Lecture 8: Value Functions

1 Introduction

Admin:

- Tue will be a lecture by Jaime Fisac. Does very cool RL + robotics. Will be talking about Q-learning, one of the most important RL algorithms, and also some recent work from his group.
- HW3 released, still due on Mon.

1.1 Recall: Value Functions and the Bellman Equations

The Q-function and the value function predict the future returns, given a state (and action):

$$\max_{\pi} \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right] \tag{1}$$

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid s_0 = s, a_0 = a\right] \tag{2}$$

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid s_0 = s\right]. \tag{3}$$

These functions express global information, about what might happen many many steps into the future. These functions obey the Bellman equations, which means that we can express the value function purely in terms of local information:

$$Q^{\pi}(s,a) = r(s,a) + \gamma \mathbb{E}_{p(s'|s,a)\pi(a'|s')}[Q^{\pi}(s',a')] \qquad \text{(Bellman optimality eq.)}$$

$$Q^{\star}(s,a) = r(s,a) + \gamma \mathbb{E}_{p(s'|s,a)}[\max_{a'} Q^{\star}(s',a')]. \qquad \text{(Bellman expectation eq.)}$$

Today, we'll see how we can use the Bellman equations to learn the value function. Because these identities are true, we can use them to search for the value function.

1.2 Today's learning objectives

At the end of today's class, you will be able to

1. Implement value iteration.
2. Implement policy iteration.
3. Identify the conditions under which the value iteration algorithm will converge to the true value function.
4. Describe properties of the policy iteration algorithm.

Key terms to make sure you understand by the end of today's class:

- Value iteration
- Policy evaluation
- Policy improvement
- policy iteration

2 Warm Up: Dijkstra's Algorithm

Let's say you want to find the shortest path in a graph. Let's say that each edge has a length cost of $c(s, s')$, and that the cost of the goal state is zero ($c(s^*, a = \text{stay}) = 0$). Dijkstra's algorithm maintains a set of values, $V(s)$. These are initialized at infinity, and then updated to be the minimum over the next-state transition:

$$V(s) \leftarrow \infty. \tag{4}$$

$$V(s) \leftarrow \min_{s' \in \mathcal{N}(s)} r(s, s') + V(s'). \tag{5}$$

3 Value Iteration

Value iteration is a method for obtaining $V^\star(s)$. Value iteration will require access to a simulator or model, as it will need to compute an expectation $\mathbb{E}_{p(s'|s,a)}[\cdot]$. Next week we'll build upon value iteration to obtain a new method (Q-learning) which does not need access to a model. The reason for introducing value iteration is that it has very nice theoretical guarantees, and helps to build intuition into how value functions can be learned.

Algorithm 1 Value iteration.

Inputs: model/simulator $p(s' \mid s, a)$
Initialize $V(s) = 0$ for all states s.
while not converged **do**
  for $s \in \mathcal{S}$ **do**
    $V(s) \leftarrow \max_a \left[r(s, a) + \gamma \mathbb{E}_{p(s'|s,a)}[V(s')] \right]$ ▷ Requires a model.
Return $V(s)$.

Exercise 1. How would you modify this procedure to give you an estimate of $V^\pi(s)$?

Exercise 2. While value iteration produces $V^\star(s)$, let's say that you want $Q^\star(s, a)$. How can you use the output from value iteration to obtain $Q^\star(s, a)$? [Hint: use the Bellman equations.]

Exercise 3. Using the result from Exercise 2, explain how you'd use value iteration to produce the optimal policy, $\pi^\star(a \mid s)$.

Exercise 4. What is the time complexity of one iteration of value iteration, expressed in terms of $|\mathcal{A}|$ and $\mathcal{S}$?
A: $\mathcal{O}(|\mathcal{S}|^2|\mathcal{A}|)$: We have a `for` loop over the states ($|\mathcal{S}|$), a `max` over the actions ($|\mathcal{A}|$), and an expectation over the next states ($|\mathcal{S}|$).

3.1 Theory

One way to think about value iteration is that we're iteratively applying function to to a table of values:

$$\mathcal{T}V(s) = \max_a \left[r(s, a) + \gamma \mathbb{E}_{p(s'|s,a)}[V(s')] \right]. \tag{6}$$

This is known as the Bellman optimality operator.[1] Value iteration applies this operation iteratively. Using V_k to denote the result of the k^{th} iteration of value iteration, we have

$$V_{k+1} = \mathcal{T}V_k. \tag{7}$$

There are two important properties of this Bellman optimality operator:

[1] Operator refers to a function that takes a function as input; if your states are discrete, you can think about this as a function that takes a big long vector as input.

Lemma 3.1. *The optimal value function is a fixed point of the Bellman optimality operator:*

$$\mathcal{T}V^\star = V^\star. \tag{8}$$

This result means that, if we arrive at $V^\star$, applying the Bellman optimality operator won't change $V^\star$. You can roughly think about this as a local minima when doing gradient descent.[2]

The second important property is that applying the Bellman optimality operator will move you "closer" towards the optimal value function. This property will use the ℓ_∞ norm, defined as the maximum coordinate-wise difference: $\|V - U\|_\infty = \max_s |V(s) - U(s)|$.

Lemma 3.2. *The Bellman optimality operator is a contraction under the norm ℓ_∞ norm. That is, let V, U be two value functions. Then*

$$\|\mathcal{T}V - \mathcal{T}U\|_\infty \leq \gamma \|V - U\|_\infty. \tag{9}$$

This means that, if you choose two random value functions and started doing value iteration, the two value functions would converge to one another. he rate of convergence is coverned by γ, the discount factor. This means that you'll converge to a single value function, which is $V^\star$. T

Lemma 3.3. *Value iteration converges to $V^\star$.*

$$\lim_{k\to\infty} V_k = V^\star \tag{10}$$

3.2 Bellman expectation operator

Above, we have considered the problem of finding the *optimal* value function, $V^\star$. We can define a similar Bellman operator for estimating V^π, the value function for a particular policy:

$$(\mathcal{T}^\pi V)(s) = \mathbb{E}_{a\sim\pi(a|s), s'\sim p(s'|s,a)}[r(s,a) + \gamma V^\pi(s')]. \tag{11}$$

The same Lemmas as above apply to the Bellman expectation operator:

$$\begin{aligned} \mathcal{T}^\pi V^\pi &= V^\pi && \text{(fixed point)} \\ \|\mathcal{T}^\pi V - \mathcal{T}^\pi U\|_\infty &\leq \gamma \|V - U\|_\infty && \text{(contraction)} \\ \lim_{k\to\infty} V_k &= V^\pi && \text{(convergence)} \end{aligned}$$

4 Policy Iteration

We'll now talk about how we can learn policies using a similar procedure. We'll start by defining **policy evaluation** as any algorithm that takes as input a policy and outputs a value function (or Q function). Value iteration is one example of policy evaluation. We'll look at other algorithms later.

Policy iteration is an algorithm that alternates between estimating the value function for a certain policy (e.g., with value iteration) and improving the policy.

Algorithm 2 Policy iteration.

Randomly initialize policy $\pi_0(a \mid s)$.
while not converged **do**
  $V^{\pi_k} \leftarrow \textsc{PolicyEvaluation}(\pi_k)$
  $\pi_k(a \mid s) = \mathbb{1}\left(a = \arg\max_a \left[r(s,a) + \mathbb{E}_{p(s'|s,a)}[V^{\pi_k}(s')]\right]\right)$ ▷ Requires model
Return π

[2] Value iteration is not exactly gradient descent. [1]

Exercise 5: Rewrite policy iteration in a form that converts V into Q and then extracts the policy from Q.
A:

$$Q^{\pi_k}(s,a) = r(s,a) + \mathbb{E}_{p(s'|s,a)}[V^{\pi_k}(s')] \tag{12}$$

$$\pi_k(a \mid s) = \mathbb{1}\left(a = \arg\max_a Q^{\pi_k}(s,a)\right). \tag{13}$$

A few notes:

- One way of doing the policy evaluation is with value iteration, as discussed above. Note that value iteration is itself an iterative procedure, so there would be a nested `for` loop.
- The second step is known as *policy improvement*.
- Policy iteration is important because almost all algorithms that we will look at in this course will do some form of it: alternating between learning a value function (policy evaluation) and optimizing a policy (policy improvement). Algorithms will differ by how exactly they implement these two steps.
- The algorithm written above finds the greedy policy at each time step. There are alternatives, such as ϵ-greedy and relative entropy methods. These have similar convergence guarantees.

4.1 Theory

The important theoretical result about policy iteration is that it converges to the optimal policy $\pi^\star$. This result is not immediately intuitive because it seems like policy iteration is a greedy procedure: is it possible that optimizing the actions now would decrease the Q values for other states?

Theorem 4.1 (Policy improvement theorem)**.** *Let policy $\pi(a \mid s)$ be given, and one step of policy improvement as*

$$\pi'(a \mid s) = \mathbb{1}(a = \arg\max_a Q^\pi(s,a)) \tag{14}$$

Then $V^{\pi'}(s) \geq V^\pi(s)$ for all states s. The inequality becomes strict if $\pi'(a \mid s)$ differs from $\pi(a \mid s)$ by selecting a higher value action (according to $Q^\pi(a \mid s)$).

Proof.

$$\begin{aligned}
V^\pi(s) &\leq \mathbb{E}_{a\sim\pi'(a|s)}[Q^\pi(s,a)] && (15)\\
&= \mathbb{E}_{a\sim\pi'(a|s),s'\sim p(s'|s,a)}[r(s,a) + \gamma V^\pi(s')] && (16)\\
&\leq \mathbb{E}_{a\sim\pi'(a|s),s'\sim p(s'|s,a)}[r(s,a) + \gamma \mathbb{E}_{a'\sim\pi'(a'|s')}[Q^\pi(s',a')]] && (17)\\
&= \mathbb{E}_{a\sim\pi'(a|s),s'\sim p(s'|s,a)}[r(s,a) + \gamma \mathbb{E}_{a'\sim\pi'(a'|s')}[r(s',a') + \gamma^2 V^\pi(s'')]] && (18)\\
&\leq \cdots && (19)\\
&\leq \mathbb{E}_{\substack{a\sim\pi(a|s),s'\sim p(s'|s,a)\\ a'\sim\pi(a'|s'),s''\sim p(s''|s',a')\\ \cdots}}[r(s,a) + \gamma r(s',a') + \gamma^2 r(s'',a'') + \cdots] = V^{\pi'}(s). && (20)
\end{aligned}$$

□

The policy improvement theorem means that each step of policy improvement results in a policy that gets higher returns (this is what the value function is telling you). This is the key step in proving that policy iteration converges.

4.2 Generalized Policy Improvement

Zooming out, we can think about a generalized version of policy improvement where we alternate between policy evaluation and policy improvement. There are many options for both of these steps, which we'll see later in this course.

There's a nice geometric picture of this [2, Chpt. 4]:

- Alternating between updating V and π.
- GPI geometry

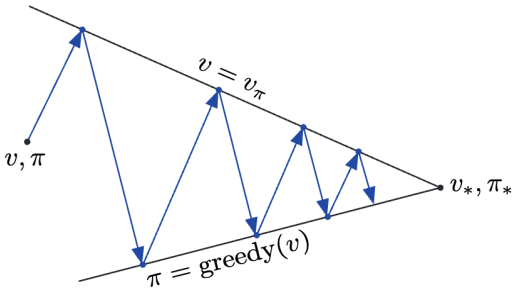

Figure 1: Generalized policy iteration. From [2].

5 Conclusion

Key terms from today:

- Value iteration
- Policy iteration
- Policy evaluation
- Policy improvement

References

[1] Barnard, E. (1993). Temporal-difference methods and markov models. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(2):357–365.

[2] Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.