R}^{|\mathcal{S}|} \qquad R^\pi_s = r(s, \pi(s))$$

Also represent the state transitions

$$P^\pi \in \mathbb{R}^{|\mathcal{S}\times\mathcal{S}} \qquad P^\pi_{s,s'} = P(s'|s,\pi(s))$$

That is, row i of P^π is a distribution over the *next state* given that the current state is s_i and we choose an action using policy π.

Using this notation, we can express the Bellman consistency equation as

$$\begin{pmatrix} \vdots \\ V^\pi(s) \\ \vdots \end{pmatrix} = \begin{pmatrix} \vdots \\ r(s,\pi(s)) \\ \vdots \end{pmatrix} + \gamma \begin{pmatrix} \vdots \\ P(s' \mid s,\pi(s)) \\ \vdots \end{pmatrix} \begin{pmatrix} \vdots \\ V^\pi(s') \\ \vdots \end{pmatrix}$$

$$\begin{aligned} V^\pi &= R^\pi + \gamma P^\pi V^\pi \\ (I - \gamma P^\pi)V^\pi &= R^\pi \\ V^\pi &= (I - \gamma P^\pi)R^\pi \end{aligned}$$

if $I - \gamma P^\pi$ is invertible, which we can prove is the case.

Theorem 2.1.3: Iterative Policy Evaluation

How can we calculate the value function V^π of a policy π?

Above, we saw an exact function that runs in $O(|\mathcal{S}|^2)$. But say we really need a fast algorithm, and we're okay with having an approximate answer. Can we do better? Yes!

Using the same notation as above, let's initialize V^0 such that the elements are drawn uniformly from $[0, 1/(1-\gamma)]$.

Then we can iterate the fixed-point equation we found above:

$$V^{t+1} \leftarrow R + \gamma P V^t$$

How can we use this fast approximate algorithm?

Theorem 2.1.4: Policy Iteration

Remember, for now we're only considering policies that are *stationary and deterministic*. There's $|\mathcal{S}|^{|\mathcal{A}}$ of these, so let's start off by choosing one at random. Let's call this initial policy π^0, using the superscript to indicate the time step.

Now for $t = 0, 1, \ldots,$ we perform the following:

1. *Policy Evaluation*: First use the algorithm from earlier to calculate $V^{\pi^t}(s)$ for

all states s. Then use this to calculate the state-action values:

$$Q^{\pi^t}(s,a) = r(s,a) + \gamma \sum_{s'} P(s' \mid s,a) V^{\pi^t}(s')$$

2. *Policy Improvement*: Update the policy so that, at each state, it chooses the action with the highest action-value:

$$\pi^{t+1}(s) = \arg\max_a Q^{\pi^t}(s,a)$$

In other words, we're setting it to act greedily with respect to the new Q-function.

What's the computational complexity of this?

2.2 Finite Horizon MDPs

Suppose we're only able to act for H timesteps.

Now, instead of discounting, all we care about is the (average) total reward that we get over this time.

$$\mathbb{E}[\sum_{t=0}^{H-1} r(s_t, a_t)]$$

To be more precise, we'll consider policies that depend on the time. We'll denote the policy at timestep h as $\pi_h : \mathcal{S} \to \mathcal{A}$. In other words, we're dropping the constraint that policies must be stationary.

This is also called an *episodic model*.

Note that since our policy is nonstationary, we also need to adjust our value function (and Q-function) to account for this. Instead of considering the total infinite-horizon discounted reward like we did earlier, we'll instead consider the *remaining* reward from a given timestep onwards:

$$V_h^\pi(s) = \mathbb{E}\left[\sum_\tau^{H-1} r(s_\tau, a_\tau) \mid s_h = s, a_\tau = \pi_h(s_h)\right]$$
$$Q_h^\pi(s,a) = \mathbb{E}\left[\sum_\tau^{H-1} r(s_\tau, a_\tau) \mid (s_h, a_h) = (s,a)\right]$$

We can also define our Bellman consistency equations, by splitting up the total reward into the immediate reward (at this time step) and the future reward, represented by our state value function from that next time step:

$$Q_h^\pi(s,a) = r(s,a) + \mathop{\mathbb{E}}_{s' \sim P(s,a)}[V_{h+1}^\pi(s')]$$

Theorem 2.2.1: Computing the optimal policy

We can solve for the optimal policy using dynamic programming.

- *Base case.* At the end of the episode (time step $H-1$), we can't take any more actions, so the Q-function is simply the reward that we obtain:

  $$Q_{H-1}^\star(s,a) = r(s,a)$$

  so the best thing to do is just act greedily and get as much reward as we can!

  $$\pi_{H-1}^\star(s) = \arg\max_a Q_{H-1}^\star(s,a)$$

  Then $V_{H-1}^\star(s)$, the optimal value of state s at the end of the trajectory, is simply whatever action gives the most reward.

  $$V_{H-1}^\star = \max_a Q_{H-1}^\star(s,a)$$

- *Recursion.* Then, we can work backwards in time, starting from the end, using our consistency equations!

Note that this is exactly just value iteration and policy iteration combined, since our policy is nonstationary, so we can exactly specify its decisions at each time step!

Analysis

Total computation time $O(H|\mathcal{S}|^2|\mathcal{A}|)$