Algorithms and Uncertainty, Summer 2020 Lecture 13 (5 pages)

Markov Decision Processes with Infinite Time Horizon

Thomas Kesselheim Last Update: June 11, 2020

After having seen many examples of a Markov decision process with a finite time horizon, we will turn today to infinite time horizons. That is, one considers an eternal process but future rewards are less valuable than current ones. Such processes play a very important role in machine learning in the context of reinforcement learning.

1 Model

We again have a Markov decision process, defined by states $\mathcal{S}$, actions $\mathcal{A}$, rewards $r_a(s)$, and state transition probabilities $p_a(s, s')$.

We start from a state $s_0 \in \mathcal{S}$. A policy π is again a function, which defines which action $\pi(s_0, \dots, s_{t-1}) \in \mathcal{A}$ to take in step t when the states so far have been $s_0, \dots, s_{t-1}$. So, again a random sequence of states $s_0^\pi, s_1^\pi, \dots$ and actions $a_0^\pi, a_1^\pi, \dots$ evolves.[1]

Given a discount factor γ, $0 < \gamma < 1$, the expected reward of policy π when starting at s_0 is

$$V(\pi, s_0) = \mathbf{E}\left[\sum_{t=0}^{\infty} \gamma^t r_{a_t^\pi}(s_t^\pi)\right] \ .$$

One motivation for this discounted reward is a less strict time horizon. After each step, we toss a biased coin. If it comes up heads (probability γ), we continue, if it comes up tails (probability $1-\gamma$), we stop right here.

2 Optimal Policies

We can use the same arguments as for finite time horizons to see that the optimal policy only depends on the current state. For such a Markovian policy, we have

$$V(\pi, s) = r_{\pi(s)}(s) + \gamma \sum_{s' \in \mathcal{S}} p_{\pi(s)}(s, s') \cdot V(\pi, s') \ .$$

Naturally, defining $V^*(s) = \max_\pi V(\pi, s)$, we have

$$V^*(s) = \max_{a \in \mathcal{A}} \left(r_a(s) + \gamma \sum_{s' \in \mathcal{S}} p_a(s, s') \cdot V^*(s') \right) \ .$$

This equation is called *Bellman equation*.

Observe that if we know the vector $V^*(s)$, then we could reconstruct the optimal policy. Unfortunately, we don't and unlike in the finite horizon case, there is no simple base of the recursion.

One way to compute an optimal policy is by linear programming: We treat the entries $V^*(s)$ as variables, which have to fulfill the Bellman equations. More precisely, the LP reads

$$\begin{aligned} \text{minimize} \quad & \sum_{s \in \mathcal{S}} V^*(s) \\ \text{subject to} \quad & r_a(s) + \gamma \sum_{s' \in \mathcal{S}} p_a(s, s') \cdot V^*(s') \le V^*(s) \qquad \text{for all } s \in \mathcal{S},\ a \in \mathcal{A} \end{aligned}$$

[1] Note that we start indexing the sequences at 0.

Note that the constraints actually only require that the left-hand side of each Bellman equation is at least as large as the respective right-hand side. The objective function ensures that an optimal solution to this LP fulfills them indeed with equality: If for any s, there is some slack with respect to all a, one can reduce $V^*(s)$ by the smallest slack and improve the solution.

3 Value Iteration

In usual applications, solving the LP is too slow and not necessary. One can find an approximate solution vector much faster using algorithms, which iteratively improve the solution.

Given a vector $(W_s)_{s\in\mathcal{S}}$, let $T(W)$ be the vector defined by

$$(T(W))_s = \max_{a\in\mathcal{A}} \left(r_a(s) + \gamma \sum_{s'\in\mathcal{S}} p_a(s,s') \cdot W_{s'} \right) \ .$$

The vector V^* is a fixed point of the function T, called the *Bellman operator*. In order to find V^*, we therefore repeatedly apply function T, starting from an arbitrary vector. This method is called *value iteration*.

Theorem 13.1. *Value iteration is well-defined, i.e., it converges to the unique fixed point of T.*

For two vectors W, W', define the distance $d(W,W') = \|W - W'\|_\infty$. So, it is the maximum amount that the two vectors differ by in one component.

Lemma 13.2. *For any vectors W and W', we have $d(T(W),T(W')) \leq \gamma d(W,W')$.*

Proof. To this end, consider any component $s \in \mathcal{S}$. We have to show that $|(T(W))_s - (T(W'))_s| \leq \gamma d(W,W')$.

Let $a^* \in \mathcal{A}$ be an action attaining the maximum in the definition of $T(W)_s$. That is, we have

$$T(W)_s = r_{a^*}(s) + \gamma \sum_{s'\in\mathcal{S}} p_{a^*}(s,s') \cdot W_{s'}$$

The action a^* might not be the optimal choice for $T(W')_s$ but it is a feasible one, so

$$T(W')_s \geq r_{a^*}(s) + \gamma \sum_{s'\in\mathcal{S}} p_{a^*}(s,s') \cdot W'_{s'}$$

In combination:

$$T(W)_s - T(W')_s \leq \gamma \sum_{s'\in\mathcal{S}} p_{a^*}(s,s') \cdot (W_{s'} - W'_{s'}) \ .$$

For any $s' \in \mathcal{S}$, we have $W_{s'} - W'_{s'} \leq \max_{s''\in\mathcal{S}} |W_{s''} - W'_{s''}| = d(W,W')$, so

$$T(W)_s - T(W')_s \leq \gamma \sum_{s'\in\mathcal{S}} p_{a^*}(s,s') \cdot d(W,W') = \gamma d(W,W') \ ,$$

because the probabilities sum up to 1.

The same argument holds if we swap the roles of W and W'. Therefore $|(T(W))_s - (T(W'))_s| \leq \gamma d(W,W')$. □

Now, we can continue to the proof of Theorem 13.1.

Proof of Theorem 13.1. Let V^* be the fixed point of T that is induced by the optimal policy. Let V^{**} be any other fixed point. Then, we have $d(V^*,V^{**}) = d(T(V^*),T(V^{**})) \leq \gamma \cdot d(V^*,V^{**})$. As $\gamma \in (0,1)$, this means that $d(V^*,V^{**}) = 0$. So the two fixed points have to be identical.

Furthermore, starting from any $W^{(0)}$, we know that $d(W^{(t)},V^*) \leq \gamma^t d(W^{(0)},V^*)$. As $d(W^{(0)},V^*)$ is finite and independent of t, the sequence has to converge on V^*. □

4 Policy Iteration

An alternative to value iteration is *policy iteration*. We start from an arbitrary policy $\pi^{(0)}$ and improve it iteratively in a sequence $\pi^{(1)}, \pi^{(2)}, \ldots$ until in one iteration the policy does not change.

Given policy $\pi^{(t)}$, we can compute an improved policy as follows. First compute all values $V(\pi^{(t)}, s)$ by solving a system of linear equations. Now set $\pi^{(t+1)}(s)$ to the action a that maximizes $r_a(s) + \gamma \sum_{s' \in \mathcal{S}} p_a(s, s') \cdot V(\pi^{(t)}, s')$. Note what this quantity is actually the expected reward of a different, non-Markovian policy, namely the one that starts from state s by choosing action a and chooses actions according to $\pi^{(t)}$ afterwards.

Theorem 13.3. *Policy iteration converges in finitely many steps to an optimal policy.*

Proof. Note that if $\pi^{(t+1)} = \pi^{(t)}$, then this policy fulfills the Bellman equation. Therefore, any fixed point is an optimal policy.

It remains to prove that the sequence converges. Because there are only finitely many Markovian policies, the only way it could possibly not converge is a cycle. We show that there is no cycle in the iteration by showing that $V(\pi^{(t+1)}, s) \geq V(\pi^{(t)}, s)$ for all t and all $s \in \mathcal{S}$. Note that if $V(\pi^{(t+1)}, s) = V(\pi^{(t)}, s)$ for all s, then we have found a fixed point.

So, let us fix t and show that $V(\pi^{(t+1)}, s) \geq V(\pi^{(t)}, s)$ for all $s \in \mathcal{S}$. To this end, define an auxiliary sequence of policies $\pi'_0, \pi'_1, \ldots$. We define π'_i as the policy that in the first i steps uses $\pi^{(t+1)}$ and then afterwards uses $\pi^{(t)}$. By this definition $V(\pi^{(t)}, s) = V(\pi'_0, s)$ and $V(\pi^{(t+1)}, s) = \lim_{i \to \infty} V(\pi'_i, s)$. It is therefore enough to show that

$$V(\pi'_i, s) \geq V(\pi'_{i-1}, s) \qquad \text{for all } i \in \mathbb{N} \text{ and all } s \in \mathcal{S} \ .$$

We show this claim by induction on i. The base case is $i = 1$. For this case, we have

$$V(\pi'_0, s) = r_{\pi^{(t)}(s)}(s) + \gamma \sum_{s' \in \mathcal{S}} p_{\pi^{(t)}(s)}(s, s') V(\pi^{(t)}, s')$$

and

$$V(\pi'_1, s) = r_{\pi^{(t+1)}(s)}(s) + \gamma \sum_{s' \in \mathcal{S}} p_{\pi^{(t+1)}(s)}(s, s') V(\pi^{(t)}, s') \ ,$$

because policy π'_1 does the first step according to $\pi^{(t+1)}$ and then uses $\pi^{(t)}$. Our definition of policy iteration was exactly that $\pi^{(t+1)}(s)$ maximizes this expression. Therefore, the claim holds.

For $i > 1$, we have

$$V(\pi'_{i-1}, s) = r_{\pi^{(t+1)}(s)}(s) + \gamma \sum_{s' \in \mathcal{S}} p_{\pi^{(t+1)}(s)}(s, s') V(\pi'_{i-2}, s')$$

and

$$V(\pi'_i, s) = r_{\pi^{(t+1)}(s)}(s) + \gamma \sum_{s' \in \mathcal{S}} p_{\pi^{(t+1)}(s)}(s, s') V(\pi'_{i-1}, s') \ .$$

By induction hypothesis, we know that $V(\pi'_{i-2}, s') \leq V(\pi'_{i-1}, s')$ for all $s' \in \mathcal{S}$. So, this immediately implies that $V(\pi'_{i-1}, s) \leq V(\pi'_i, s)$ because every term in the expression for $V(\pi'_i, s)$ is at least as large as the respective term in the expression for $V(\pi'_{i-1}, s)$. □

5 Markovian Multi-Armed Bandits

We have now seen a couple of algorithms to compute an optimal policy. However, like in the case of Markov decision processes with finite time horizon, the state space might be huge, making it infeasible to run these algorithms. Fortunately, often optimal policies are easier to compute if one exploits the structure of the underlying process. We will show this for one particular class of Markov decision processes called *Markovian multi-armed bandits*.

5.1 Single-Armed Bandit

To define them, we first define a *single-armed bandit*. This is a Markov decision process that has only two actions $\mathcal{A} = \{\texttt{play}, \texttt{pause}\}$. The state transitions and rewards for action `play` are arbitrary, but $p_{\texttt{pause}}(s, s) = 1$, $r_{\texttt{pause}}(s) = 0$ for all $s \in \mathcal{S}$. That is, when using action `pause`, the process remains in its state and gives no reward.

Already finding an optimal policy of such a single-armed bandit is a nice exercise.

Example 13.4. *Consider the single-armed bandit in Figure 1.*

For state 2*, it is pretty clear that we should play. Formally, we can show this as follows. If* π *is a policy that plays in state* 2*, then we have* $V(2,\pi) = 10 + \gamma V(2,\pi)$*, and so* $V(2,\pi) = \frac{10}{1-\gamma} \geq 0$*. So, this is better than not playing in state* 2*, regardless of* γ*.*

For state 1*, we can do the same comparison. Let's consider a policy* π *that chooses to play both states. Then we have* $V(1,\pi) = -1 + \gamma\frac{99}{100}V(1,\pi) + \gamma\frac{1}{100}V(2,\pi)$*. This implies* $\left(1 - \frac{99}{100}\gamma\right) V(1,\pi) = -1 + \gamma\frac{1}{100}V(2,\pi) = \frac{\gamma}{10(1-\gamma)} - 1$*. So, as we see, depending on* γ*,* $V(1,\pi)$ *will be positive or not. If it is positive or zero, then it is an optimal policy to play. Otherwise, the optimal policy chooses to stop in state* 1*.*

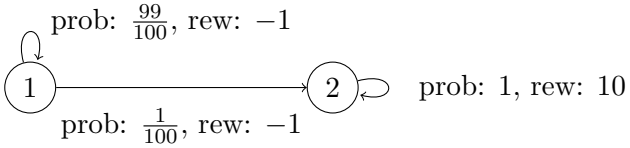

Figure 1: A simple example of one arm. In state 2, it is clearly better to play. But what about the state 1?

5.2 Multi-Armed Bandit

A multi-armed bandit is a parallel composition of such single-armed bandits. We have $\mathcal{S} = \mathcal{S}_1 \times \ldots \times \mathcal{S}_n$, where $\mathcal{S}_i$ is the state space of the i^{th} single-armed bandits. Available actions are $\mathcal{A} = \{\texttt{play}_1, \ldots, \texttt{play}_n, \texttt{pause}\}$, where $\texttt{play}_i$ means that we run the `play` action on the i^{th} single-armed bandit and `pause` on any other. So the different single-armed bandits operate independently but we may only play one arm at a time.

Note that when a Markovian policy (for example an optimal policy) decides to pause, it remains in the state and therefore will keep pausing from now on and not resume playing an arm again. If $\gamma = 1$ then it would be irrelevant in which order we play the arms. However, because $\gamma < 1$, time is the distinguishing factor.

We could always myopically choose the arm with the highest upcoming reward. However, in the example in Figure 2, we would want to play the first arm first without getting any reward. Depending on which state we are in, we would play it again to get some big reward.

Next time, we will show what an optimal policy actually looks like.

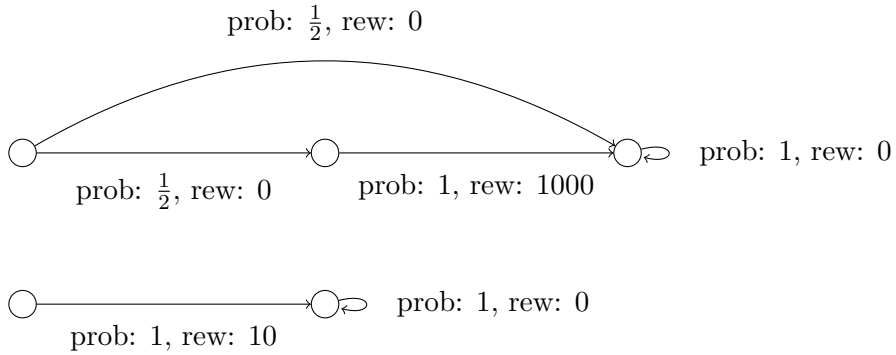

Figure 2: A simple example of two arms. For large values of γ, it is better to play the first arm first. Depending on the outcome, one then continues with the first or the second arm.