Algorithms and Uncertainty, Winter 2018/19 Lecture 13 (4 pages)

# Markov Decision Processes with Infinite Time Horizon

*Instructor: Thomas Kesselheim*

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 $\pi$ is again a function $\pi$, 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 $\gamma$, $0 < \gamma < 1$, the expected reward of policy $\pi$ 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 $\gamma$), 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

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

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

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

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

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

[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 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

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

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 $d(T(W),T(W')) \le \gamma d(W,W')$.*

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

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

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

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

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

In combination:

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

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

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

because the probabilities sum up to 1.

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

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 $d(V^*,V^{**}) = d(T(V^*),T(V^{**})) \le \gamma \cdot d(V^*,V^{**})$. 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 $d(W^{(t)},V^*) \le \gamma^t d(W^{(0)},V^*)$. 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 $r_a + \gamma \sum_{s' \in \mathcal{S}} p_a(s, s') \cdot V(\pi^{(t)}, s')$. One way to think about this is as follows: In the first step, from state $s$, we choose one action $a$. For the following steps, we follow $\pi^{(t)}$.

**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 $s \in \mathcal{S}$.

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 $\pi'_i$ 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 $\pi'_1$ 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 Q-Learning

Both value and policy iteration require us to know the state transition probabilities and rewards. In a typical problem in machine learning, you might not know them but instead try to infer them from what you observe.

For any policy $\pi$, we can define $Q(\pi, s, a) = r_a(s) + \gamma \sum_{s' \in \mathcal{S}} p_a(s, s') V(\pi, s')$ as the *quality* of taking action $a$ in state $s$. Again, for an optimal policy $\pi^*$ write $Q^*(s, a)$ for $Q(\pi^*, s, a)$. An optimal policy fulfills that $V^*(s) = \max_{a \in \mathcal{A}} Q^*(s, a)$, so we get the recursion

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

Even if we do not know the transition probabilities or rewards, we can try to "learn" the solution to this recursion and policy while playing the Markov decision process, only observing the states we move into.

The algorithm maintains a current estimate of $Q^*$, which we denote by $Q^{(1)}, Q^{(2)}, \ldots$. In the $t$-th step, it is in state $s_t$ and chooses the currently best action $a$, which maximizes $Q^{(t)}(s_t, a)$. After having played this action $a_t$, the reward $r_t$ and new state $s_{t+1}$ are observed. Based on this, the $Q$ function is updated by setting $Q^{(t+1)}(s, a) = Q^{(t)}(s, a)$ for $(s, a) \neq (s_t, a_t)$ and

$$Q^{(t+1)}(s_t, a_t) = (1 - \eta) Q^{(t)}(s_t, a_t) + \eta \left( r_t + \gamma \max_a Q^{(t)}(s_{t+1}, a) \right) \ .$$

Here, $\eta \in (0, 1)$ denotes the "learning rate". It has to be chosen appropriately so that the current observation is not neither overestimated nor underestimated.

Using more sophisticated tools, one can show that this *Q-Learning* also converges to $Q^*$. (Note that this requires the analysis of a random process.) We will see simpler models in which we will use very similar approaches.