CSE6243: Advanced Machine Learning **Fall 2023**

Lecture 20: Markov Decision Process: Bellman Recursion

Lecturer: Bo Dai *Scribes: Uzair Akbar, Yantong Lin*

Note: *LaTeX template courtesy of UC Berkeley EECS Department.*

20.1 Comparison of Supervised Learning and Reinforcement Learning

Table 20.1: Comparison of supervised learning and reinforcement learning.

	Supervised Learning	Reinforcement Learning
Target Function	$p(.\|X) : \mathcal{X} \to \Delta(\mathcal{Y})$	$\pi(.\|S) : \mathcal{S} \to \Delta(\mathcal{A})$
Loss Function	$\min_\theta \hat{\mathbb{E}}_{\mathcal{D}}[\ell(p_\theta(.\|X), Y)]$	$\max_\pi \mathbb{E}_\pi[v(\pi)]$

The difference between *supervised learning (SL)* and *reinforcement learning (RL)* can be summarized in Table 20.1. Essentially, the data generating distribution remains fixed in SL whereas it changes in RL. Therefore we collect the samples $\mathcal{D}$ in SL only once and then minimize the empirical loss over $\mathcal{D}$. Conversely, in RL as the agent reacts with the environment the policy π changes and subsequently so does the data generating process/distribution. Therefore we need to collect new samples periodically in order to minimize the loss. We can therefore think of RL as SL with a dynamic data generating process/distribution.

20.2 Markov Decision Process

A *Markov decision process (MDP)* μ is defined as a tuple

$$\mu \coloneqq \langle \mathcal{S}, \mathcal{A}, R, P, \gamma, \mu_0 \rangle,$$

where

- $\mathcal{S}$ is the state space.
- $\mathcal{A}$ is the action space.
- $R(s, a) : \mathcal{S} \times \mathcal{A} \to [0, R_{\max})$ is the reward function for some constant $R_{\max} > 0$. We shall write $r_{s,a} \coloneqq R(s, a)$ or $r_t \coloneqq R(s_t, a_t)$ as shorthand.
- $P(.|s, a) : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the transition probability.
- $\gamma \in [0, 1)$ is the discount factor.
- $\mu_0 : \mathcal{S} \to \Delta(\mathcal{S})$ is some initial distribution over the set of states.

Given such an MDP μ, an agent interacts with the environment by first sampling an initial state $s_0 \sim \mu_0$. Then at each time-step $t \geq 0$ the agent performs an action $a_t \sim \pi(.|s_t)$ and observes the resulting reward $r_t \coloneqq R(s_{t-1}, a_{t-1})$ and transitions to the next state $s_{t+1} \sim P(.|s_t, a_t)$. This continues until $t = t_{\max}$. A depiction of this is given in Fig. 20.1. For a *finite horizon problem*, $t_{\max}$ is finite. For an *infinite horizon problem*, $t_{\max} \to \infty$.

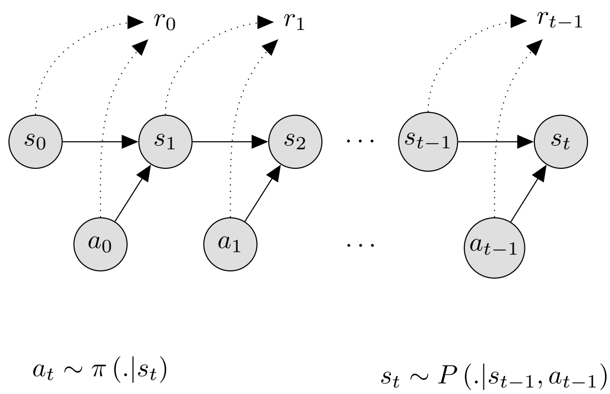

Figure 20.1: MDP Interplay setting.

20.3 Value Functions

20.3.1 Value Functions

The *value function* of a policy π is defined as

$$v(\pi) \coloneqq \mathbb{E}_{\mu_0, P, \pi}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right],$$

where $\gamma \in [0, 1)$ so that $v(\pi)$ converges. For finite horizon, $\gamma = 1$ but for simplicity of notation we only consider the infinite horizon case here.

We also have the value function

$$V^\pi(s) \coloneqq \mathbb{E}_{P,\pi}\left[\sum_{t=0}^{\infty} \gamma^t r_t \middle| s_0 = s\right].$$

Therefore,

$$v(\pi) = \mathbb{E}_{s \sim \mu_0}[V^\pi(s)].$$

If state-space $\mathcal{S}$ is finite then we can represent $V^\pi(s)$ as a vector

$$V^\pi \coloneqq \begin{bmatrix} V^\pi(s_0) \\ V^\pi(s_1) \\ \vdots \\ V^\pi(s_n) \\ \vdots \\ V^\pi(s_{|\mathcal{S}|}) \end{bmatrix} \in \mathbb{R}^{|\mathcal{S}| \times 1}, s_n \in \mathcal{S}.$$

We also define a *state-action value function* for policy π as

$$Q^\pi(s,a) := \mathbb{E}_{P,\pi}\left[\sum_{t=0}^{\infty}\gamma^t r_t \middle| s_0 = s, a_0 = a\right],$$

which can also be represented as a vector of length $|\mathcal{S}||\mathcal{A}|\times 1$. Furthermore, $Q^\pi(s,a)$ is related to $V^\pi(s)$ as

$$V^\pi(s) = \mathbb{E}_{a\sim\pi(.|s)}[Q^\pi(s,a)]. \tag{20.1}$$

20.3.2 Optimal Value Functions

The *optimal policy* is defined as

$$\pi^* := \arg\max_{\pi} v(\pi).$$

Subsequently, the *optimal value functions* are defined as

$$V^*(s) := V^{\pi^*}(s),$$
$$Q^*(s,a) := Q^{\pi^*}(s,a).$$

Lemma 20.1 *π^* is greedy with respect to Q^*, i.e.*

$$V^*(s) = \max_{a\in A} Q^*(s,a)\,. \tag{20.2}$$

Proof: (Sketched)
It is obvious that $V^*(s) \le \max_a Q^*(s,a)$ by definition. Assume for contradiction that $V^*(s) < \max_a Q^*(s,a)$ at some s. We can construct a new policy π' that is the same as π^* except that it chooses among $\arg\max_a Q^*(s,a)$ at s. Then, (intuitively) we have $V^{\pi'}(s) > V^*(s)$ and $V^{\pi'}(s') \ge V^*(s')$ for other s', which contradicts the optimality of V^*. ∎

20.4 Bellman Equations

20.4.1 Bellman Equation

Lemma 20.2 *For $s' \sim P(.|s,a)$,*

$$Q^\pi(s,a) = R(s,a) + \gamma\mathbb{E}_{P,\pi}\left[V(s')\right]. \tag{20.3}$$

Proof:

$$\begin{aligned}
Q^\pi(s,a) &:= \mathbb{E}_{P,\pi}\left[\sum_{t=0}^{\infty}\gamma^t r_t \middle| s_0 = s, a_0 = a\right],\\
&= \mathbb{E}_{P,\pi}\left[\gamma^0 r_0 + \sum_{t=1}^{\infty}\gamma^t r_t \middle| s_0 = s, a_0 = a\right],\\
&= R(s,a) + \gamma\mathbb{E}_{s'\sim P(.|s,a)}\left[\mathbb{E}_{P,\pi}\left[\sum_{t=0}^{\infty}\gamma^t r_t \middle| s_0 = s'\right]\right], \qquad (20.4)\\
&= R(s,a) + \gamma\mathbb{E}_{s'\sim P(.|s,a)}\left[V^\pi(s')\right],
\end{aligned}$$

where Eq. (20.4) is obtained by pulling a γ outside of the sum. ■

Plugging Eq. (20.3) into Eq. (20.1), we have

$$\begin{aligned} V^{\pi}(s) &= \mathbb{E}_{a\sim\pi(.|s)}\left[Q^{\pi}(s,a)\right] \\ &= \mathbb{E}_{a\sim\pi(.|s)}\left[R(s,a)+\gamma\mathbb{E}_{s'\sim P(.|s,a)}\left[V^{\pi}(s')\right]\right] \\ &= \mathbb{E}_{a\sim\pi(.|s)}\left[R(s,a)\right]+\gamma\mathbb{E}_{a\sim\pi(.|s)}\left[\mathbb{E}_{s'\sim P(.|s,a)}\left[V^{\pi}(s')\right]\right] \\ &= R(s)+\gamma\mathbb{E}_{s'\sim P(.|s)}\left[V^{\pi}(s')\right] \end{aligned} \tag{20.5}$$

Similarly, by plugging Eq. (20.1) into Eq. (20.3), we have

$$Q^{\pi}(s,a) = R(s,a)+\gamma\mathbb{E}_{P,\pi}\left[Q^{\pi}(s',a')\right], \tag{20.6}$$

where $a' \sim \pi(.|s')$. Equation (20.5) and Eq. (20.6) are the *Bellman equations* for $V^{\pi}(s)$ and $Q^{\pi}(s,a)$ respectively.

20.4.2 Bellman Optimal Equation

Using Eq. (20.2) and the Bellman equation for general policy we obtain the Bellman optimal equation for V^*, i.e.

$$V^*(s) = \max_{a\in A}\left\{R(s,a)+\gamma\mathbb{E}_{s'\sim P(\cdot|s,a)}\left[V^*(s')\right]\right\}$$

Similarly, we have the Bellman optimal equation for Q^*, i.e.

$$Q^*(s,a) = R(s,a)+\gamma\mathbb{E}_{s'\sim P(\cdot|s,a)}\left[\max_{a'\in A} Q^*(s',a')\right]$$

20.5 Problems in Reinforcement Learning

20.5.1 Prediction and Control

Based on the formulation above, we can identify 2 main problems in reinforcement learning:

- Policy Evaluation (Prediction): Given a policy π, compute the value functions (V^{π}, Q^{π}).
- Policy Optimization (Control): Find an optimal policy (π^*) and the corresponding value functions (V^*, Q^*).

20.5.2 Planning and Learning

We also consider the following 2 settings:

- Planning: The environment is given as a model (direct access to transition probabilities).
- Learning: The environment is unknown and we need to learn from experience (sample from distributions).

20.6 Algorithms in the Planning Setting

20.6.1 Policy Evaluation via Solving Linear Equations in the Planning Setting

We revisit the Bellman Equation for V^π in a slightly different form.

$$V^\pi(s) = R^\pi(s) + \gamma \mathbb{E}_{s' \sim P(\cdot|s)} \left[V^\pi(s')\right] ,$$

where $P(s'|s) = \sum_{a \in A} \pi(a|s) P(s'|s,a)$ standing for the transition probability from s to s' following policy π.

It is easy to see that the Bellman equations can be expressed in the following linear algebra form.

$$V^\pi = R^\pi + \gamma P^\pi V^\pi ,$$

where $V^\pi \in \mathbb{R}^{|\mathcal{S}|}$ is the value vector, $R^\pi \in \mathbb{R}^{|\mathcal{S}|}$ is the reward vector, $P \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{S}|}$ is the transition probability matrix.

This is just a linear system of equations with an analytical solution

$$V^\pi = (I - \gamma P^\pi)^{-1} R^\pi .$$

In fact, if we expand the matrix inverse, we can rediscover the matrix form of the Bellman Equation for V^π

$$V^\pi = (I - \gamma P^\pi)^{-1} R^\pi = (I + \gamma P^\pi + (\gamma P^\pi)^2 + \ldots) R^\pi .$$

20.6.2 Policy Optimization via Iterative Algorithms in the Planning Setting

The analytical prediction algorithm is not feasible for large state space since it runs in $|\mathcal{S}|^3$.

20.6.2.1 Value Iteration

Consider a policy improvement operator Φ defined as

$$\Phi(V) = \max_{a \in \mathcal{A}} \left\{ R(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)} \left[V(s')\right] \right\} ,$$

the value iteration algorithm is defined in Algorithm 1.

Algorithm 1 Value Iteration

1: Initialize V_0
2: **while** $\|V - \Phi(V)\| \geq \epsilon$ **do**
3: $\quad V \leftarrow \Phi(V)$
4: **end while**

20.6.2.2 Policy Iteration

Another iterative algorithm is policy iteration (Algorithm 2) which iterates over the policies.

Proof of convergence of the algorithms will be covered in the next lecture.

Algorithm 2 Policy Iteration

Initialize π_0
for $k = 0, 1, 2, \ldots$ or $\pi_{k+1} \neq \pi_k$ or $\|V^{\pi_k} - V^{\pi_{k+1}}\| \geq \epsilon$ **do**
    (Policy Evaluation) Evaluate V^{π} (e.g., using the Bellman Expectation Equation)
    (Policy Improvement) $\pi_{k+1}(s) \leftarrow \arg\max_{a \in A} \left\{ R(s,a) + \gamma \mathbb{E}_{s' \sim P(s'|s,a)} V^{\pi_k}(s') \right\}$
end for