Lecture 14: Markov Decision Process

Lecturer: *Sahil Singla* Last updated: March 4, 2022

Let's recall the Prophet Inequality problem: we are given distributions $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_n$ of n independent non-negative random variables; the outcome values $X_t \sim \mathcal{D}_t$ are revealed one-by-one and we need to immediately accept/reject this value; the goal is to maximize the value of the first accepted value $\mathbb{E}[X_\tau]$. In the last lecture we used the Competitive Ratio framework from Online Algorithms to define the benchmark of best hindsight optimum. Although interesting, a criticism of this framework for online problems is that the hindsight optimum benchmark is often unachievable even with unlimited computational power, so why do we even compare to it?

We already saw an alternative to hindsight optimum in the Regret minimization framework. The idea here is to restrict the class of algorithms and then define our benchmark to be the best algorithm in this class. Both competitive ratio and regret minimization are satisfactory models when the input is adversarial, however, for stochastic inputs there is another natural (and perhaps better) benchmark, that of the best "policy".

Today's goal will be to introduce a general model for stochastic decision problems such as Prophet Inequalities, to describe optimal policies, and to give algorithms to compute them.

1 Markov Decision Processes

A Markov Decision Process (MDP) is defined by a set of states $\mathcal{S}$, a set of actions $\mathcal{A}$, a reward function that defines a reward $r_a(s)$ for taking action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$ and a random transition function, which is defined by probabilities $p_a(s, s')$: If we are in state $s \in \mathcal{S}$ and we take action $a \in \mathcal{A}$, then we move on to state $s' \in \mathcal{S}$ with probability $p_a(s, s') \in [0, 1]$.

The process works as follows. We start from state $s_1 \in \mathcal{S}$, choose one action $a \in \mathcal{A}$. We immediately get reward $r_a(s) \in \mathbb{R}$ and then continue to a random state s', which is given by the probability distribution $p_a(s_1, \cdot)$. This way, a sequence $s_1, s_2, \ldots$ evolves. We move from s_t to s_{t+1} by the probability distribution $p_a(s_t, \cdot)$. So, the probabilities only depend on the current state and the current action but not on which states we have seen before. This makes the process *Markovian*.

Generally, rewards may also be random. To capture this, set $r_a(s)$ to the *expected* reward that you get when taking action a in state s.

On the one hand, this generalizes a deterministic finite automaton. Here, for each a and s, there is exactly one s' for which $p_a(s, s') = 1$ and $p_a(s, s') = 0$ otherwise. On the other hand, it is also a generalization of a Markov chain. Here, $\mathcal{A}$ has only one element (an action like "continue") and then we move through states without having a real choice.

Example 1. *Let us define the Markov decision process for the motivating prophet inequality example. For simplicity, let's assume weighted Bernoulli distributions, i.e., $X_t = v_t$ w.p.*

p_t and 0 otherwise. In the states, we have to keep track of which random variables were observed so far, their outcomes, and whether we accepted any one of them. We can now define the state-space $\mathcal{S} = \{2^{[n]} \times [n]\} \cup \{\text{STOP}\}$, where $[n] = \{1, \dots, n\}$.

Each action corresponds to either accepting the t-th random variable or rejecting it. Therefore, $\mathcal{A} = \{Accept, Reject\}$. Let us define the state transitions. For $(s,t) \in 2^{[n]} \times [n]$, we set $p_{Accept}((s,t), \text{STOP}) = 1$ to mean that we stop whenever the t-th step is accepted. We set $p_{Reject}((s,t), (s \cup \{t+1\}, t+1) = p_{t+1}$ to mean that we reject t-th element and the $(t+1)$-th element has value v_{t+1} and we set $p_{Reject}((s,t), (s, t+1) = 1 - p_{t+1}$ to mean the $(t+1)$-th element has value 0. Furthermore, $p_a(\text{STOP}, \text{STOP}) = 1$ to ensure that we remain in state STOP *once an envelope was empty. All other probabilities are set to 0.*

The reward for rejecting an element is 0, i.e. $r_{Reject}((s,t)) = 0$, and the reward for accepting the t-th element is X_t, i.e., $r_{Accept}((s,t)) = X_t$ which depends on the last coordinate of s and either equals v_t or 0. (Although any reasonable algorithm will never Accept when $X_t = 0$ but it's still a valid move.) Furthermore, $r_a(\text{STOP}) = 0$.

2 Finite Time MDPs: Policies and Their Structure

A *policy* π assigns to each sequences of states $s_1, \dots, s_{t-1} \in \mathcal{S}$ an action $\pi(s_1, \dots, s_{t-1}) \in \mathcal{A}$. So, if we run policy π starting from s_1, we pass through a random sequence of states $s_1^\pi, s_2^\pi, \dots$, using a random sequence of actions $a_1^\pi, a_2^\pi, \dots$.

Generally, we can move through a Markov decision process for unbounded time. We will first focus on the case of a finite time horizon and discuss infinite time MDPs in Section 3. That is, there is some T such that we do not care what happens after time T. In this case, we can write the expected reward of policy π when starting at s_1 as

$$V(\pi, s_1, T) = \mathbb{E}\Big[\sum_{t=1}^{T} r_{a_t^\pi}(s_t^\pi)\Big] \ .$$

We also define $V^*(s_1, T)$ as the highest expected reward that one can achieve starting from s_1 in T steps, that is, $V^*(s_1, T) = \max_{\text{policy } \pi} V(\pi, s_1, T)$. (Note that there are only finitely many histories $s_1, \dots, s_{t-1}$ for $t \le T$ and therefore only finitely many different policies, so the maximum is well-defined.)

Consider an optimal policy π, that is $V(\pi, s_1, T) = V^*(s_1, T)$. As a_1^π is deterministic, we might as well write

$$V(\pi, s_1, T) = r_{a_1^\pi}(s_1) + \mathbf{E}\left[\sum_{t=2}^{T} r_{a_t^\pi}(s_t^\pi)\right] = r_{a_1^\pi}(s_1) + \sum_{s' \in \mathcal{S}} p_{a_1^\pi}(s_1, s') \mathbf{E}\left[\sum_{t=2}^{T} r_{a_t^\pi}(s_t^\pi) \mid s_2^\pi = s'\right] \ .$$

Let us inspect the expectation on the right-hand side. We claim that

$$\mathbf{E}\left[\sum_{t=2}^{T} r_{a_t^\pi}(s_t^\pi) \mid s_2^\pi = s'\right] = V^*(s', T-1) \ .$$

The reason is simple: Both is the maximum expected reward that we would receive from a Markov decision process running for $T-1$ steps, starting from s'. On the left-hand side, we actually start from s_1 but this does not make a difference for the remaining steps. Importantly, rewards in the current step only depend on the current state and action, not on the past ones.

We skip the fleshed out formal argument here. One possible way is to assume that either side is strictly larger than the other and observe that one could either add or remove s_1 from the beginning of the history.

Consequently, we can define $V^*(s,T)$ for $T \geq 1$ recursively as

$$V^*(s,T) = \max_{a \in \mathcal{A}} \left(r_a(s) + \sum_{s' \in \mathcal{S}} p_a(s,s') V^*(s', T-1) \right) \tag{1}$$

and $V^*(s,0) = 0$. These observations now lead us to the following theorem.

Theorem 1. *An optimal policy for a time horizon of T steps can be computed in time $O(T \cdot |\mathcal{S}|^2 \cdot |\mathcal{A}|)$. Moreover the computed policy is Markovian.*

Besides stating that an optimal policy can indeed be computed, this theorem gives us also an insight into the structure of optimal policies.

Definition 1. *A policy π is* Markovian *if actions only depend on the current state and the number of remaining steps. For a Markovian policy, we write $\pi(s,T')$ for the action being taken when there are T' steps remaining.*

Proof. We can compute an optimal policy by dynamic programming using the following algorithm.

- Initialize $V^*(s,0) = 0$ for all $s \in \mathcal{S}$
- For $T' = 1, \ldots, T$
  - For all $s \in \mathcal{S}$
    - Set $V^*(s,T') = \max_{a \in \mathcal{A}} \left(r_a(s) + \sum_{s' \in \mathcal{S}} p_a(s,s') V^*(s', T'-1) \right)$
    - Let $\pi(s,T') = \arg\max_{a \in \mathcal{A}} \left(r_a(s) + \sum_{s' \in \mathcal{S}} p_a(s,s') V^*(s', T'-1) \right)$

The policy π we compute is Markovian because $\pi(s,T')$ only depends on s and T'. It is optimal because by construction $V(\pi, s, T') = V^*(s,T')$ for all s and T'.

Regarding the running time, we observe that we compute $T \cdot |\mathcal{S}|$ values of V^* in total, each computation takes $|\mathcal{S}| \cdot |\mathcal{A}|$ steps. □

Deriving the Optimal Prophet Inequality Policy. As we have just seen, the optimal policy for our Prophet Inequality problem (Example 1) can be computed via dynamic programming. The downside is that the number of states is exponential in n, which is huge. Can we find the optimal PI policy in polynomial time?

Indeed, this is possible by formulating a different MDP for the problem which only has a polynomial size. The main observation is that we don't need to store the values of the rejected random variables in a state, since they do not influence whether the current X_t should be rejected/accepted. Thus, we can define the state-space as $\mathcal{S} = \bigcup_{t\in[n]} \Big(\{(v_t, t)\} \cup \{(0, t)\}\Big) \cup \{\text{STOP}\}$ where (v, t) denotes that $X_t = v$.

The action space is still $\mathcal{A} = \{\text{Accept, Reject}\}$. Let us define the state transitions. For (v, t), we set $p_{\text{Accept}}((v, t), \text{STOP}) = 1$ and $r_{\text{Accept}}((v, t)) = v$ to mean that we stop whenever the t-th element is accepted and get reward v. We set $p_{\text{Reject}}((v, t), (v_{t+1}, t+1)) = p_{t+1}$ and $p_{\text{Reject}}((v, t), (0, t+1)) = 1 - p_{t+1}$ to denote transition to the random value of X_{t+1}. The reward $r_{\text{Reject}}((v, t)) = 0$ since we don't get any value on rejecting the t-th element. Finally, $p_a(\text{STOP}, \text{STOP}) = 1$ and $r_a(\text{STOP}) = 0$ since we cannot escape the STOP) state and don't get any reward there.

3 Infinite Time MDPs

After having seen many examples of a Markov decision process with a finite time horizon, we will turn 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, e.g., see this book [1].

3.1 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 $\pi(s_0, \ldots, s_{t-1}) \in \mathcal{A}$ to take in step t when the states so far have been $s_0, \ldots, s_{t-1}$. So, again a random sequence of states $s_0^\pi, s_1^\pi, \ldots$ and actions $a_0^\pi, a_1^\pi, \ldots$ 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.

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

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

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

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*.

3.2 Computing Optimal Policies via Linear Programming

The key observation to derive (approximations of) optimal policies is that we only need to know (an approximation of) the vector $V^*(s)$.

Lemma 2. *Let $(W_s)_{s \in \mathcal{S}}$ be a vector such that $|W_s - V^*(s)| \leq \epsilon$ for all s for an $\epsilon \geq 0$, then the policy π that in state s chooses the action a that maximizes $r_a(s) + \gamma \sum_{s' \in \mathcal{S}} p_a(s, s') \cdot W_{s'}$ fulfills $V(\pi, s) \geq V^*(s) - \frac{2\epsilon}{1-\gamma}$ for all s.*

Note that in particular the case $\epsilon = 0$ tells us that the policy π will be optimal if $W_s = V^*(s)$.

Proof. Note that W_s approximates the expected reward of an optimal policy π^* starting from s. Our policy might be suboptimal; its expected reward is $V(\pi, s)$. We will show that nonetheless, W_s is also a good approximation.

To this end, let $\hat{s}$ be the state s for which $W_s - V(\pi, s)$ is largest. Let $\delta = W_{\hat{s}} - V(\pi, \hat{s})$. We will show that $\delta \leq \frac{1+\gamma}{1-\gamma}\epsilon$. This then implies that for every state s we have $V(\pi, s) \geq W_s - \delta \geq V^*(s) - \epsilon - \delta \geq V^*(s) - \frac{2\epsilon}{1-\gamma}$. In order to derive the desired bound for δ, let us consider the following.

$$\begin{aligned} V(\pi, \hat{s}) &= r_{\pi(\hat{s})}(\hat{s}) + \gamma \sum_{s' \in \mathcal{S}} p_{\pi(\hat{s})}(\hat{s}, s') \cdot V(\pi, s') \\ &\geq r_{\pi(\hat{s})}(\hat{s}) + \gamma \left(\sum_{s' \in \mathcal{S}} p_{\pi(\hat{s})}(\hat{s}, s') \cdot W_{s'} - \delta \right) \end{aligned}$$

Observe that our choice of $\pi(\cdot)$ ensures that $\pi(\hat{s})$ maximizes $r_{\pi(\hat{s})}(\hat{s}) + \gamma \sum_{s' \in \mathcal{S}} p_{\pi(\hat{s})}(\hat{s}, s') \cdot W_{s'}$. Hence, we can replace $\pi(\hat{s})$ by the optimal policy $\pi^*(\hat{s})$ and only decrease the value of the expression. In addition, using that W_s and $V^*(s)$ can only differ by at most ϵ, we get:

$$\begin{aligned} V(\pi, \hat{s}) &\geq r_{\pi^*(\hat{s})}(\hat{s}) + \gamma \left(\sum_{s' \in \mathcal{S}} p_{\pi^*(\hat{s})}(\hat{s}, s') \cdot W_{s'} - \delta \right) \\ &\geq r_{\pi^*(\hat{s})}(\hat{s}) + \gamma \left(\sum_{s' \in \mathcal{S}} p_{\pi^*(\hat{s})}(\hat{s}, s') \cdot V^*(s') - \epsilon - \delta \right) \\ &= \left(r_{\pi^*(\hat{s})}(\hat{s}) + \gamma \sum_{s' \in \mathcal{S}} p_{\pi^*(\hat{s})}(\hat{s}, s') \cdot V^*(s') \right) - \gamma(\epsilon + \delta) \\ &= V^*(\hat{s}) - \gamma(\epsilon + \delta) \ . \end{aligned}$$

We also have $V(\pi, \hat{s}) = W_{\hat{s}} - \delta \leq V^*(\hat{s}) + \epsilon - \delta$ by the definition of δ. In combination, this gives us $\epsilon - \delta \geq -\gamma(\epsilon + \delta)$ and therefore $\delta \leq \frac{1+\gamma}{1-\gamma}\epsilon$. □

So, all we would need to know are the values of $V^*(s)$. Unfortunately, unlike in the finite horizon case, there is no simple base of the recursion. Therefore, computing them is more complicated here.

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

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

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.

3.3 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 $W^{(0)}$. This method is called *value iteration*.

Theorem 3. *For any starting point $W^{(0)}$, the sequence $W^{(0)}, W^{(1)}, \ldots$ defined by value iteration converges to V^*. More precisely, for every $\epsilon > 0$, there is a $t_0 \in \mathbb{N}$ such that for all $t \geq t_0$ we have $|W_s^{(t)} - V^*(s)| < \epsilon$.*

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 4. *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 3.

Proof of Theorem 3. By Lemma 4, 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, for each $\epsilon > 0$ there has to be a t_0 such that $\gamma^t d(W^{(0)}, V^*) < \epsilon$ for $t \geq t_0$. □

3.4 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 that 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 5. *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 $V(\pi'_{i-1}, s)$. □

Notes

Lecture notes based on [2].

References

[1] Sutton, Richard S., and Andrew G. Barto. "Reinforcement learning: An introduction". MIT press, 2018.

[2] Class notes by Thomas Kesselheim: https://tcs.cs.uni-bonn.de/doku.php?id=teaching:ss20:vl-aau