CAUTION: These lecture notes are under construction. You may find parts that are incomplete.

# 8 Markov Decision Processes

In the previous chapter, we covered a simple example of how to use Markov reward processes for informing in-game strategy in sports. In that example, the values of all states were known and static—from a starting state, we chose the action corresponding to the resulting state with the highest value. The chosen action, therefore, influences the value of that starting state. And future actions influence the values of the resulting states between which we are choosing with our current action. The transition probabilities were estimated using a dataset of observed transitions, but now we are choosing actions which influence those transitions.

In order to optimize decision-making in this setting, we need a framework to update recommended actions while simultaneously updating the estimated value of each state. The model we use for this is called a *Markov decision process* (MDP). An MDP is defined by four components:

1. a state space,
2. a transition probability function,
3. a reward function, and
4. an *action space*.

## 8.1 Actions

The action space is what distinguishes an MDP from a Markov reward process. The concept is that on each transition from one state to a next, there is an action $a \in \mathcal{A}$ chosen which influences the transition probability to the next state. The *action space* $\mathcal{A}$ is the set of all possible actions.

Because the action chosen from state $s \in \mathcal{S}$ determines the transition probability, the transition probability function $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ for an MDP depends on the action, not just the starting and resulting states. We write $p(s, a, s')$ to denote the probability of transitioning to state $s'$ given that action $a$ was taken in state $s$. The MDP also allows for the possibility that the reward function $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S}$ depends on the action as well. We use $r(s, a, s')$ to denote the reward accrued on the transition from state $s$ to state $s'$ on action $a$.

## 8.2 Policies

A *policy* is a function $\pi : \mathcal{S} \to \mathcal{A}$ which maps states to actions. In other words, a policy tells you what action to take from any state. For a given policy $\pi$, we use $\pi(s)$ to denote the action taken from state $s$ according to the policy. We can think of a policy as a pre-determined set of actions for a decision-maker to follow: Given the state $s$, there is a simple for for determining the corresponding action $a$.

Once the policy $\pi$ is specified, all transition probabilities are specified, and we can determine the value function $v_\pi$ corresponding to the policy $\pi$. This bears repeating: The value function depends on the policy. As we learned in the previous chapter, this value function will satisfy the Bellman equation:

$$v_\pi(s) = \sum_{s' \in \mathcal{S}} p(s, \pi(s), s') \cdot (p(s, \pi(s), s') + v_\pi(s')).$$

The objective of an MDP is to learn the *optimal policy* $\pi^*$ which gives the appropriate action $\pi^*(s)$ from state $s$ to maximize the value $v(s)$ of state $s$. The optimal action from each state depends on the value of each possible resulting state, but the value of each state depends on the policy, i.e. the action taken from each state. We have a circular dependence here, but fortunately it is a similarly well-behaved circular dependence as we saw with the Bellman equation in the previous chapter.

### 8.2.1 Policy Iteration

We can generalize the Bellman equation in this setting to make the following claim about $v^*$, the value function corresponding to the optimal policy $\pi^*$:

$$v^*(s) = \max_{a \in \mathcal{A}} \left[ \sum_{s' \in \mathcal{S}} p(s, a, s') \cdot (p(s, a, s') + v^*(s')) \right]. \tag{1}$$

The strategy for finding $v^*$ satisfying (1) is the same as with value iteration for Markov reward processes in the previous chapter. We naively initialize the value function and then repeatedly update it using the equation (1). At each step, the policy updates according to the optimal action for the current iteration of the value function. We repeat this step until convergence in the policy, rather than convergence in the value function, and this is known as *policy iteration*.

In detail, the policy iteration algorithm is defined as follows:

1. Initialize $v^{(0)}(s) = 0 \quad \forall\ s \in \mathcal{S}$.
2. For step $n = 1, 2, ...$ (until convergence in $\pi$)
   (a) Update

$$v^{(n)}(s) = \max_{a \in \mathcal{A}} \left[ \sum_{s' \in \mathcal{S}} p(s, a, s') \cdot \Big( r(s, a, s') + v^{(n-1)}(s') \Big) \right] \qquad \forall\ s \in \mathcal{S},$$

$$\pi^{(n)}(s) = \arg\max_{a \in \mathcal{A}} \left[ \sum_{s' \in \mathcal{S}} p(s, a, s') \cdot \Big( r(s, a, s') + v^{(n-1)}(s') \Big) \right] \qquad \forall\ s \in \mathcal{S}.$$

Under mild assumptions, we are guaranteed that policy iteration will converge to the optimal policy $\pi^*$ and the corresponding value function $v^*$. This solution $\pi^*$ gives the optimal action from each starting state.

## 8.3 Examples