Lecture 18
Markov Decision Processes, Part 1

Alice Gao

November 2, 2021

Contents

1 Learning Goals

By the end of the lecture, you should be able to

- Describe motivations for modeling a decision problem as a Markov decision process.
- Describe components of a fully-observable Markov decision process.
- Describe reasons for using a discounted reward function.
- Define the policy of a Markov decision process.
- Give examples of how the reward function affects the optimal policy of a Markov decision process.

2 Introduction to Markov Decision Processes

2.1 Modeling an ongoing decision process

We'll look at a new tool for solving decision problems involving uncertainty: the Markov decision process.

Previously, we focused on *finite-stage* decision problems—problems with a finite number of stages. It was sufficient to model these problems by incorporating a finite number of decision nodes into a decision network.

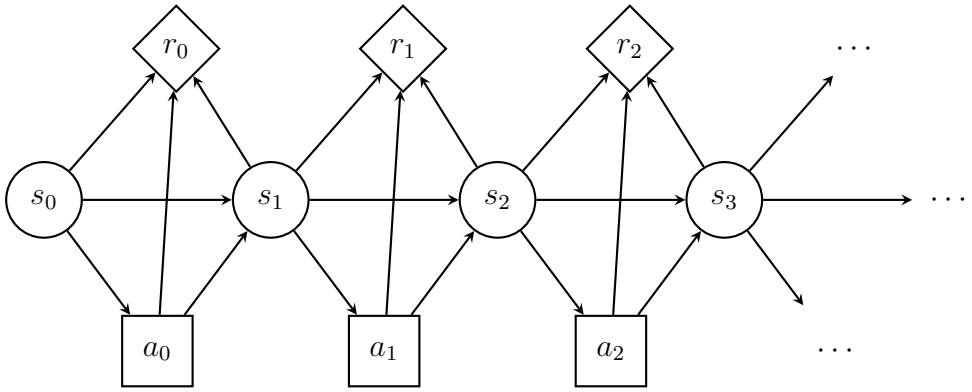
However, in general, we may have to solve *ongoing* decision problems. One case is that a problem could have an *infinite horizon*: the process may go on forever. Another case is that a problem could have an *indefinite horizon*: the agent will eventually stop, but it does not know when it will stop. There may be goal states in the world, but the agent will have to keep exploring until it finds one.

For either type of ongoing problem, because we don't know when or even if the agent will reach an end, we cannot use a decision network to model the problem. Because of the differences between finite-stage and ongoing problems, we will have to consider the utility function differently.

With a finite-stage problem, we can make multiple decisions in sequence and then calculate the utility at the end. However, with an ongoing problem, it does not make sense to consider the utility at the end. There might not be an end for an infinite horizon problem, and we might not know when we'll reach the end in an indefinite horizon problem. Instead, we will consider a sequence of rewards. These rewards may incorporate the costs of actions we took to get to a certain state and any rewards or punishments received along the way.

2.2 A Markov decision process

Let's look at a Markov decision process (MDP) graphically. Because there are an unlimited number of time steps, this picture won't be able to show the entire model. It will only show the first few time steps of the model.



In this example, we start in the state s_0. We can take some action a_0 and transition to the state s_1. Because of this state transition, we will receive a reward r_0.

You can see that this is almost like a decision network. We're using the same nodes: ellipses for chance nodes, rectangles for decision nodes, and diamonds for utility nodes. You can imagine this process carries on with more states, actions, and rewards.

To define a Markov decision process, we need to specify:

- A set of states S.
- A set of actions A.
- Transition probabilities $P(s'|s,a)$. Like in the hidden Markov model, we will assume the Markov process is stationary, so the transition probabilities are constant for each time step.
- A reward $R(s,a,s')$. In its most general form, the reward depends on the starting state, the action taken, and the new state. Sometimes, not all of these variables will be needed.

2.3 Rewards

For a finite-stage problem, modelling the agent's utility of each state is easy. We can list all the combinations of decisions and calculate the utility for each combination. How can we model the agent's utility when there is an unlimited number of states?

We will consider three ways. For these reward functions, we'll simplify our notation a little bit. We'll assume the reward only depends on the state s we are entering: $R(s)$ is the reward of entering state s.

- Total reward:
$$R(s_0) + R(s_1) + R(s_2) + \cdots .$$
The problem here is that if the sum is infinite, then we cannot compare two different policies of a Markov decision process.

- Average reward:
$$\lim_{n\to\infty} \frac{1}{n} \left(R(s_0) + R(s_1) + R(s_2) + \cdots\right).$$
The problem here is that if the total reward is finite, this limit is zero.

- Discounted reward:
$$R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \cdots$$
where $0 \leq \gamma < 1$ is the **discount factor**.

  This idea might be foreign to you, but it has many uses in microeconomics and game theory.

  One reason behind discounting each successive reward is that we prefer getting the same reward sooner rather than later: if we are to eventually receive a reward $R(s_i)$, we would prefer to receive it today rather than tomorrow or after tomorrow. A second reason is that everyday, there is a chance tomorrow won't come. For example, if $\gamma = 0.9$, this could represent a 10% chance of not reaching the next state.

  An advantage to using the discounted reward is that it is finite, so we can compare two policies.

We will use the discounted reward function.

2.4 Variations of MDPs

Two variations of MDPs will be discussed.

In a **fully-observable MDP** (or just an MDP), the agent knows what state it is in.

In a **partially-observable MDP (POMDP)**, the agent does not know what state it is in, but it can get some noisy signal of the state. You can intuitively think of this as combining an MDP and a hidden Markov model.

In this course, we will use fully-observable MDPs. However, in practice, there are many situations where the agent cannot easily tell what state it is in, so using a POMDP is more appropriate.

3 A Grid World

As a running example, let's consider a 3×4 grid world.

Example: A robot is in a 3×4 grid world. What should the robot do to maximize its rewards?

	1	2	3	4
1	Start			
2		X		-1
3				$+1$

- Let s_{ij} be the position in row i and column j.
- s_{11} is the initial state.
- There is a wall at s_{22}.
- s_{24} and s_{34} are goal states. The robot escapes the world at either goal state. The value in each goal state cell is its corresponding reward.

Since this is an ongoing decision problem, we need to use an MDP.

Example: Here is an MDP for the 3×4 grid world. We need to define the possible actions, the transition probabilities, and the reward function.

There are four actions: up, down, left, and right (relative to us, the observers). Every action is possible in every state.

The transition model $P(s'|s, a)$ is defined as below:

- An action achieves its intended effect with probability 0.8.
- An action leads to a 90-degree left turn (relative to the robot) with probability 0.1.
- An action leads to a 90-degree right turn (relative to the robot) with probability 0.1.
- If the robot bumps into a wall by taking an action, it stays in the same square.

The reward function $R(s)$ is simplified here, like previously discussed: $R(s)$ is the reward of entering state s. We'll define:

- $R(s_{24}) = -1$.
- $R(s_{34}) = 1$.
- Otherwise, $R(s) = -0.04$.

It makes sense to have a small, negative utility in each non-goal state since it takes time and energy to reach any state. The more states we visit, the more time and energy we need to spend.

Understanding the transition model can be tricky, so here is a practice example to help.

Problem: The robot is currently in s_{14} and tries to move to our right. What is the probability that the robot stays in s_{14}?

(A) 0.1
(B) 0.2
(C) 0.8
(D) 0.9
(E) 1.0

	1	2	3	4
1	Start			→
2		X		−1
3				+1

Solution: The intended direction is to our right. With probability 0.8, the robot will go to our right, but it will bump into the right wall and stay in the same state. With probability 0.1, the robot will go up, but it will bump into the top wall and stay in the same state. With probability 0.1, the robot will go down and move to s_{24}.

Therefore, the robot will stay in s_{14} with probability $0.8 + 0.1 = 0.9$. The correct answer is (D).

4 Policies

When we solve an MDP, we are looking for a policy that tells the agent what to do. In general, this is non-trivial. Let's explore some ideas for such a policy.

Problem: If the environment is deterministic, an optimal solution to the grid world problem is the fixed action sequence: "down, down, right, right, and right".

(A) True
(B) False
(C) I don't know

	1	2	3	4
1	Start			
2		X		−1
3				+1

Solution: In a deterministic world, a fixed action sequence is indeed a possible optimal solution since we know exactly where our actions will take us. In fact, there are several fixed action sequences which are also optimal. For example, we could begin by going right instead of down.

The correct answer is (A).

The takeaway is that deterministic worlds are very nice to work with.

Problem: Consider the action sequence "down, down, right, right, and right". This action sequence could take the robot to more than one square with positive probability.

(A) True
(B) False
(C) I don't know

	1	2	3	4
1	Start			
2		X		−1
3				+1

Solution: Consider s_{11}. Whenever we try to go down, we could instead go left with probability 0.1. Whenever we try to go right, we could also instead go up with probability 0.1. If we go up at every step, then we will always remain in s_{11}. Then it is possible for the robot to end in s_{11} with probability 0.1^5.

Consider s_{14}. We could instead have taken the following steps: right (probability 0.1), right (probability 0.1), right, (probability 0.8), right (probability 0.8), and right (probability 0.8). In this case, we would end up in s_{14} with probability $0.1^2 0.8^3$.

There are many other states we could have reached with positive probability, but these are just two examples. (I conjecture this sequence of actions can take the robot to any square in the grid world!) The correct answer is (A).

The second example illustrates that a fixed sequence of actions is not enough for a policy in an MDP since a fixed sequence could take us anywhere in the world. This is due to the uncertainty in the transition probabilities. In fact, a policy needs to take into account any contingencies: any state could be reached, so we need to know how to act in each state.

A policy specifies what the agent should do as a function of the current state. A policy is

- **non-stationary** if it is a function of the state and the time.
- **stationary** if it is a function of only the state.

Since we're only considering stationary MDPs, we will only consider stationary policies.

5 The Optimal Policies of the Grid World

The optimal policy of the grid world changes based on $R(s)$ for any non-goal state s. It shows a careful balancing of risk and reward.

In our case, the reward is getting to the +1 state. If the agent wants to get there as soon as possible, there will be two paths (initially going right or down).

The risk is falling into the −1 state by accident. There is also the risk of accumulating too many negative rewards through long explorations.

We'll consider five cases for $R(s)$; these are not comprehensive, but should help illustrate

the possibilities:

(1) When life is quite unpleasant: $-0.4278 < R(s) < -0.0850$

(2) When life is painful: $R(s) < -1.6284$

(3) When life is unpleasant: $R(s) = -0.04$

(4) When life is only slightly dreary: $-0.0221 < R(s) \leq 0$

(5) When life is good: $0 < R(s)$

In each case, make a guess about the optimal policy first before looking at the solution. The titles should give you some hints.

5.1 When life is quite unpleasant

Problem: When $-0.4278 < R(s) < -0.0850$, what does the optimal policy look like?

Solution: In this case, the agent tries to take the shortest route to the $+1$ state, but it is willing to risk falling into the -1 state by accident.

Since there are two shortest paths from s_{11} (down or right) to the $+1$ state, but the down path is safer (less risk of falling into the -1 state), the agent chooses to go down first.

If the agent accidentally gets to s_{14}, it tries to get back on the right path.

	1	2	3	4
1	$\downarrow$	$\rightarrow$	$\downarrow$	$\leftarrow$
2	$\downarrow$	X	$\downarrow$	-1
3	$\rightarrow$	$\rightarrow$	$\rightarrow$	$+1$

5.2 When life is painful

Problem: When $R(s) < -1.6284$, what does the optimal policy look like?

Solution: Because each step is so painful in this case, the agent (generally) heads straight for the nearest exit, even if the exit is worth -1.

In s_{21} and along the bottom row, the nearest goal state is the $+1$ state, so the agent goes right.

Along the top row and above the X, the closest state is the -1 state, so the agent follows that path.

In s_{23}, the -1 state is closer, but going to the $+1$ state is actually better on average, so the agent chooses to go down instead.

	1	2	3	4
1	$\rightarrow$	$\rightarrow$	$\rightarrow$	$\downarrow$
2	$\downarrow$	X	$\downarrow$	-1
3	$\rightarrow$	$\rightarrow$	$\rightarrow$	$+1$

In the next three examples, life will only get better. Let's see how the optimal policy changes.

5.3 When life is unpleasant

Problem: When $R(s) = -0.04$, what does the optimal policy look like?

Solution: The optimal policy is relatively conservative.

The agent prefers to take the bottom, safer route to avoid falling into the -1 state by accident. This means even at s_{14}, the agent wants to go all the way left to go along the bottom path.

	1	2	3	4
1	$\downarrow$	$\leftarrow$	$\leftarrow$	$\leftarrow$
2	$\downarrow$	X	$\downarrow$	-1
3	$\rightarrow$	$\rightarrow$	$\rightarrow$	$+1$

Note that this policy is conservative, but not the most conservative. By choosing to go left in s_{14} and down in s_{23}, there is still a risk of falling into the -1 state.

5.4 When life is only slightly dreary

Problem: When $-0.0221 < R(s) \leq 0$, what does the optimal policy look like?

Solution: The optimal policy takes no risk.

This is the same policy as in the previous case except that the agent completely avoids falling into the -1 by accident, even though it might bump into a wall many times.

To achieve this, the agent chooses to go left instead of down in s_{23}, and up instead of left in s_{14}. In s_{23}, the agent may bump into the wall many times, but eventually it will go up or down. In s_{14}, the agent may bump into either wall many times, but eventually it will go left.

	1	2	3	4
1	$\downarrow$	$\leftarrow$	$\leftarrow$	$\uparrow$
2	$\downarrow$	X	$\leftarrow$	-1
3	$\rightarrow$	$\rightarrow$	$\rightarrow$	$+1$

In general, the better the conditions, the more conservative the policy.

5.5 When life is good

Problem: When $0 < R(s)$, what does the optimal policy look like?

Solution: The optimal policy avoids both goal states. The agent doesn't want to leave because every step has positive reward!

In most states, the agent can choose any direction and it will be safe from reaching a goal state. However, in s_{14}, s_{23}, and s_{33}, it needs to be careful.

	1	2	3	4
1	$\leftrightarrow\updownarrow\rightarrow$	$\leftrightarrow\updownarrow\rightarrow$	$\leftrightarrow\updownarrow\rightarrow$	$\uparrow$
2	$\leftrightarrow\updownarrow\rightarrow$	X	$\leftarrow$	-1
3	$\leftrightarrow\updownarrow\rightarrow$	$\leftrightarrow\updownarrow\rightarrow$	$\leftarrow$	$+1$

6 Determine the Optimal Policy Given $V^*(s)$

6.1 A 3×4 grid world problem

We will consider the same example as above and determine the optimal policy.

Example: A robot is in a 3×4 grid world. What should the robot do to maximize its rewards?

	1	2	3	4
1	Start			
2		X		-1
3				$+1$

- Let s_{ij} be the position in row i and column j.
- s_{11} is the initial state.
- There is a wall at s_{22}.
- s_{24} and s_{34} are goal states. The robot escapes the world at either goal state. The value in each goal state cell is its corresponding reward.
- $R(s) = -0.04, \forall s \neq s_{24}, s \neq s_{34}$.
- $\gamma = 1$ (in general, this discount factor should be between 0 and 1, but I'll use 1 to simplify our calculations).

6.2 The expected utility of a policy

We will start by defining some notation: a capital V denotes the expected utility of following some policy; a superscript indicates which specific policy.

- $V^\pi(s)$: expected utility of entering state s and following the policy π thereafter.
- $V^*(s)$: expected utility of entering state s and following the optimal policy π^* thereafter.

Note that V is also a function of s, the state the agent enters before following a given policy. Note also that V is not the one-time reward of entering s; we use $R(s)$ to denote the one-time reward. Instead, V is a long-term expected reward: the reward the agent expects to receive if it follows a given policy.

6.3 The values of $V^*(s)$

Here is one example of the values of $V^*(s)$.

Example:

	1	2	3	4
1	0.705	0.655	0.611	0.388
2	0.762	X	0.660	-1
3	0.812	0.868	0.918	$+1$

$V^*(s)$ for $\gamma = 1$ and $R(s) = -0.04, \forall s \neq s_{24}, s \neq s_{34}$.

In general, the expected utility is larger around the $+1$ state. You can see this most clearly along the path from the starting state that goes down and to the right towards the $+1$ state. The intuition here is that the closer we are to the $+1$ state, the fewer steps we need to take to reach the $+1$ state, and the less cost we need to incur to get there. Therefore, in the long run, we expect the total discounted reward accumulated up to the $+1$ state to be higher.

As for the other states, the expected utilities are lower: not only do we need to account for the distance to the $+1$ state, we also have to account for the effects of the -1 state.

For example, at s_{23} there is a sharp decrease from s_{33}. This big difference is due to s_{23} having a relatively large chance of falling into the -1 state if we follow the optimal policy, whereas s_{33} does not have an immediate chance of falling into the -1 state. From s_{33}, the optimal action is to go right, so we cannot possible fall into the -1 state.

As a second example, consider s_{14}. The expected utility there is the lowest among all the non-goal states since we are trapped in the corner. In trying to escape, it is very likely that we will fall into the -1 state.

The purpose of these example values is just to show you some numbers. You would become clear where these numbers come from once we cover the value iteration algorithm.

6.4 Calculate the optimal policy given $V^*(s)$

For the next step, let's assume we are given $V^*(s)$, the expected utility of following the optimal policy from each state. Given these numbers, we can determine the optimal policy in two steps.

For each state s, we will calculate the long-term expected utility of taking action a, $Q^*(s, a)$.

$$Q^*(s, a) = \sum_{s'} P(s'|s, a)V^*(s') \tag{1}$$

Taking action a may result in the agent moving to one of many next states s'. Thus, we need to sum over all the possible s'. The agent moves to each s' with transition probability $P(s'|s, a)$, yielding long-term expected utility $V^*(s')$.

The next step is quite simple. In state s, choose an action a that maximizes the expected utility.

$$\pi^*(s) = \arg\max_a Q^*(s, a) \tag{2}$$

This gives us the optimal policy, π^*.

Problem: What is the optimal action for state s_{13}?

(A) Up (B) Down (C) Left (D) Right

$$Q^*(s,a) = \sum_{s'} P(s'|s,a)V^*(s')$$
$$\pi(s) = \arg\max_a Q^*(s,a).$$

The values of $V^*(s)$ are given below.

	1	2	3	4
1	0.705	0.655	0.611	0.388
2	0.762	X	0.660	−1
3	0.812	0.868	0.918	+1

Solution: To solve this, we just need to calculate the Q-values for each possible action. This comes down to plugging in the right values.

If we try to go down, then we will travel down with probability 0.8, giving expected utility 0.660. However, with probability 0.1 we will travel to the left instead, giving expected utility 0.665, and with probability 0.1 we will travel to the right instead, giving expected utility 0.388. The complete calculation is:

$$Q(s_{13}, \text{down}) = 0.8 * 0.660 + 0.1 * 0.388 + 0.1 * 0.655 = 0.6323$$

An example where we might bump into a wall is going left. With probability 0.8, we go left with expected utility 0.655. With probability 0.1, we go down with expected utility 0.660. With probability 0.1, we go up and bump into the wall. We stay in the same state, with expected utility 0.611. Overall, the expected utility is:

$$Q(s_{13}, \text{left}) = 0.8 * 0.655 + 0.1 * 0.660 + 0.1 * 0.611 = 0.6511$$

The remaining two actions are similar:

$$Q(s_{13}, \text{right}) = 0.8 * 0.388 + 0.1 * 0.611 + 0.1 * 0.660 = 0.4375$$
$$Q(s_{13}, \text{up}) = 0.8 * 0.611 + 0.1 * 0.655 + 0.1 * 0.388 = 0.5931$$

For the optimal policy, we can simply look at the four actions and choose the one

giving the largest expected utility. In this case, that action is left:

$$\pi(s_{13}) = \text{left}.$$

The correct answer is (C).

If you want some extra practice questions, try this problem with some other states and compare your results with the examples given in the previous sections.