Markov Decision Processes & Reinforcement Learning

Frederick Widjaja

Markov Decision Process

In a Markov Decision Process (MDP), we have:

- A set of states $\mathcal{S}$ that the agent may observe.
- A set of actions $\mathcal{A}$ that the agent gets to choose.
- A reward function $R(s,a) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, which is the reward the agent gets when taking action a under state s.
- A transition model $T(s'|s,a) : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0,1]$, which is the probability of a state transition to s' when the agent takes action a under state s. This is what the environment does to the agent.

The goal of the agent is therefore to learn a policy $\pi : \mathcal{S} \to \mathcal{A}$, which decides what action to take under state s, to interact with its environment such that it maximizes future rewards.

Finite Horizon

For a world with a finite time horizon T, our objective is to maximize

$$\sum_{t=1}^{T} R(s_t, a_t)$$

where s_t is the state that the agent is at and a_t is the action that the agent takes at time t. Replacing a_t with our policy π, we have

$$\sum_{t=1}^{T} R(s_t, \pi(s_t))$$

What would we do in the last step of the game? We simply choose the action that maximizes the reward in the last step before the world ends. Let us index our policy π with the number of steps before the game ends. Then

$$\pi_1(s) = \arg\max_{a \in \mathcal{A}} R(s,a)$$

From this, we can define value, which is the reward that we can get (by following an optimal policy) in state s: $V_k : \mathcal{S} \to \mathbb{R}$. Again, this is indexed by the number of steps away from the end of the game

$$V_1(s) = \max_{a \in \mathcal{A}} R(s,a)$$

If we are instead two steps away from the game ending, then the policy is trying to maximize is the reward on the second last step, plus the reward on the last step (assuming that we follow the optimal policy on that step)

$$\begin{aligned}\pi_2(s) &= \arg\max_{a\in\mathcal{A}}\left\{R(s,a) + \max_{a'} E_{s'}\left[R(s',a')\right]\right\}\\ &= \arg\max_{a\in\mathcal{A}}\left\{R(s,a) + \sum_{s'} T(s'|s,a)\max_{a'} R(s',a')\right\}\\ &= \arg\max_{a\in\mathcal{A}}\left\{R(s,a) + \sum_{s'} T(s'|s,a)V_1(s')\right\}\end{aligned}$$

We can then use this to define the value two steps away from the game ending

$$V_2(s) = \max_{a\in\mathcal{A}}\left\{R(s,a) + \sum_{s'} T(s'|s,a)V_1(s')\right\}$$

This generalizes to

$$V_k(s) = \max_{a\in\mathcal{A}}\left\{R(s,a) + \sum_{s'} T(s'|s,a)V_{k-1}(s')\right\}$$

It is then natural to define a "value" function not only for states, but for state-action pairs. This is known as the "Q" function: $Q_k : \mathcal{S}\times\mathcal{A}\to\mathbb{R}$, again indexed by the number of steps away from the end of the game. This function is defined as

$$Q_k(s,a) = R(s,a) + \sum_{s'} T(s'|s,a)V_{k-1}(s')$$

and thus the optimal policy π^* is given by

$$\pi^*_k(s) = \arg\max_{a\in\mathcal{A}} Q_k(s,a)$$

With our "Q" function, we can rewrite our value function as

$$V_k(s) = \max_{a\in\mathcal{A}} Q_k(s,a) = Q_k(s,\pi^*_k(s))$$

with initialization $V_0(s) = 0$

Infinite Horizon

We saw in the previous section how to deal with finite horizon games, but what if the agent lives forever? Our recursion in the previous case did not converge (there is no reason that it would), so we cant simply extend that method to $T\to\infty$. The reason is that the longer we lived the more rewards we would get (simply because we can take more actions). To deal with the infinite horizon case, we have to discount future rewards to make it converge (this may be because future events are not guaranteed to happen, or we are impatient and want rewards now rather than later, etc). We introduce a discount factor $\gamma \in (0, 1)$, and now we want to maximize

$$\sum_{t=1}^{\infty} \gamma^t R(s_t, a_t)$$

Our new "Q", value, and policies are

$$\begin{aligned} Q(s,a) &= R(s,a) + \gamma \sum_{s'} T(s'|s,a) V(s') \\ \pi^*(s) &= \arg\max_{a \in \mathcal{A}} Q(s,a) \\ V(s) &= \max_{a \in \mathcal{A}} Q(s,a) = Q(s, \pi^*(s)) \end{aligned}$$

We can estimate this "Q" function by iteratively updating it to the rewards we get in each step and the expected value going forward. This is known as **value iteration**. It turns out that this "Q" function is guaranteed to converge, but the time it takes to converge will depend on the mixing time of the Markov chain. However, this is out of scope, and we will not be discussing it.

Policy Iteration

Instead of watching our "Q" function converge, we could instead watch our policy (which is something discrete that tells us which action a to take under state s). Let us define a matrix $\mathbf{T}$ whereby the element at row s and column s' is

$$T_{s,s'} = T(s'|s, \pi(s))$$

In other words, this is the probabily of getting to state s' forom state s, given that we are following policy π. Now, we define vectors

- $\mathbf{V} \in \mathbb{R}^{|\mathcal{S}|}$, where $V_s = V(s)$
- $\mathbf{r} \in \mathbb{R}^{|\mathcal{S}|}$, where $r_s = R(s, \pi(s))$

Now, we can write down our problem as

$$\mathbf{V} = \mathbf{r} + \gamma \mathbf{T} \mathbf{V}$$

This is simply a vectorized form of our previous equations, namely

$$V(s) = Q(s, \pi(s)) = R(s, \pi(s)) + \gamma \sum_{s'} T(s'|s, \pi(s)) V(s')$$

This allows us to solve for $\mathbf{V}$ (since this is just a linear equation) and from there, we can optimize our policy π until it converges.

Reinforcement Learning

Value iteration and policy iteration allow you to plan for the optimal policy in a known environment, in which both the reward function and the transition probabilities are known. However, in reinforcement learning, we have unknown rewards and unknown transitions, while the states and actions are still known. There are two different approaches: model-based and model-free reinforcement learning. Model-based says that we're going to learn what the model is, and then use value iteration or policy iteration to solve that Markov Decision Process. Essentially, we loop in which we wander around and gather data, using it to estimate our model, and then plan with V.I. or P.I. What are the kinds of statistics we may want to gather as we wander? Our transition model and reward function.
We gather data by tracking counts and modeling averages. We can define $N_{s,a}$ as the number of times we took action a from state s. We can also define $N_{s,a,s'}$ as the number of times we landed in s' after taking action a in state s. We can then estimate $T(s'|s,a)$ as the ratio $\frac{N_{s,a,s'}}{N_{s,a}}$. The last thing we need to do is track rewards $R_{s,a}^{total}$ as the total reward amount from all of the times we have taken action a in state s, and then estimate $R(s,a)$ as $\frac{R_{s,a}^{total}}{N_{s,a}}$.
This model isn't too ideal. We can't generalize to all states, and we may have to wander a long time to see all our possible states and outcomes. This also uses up a lot of memory in the case we have a lot of states. Planning may also take too long, which leads to our idea of model-free RL. This says that all we really need is the Q Function - we don't need to do planning if we can estimate what the Q Function is, since the Q Function determines everything we need to do.

Q-Learning

The key idea behind Q-Learning is that the sum in the Bellman equation is an expectation

$$\begin{aligned} Q(s,a) &= R(s,a) + \gamma \sum_{s'} T(s'|s,a) \max_{a'} Q(s',a') \\ &= R(s,a) + \gamma E_{s'} \left[\max_{a'} Q(s',a') \right] \\ &= E_{s'} \left[R(s,a) + \gamma \max_{a'} Q(s',a') \right] \end{aligned}$$

Hence, $Q(s,a)$ is an expectation, and moreover, the expectation of something we can compute. For some s and some a, we can take that action and go to s', so we can end up with a future estimate of what that reward is based on where we go. There's an error between what we thought would happen when we took an action and a state, and what ended up happening. Using this error, we can perform a stochastic-gradient-descent-like update to learn the optimal solution.
Our algorithm is the following, where α is our learning rate, and r is the reward we actually got when taking action a in state s:

$$Q_{s,a}^{new} \leftarrow Q_{s,a}^{old} + \alpha \left[\left(r + \gamma \max_{a'} Q_{s',a'}^{old} \right) - Q_{s,a}^{old} \right]$$

The bracket following the learning rate is our expected quantity from the equations above subtracted by $Q_{s,a}^{old}$, which represents the error in our estimations. That is, the parenthetical term is like an estimate arising from the world, and we have stochastic gradient descent with this noisy estimate.