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 π, 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 π is specified, all transition probabilities are specified, and we can determine the value function v_π corresponding to the policy π. 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:

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

The objective of an MDP is to learn the *optimal policy* π^* 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 π^*:

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

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 π)
   (a) Update

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

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

8.3 Examples

Nadimpalli and Hasenbein (2013) used an MDP to model the decision of whether to challenge an in/out call in tennis. As their **game state** they used (1) the outcome of the point, (2) the score, (3) the number of challenges remaining, (4) the probability that the call is correct and (5) the result of a successful challenge. Each of these components is descretized into a few possibilities, yielding a total of 694 non-terminal states. They provide the example state 233132, indicating that the server lost the current played point; the game is at duece; they are left with one challenge; they are s_3% sure that he can successful challenge; and the point does not get replayed if they challenge correctly. In their model the decision-maker is the server, and the **action space** is (c, nc), meaning *challenge* or *do not challenge*. They modelled their **transition probability function** by introducing some parameters to describe point outcome probability. They defined their **reward function** as 1 for each transition into a game-winning state (for the server) and 0 for all other transitions. Having defined these components of their MDP, they followed policy iteration to calculate the optimal **policy**, yielding a table that recommended whether or not to challenge from each possible game state.

Van Roy et al. (2021) used an MDP to model player decision-making in soccer. As the **game state** they used the location of the ball discretized into 374 squares on offense and 192 squares on defense, a total of 567 non-terminal states along with three terminal states: lost possession, no goal and goal. In their model the decision-maker is the player in possession of the ball, and the **action space** contains the options of attempting a shot or attempting to dribble the ball into any of the 374 offensive squares. They model the **transition probability** by modelling a distribution of player intention, which we do not observe. They defined their **reward function** as 1 for each transition into the goal state and 0 for all other transitions. Rather than using policy iteration to determine the optimal policy, they present a sophisticated Bayesian model for learning the policies followed by specific teams. They noted that the computation time for estimating their policy model was 15 hours using a server with 128GB of RAM.

Hahn et al. (2024) used an MDP to model the decision made by a baserunner of how long to take their leadoff from first base. As their **game state** they used (1) which bases were occupied by runners, (2) the ball-strike count, (3) the number of disengagements already used by the pitcher, and (4) the number of outs. This is a total number of 864 non-terminal states. For example, the state $((1, 0, 0), (3, 2), 1, 2)$ means that there is a runner on first base; the count is full; the pitcher has used one disengagement; and there are two outs. In their model the decision-maker is the baserunner, and the **action space** is $[0, 20]$, meaning that the runner can take any lead distance between 0 and 20 feet. They modelled their **transition probability function** using mixed-effect logistic regressions with the lead distance as a fixed effect. They defined their **reward function** as the number of runs scored on the transition between two states. Havine defined these components of their MDP, they followed policy iteration to calculate the optimal **policy**, yielding a table that recommended the optimal lead distance for the average runner in each possible game state. They also generalized the model to make recommendations for baserunners of any talent level.

References

Chan TCY, Fernandes C, Walker R (2024) "No More Throwing Darts at the Wall: Developing fair handicaps for darts using a Markov decision process" *MIT Sloan Sports Analytics Conference* `https://www.sloansportsconference.com/research-papers/no-more-throwing-darts-at-the-wall-developing-fair-handicaps-for-darts-using-a-markov-decision-process`

Hahn J, Powers S, Pai M, Schaefer A (2024) "The Two-Foot Rule: A game theoretic analysis of the pickoff limit in Major League Baseball" *Cascadia Symposium on Statistics in Sports* `https://www.youtube.com/watch?v=oOvvNnDCD5Y`

Nadimpalli VK, Hasenbein JJ (2013) "When to challenge a call in tennis: A Markov decision process approach" *Journal of Quantitative Analysis in Sports* 9(3): 229-238 `https://doi.org/10.1515/jqas-2012-0051`

Sandholtz N, Bornn L (2020) "Markov decision processes with dynamic transition probabilities: An analysis of shooting strategies in basketball" *Annals of Applied Statistics* 14(3): 1122-1145 `https://doi.org/10.1214/20-AOAS1348`

Sandholtz N, Wu L, Puterman M, Chan TCY (2024) "Learning risk preferences in Markov decision processes: An application to the fourth down decision in the national football league" *Annals of Applied Statistics* 18(4):3205-3228 `https://doi.org/10.1214/24-AOAS1933`

Van Roy M, Robberechts P, Yang W, De Raedt L, Davis J (2021) "Learning a Markov model for evaluating soccer decision-making" *International Conference on Machine Learning* `https://lirias.kuleuven.be/retrieve/627708`