Markov decision process: basics

In today's story we introduce the basic concepts of MDP using a grid world example from the book *Artificial Intelligence A Modern Approach* by Stuart Russel and Peter Norvig. We will talk about other topics such as value iterations and policy iterations in future stories.

1 Our version of Pac-Man game

Here we have a 4×3 grid world somewhat like Pac-Man. We have a robot moving inside this 2D world playing a game.

Our robot can move four ways: up, down, left, and right, exactly like a Pac-Man. Another similarity with Pac-Man is that our world is also surrounded by non-passable walls. There are also non-passable walls inside the boundary represented by the black square.

The green square at the upper right corner represents a finish line. If we reach this green finish line, we win this game and earn a lot of points (+1 in our game).

In Pac-Man there are ghosts always trying to hurt you. In our game, we have a red square representing a pitfall. If we step into this red pitfall, we lose the game and incur a lot of penalty (-1 in our game).

Each time we step into a passable square that is neither a green finish line nor a red pitfall, we lose a small amount of points (-0.04 in our game). This means if we move randomly with the hope of eventually reaching the green finish line by luck, we lose 0.04 points each move we wandered and thus cumulatively lose a lot of points in this game. One way to think about this is the electric consumed by the robot. Each move of the robot costs us a certain amount of money.

In summary, when playing this game, our target is to keep away from the red pitfall and reach the green reward within the smallest number of moves. Each unnecessary move costs us an extra -0.04 points. We want to earn the +1 points as quickly as possible without paying too many -0.04 along the way.

2 Definition of MDP

In *Artificial Intelligence A Modern Approach*, MDP is defined as

> a sequential decision problem for a fully observable, stochastic environment with a Markovian transition model and additive rewards is called a Markov decision process, or MDP, and consists of a set of states (with an initial state s_0); a set ACTIONS(s) of actions in each state; a transition model $P(s'|s, a)$; and a reward function $R(s)$.

Let's break down some keywords in this definition using our example.

Agent refers to the robot or the Pac-Man or whatever we put inside this world that looks around, thinks about the pros and cons of possible moves, makes a decision of next move, and executes the move.

Environment refers to what the world looks like and the rules of that world. In our example, environment includes how big our world is, where are the walls placed, what happens if we bump into a wall, where is the finish line and if we reach there how much points we get, where is the pitfall and if we reach there how much points we lose, and so on.

State in MDP refers to the *state* of agent, not the environment. In MDP the environment is given and does not change. In contrast, our agent can change states from previous move to current move. In our example, our agent has 12 states which represent the 12 possible squares that our agent is in. Technically our agent is not able to be in the black square, but for simplicity we still count the black square as a state. In MDP we use s to denote state.

Action refers to the moves can be taken by our agent in each states. The set of possible actions is not necessarily the same for each state. In our example, we have four possible actions up, down, left, and right for every state. We use a to denote actions.

Reward refers to the points we receive when we move into a state. In our example, that is +1, -1, or -0.04 for different states. In MDP, reward is a function of states only and this function is denoted as $R(s)$. We receive the same reward when entering a certain state regardless of our previous state. For example, when $s = 3$, $r = R(s) = R(3) = +1$. No matter we move right into the green finish line from the square left to it or we move up into the green finish line from the square below it, we receive the same +1 reward.

Additive rewards means the reward from each moves are cumulative. In our example, our final points is the points received from the finish line or the pitfall and the points received from passable squares along the way added together. Not only the reward received in the final step but also all the small rewards received along the way matter. In MDP, we assume we can *add* these rewards together (as opposed to multiple together, for example).

Transitional model tells us the which state we move to if we choose a certain action at a certain state. This is usually represented as a table P. If we are at state s and we choose action a, then the probability of moving into next state s' is $P(s'|s, a)$. If we assume our world is *deterministic*, our agent always accurately executes the desired action. For example, if we are at $s = 10$ and we choose $a = \text{UP}$, we are guaranteed to move into a new state $s' = 6$. This is

written as

$$\begin{aligned}
P(s'=0|s=10,a=\text{UP})&=0\\
P(s'=1|s=10,a=\text{UP})&=0\\
P(s'=2|s=10,a=\text{UP})&=0\\
&\dots\\
P(s'=6|s=10,a=\text{UP})&=1\\
P(s'=7|s=10,a=\text{UP})&=0\\
P(s'=8|s=10,a=\text{UP})&=0\\
&\dots\\
P(s'=9|s=10,a=\text{UP})&=0\\
P(s'=10|s=10,a=\text{UP})&=0\\
P(s'=11|s=10,a=\text{UP})&=0
\end{aligned} \tag{1}$$

As a probability distribution, the summary of $P(s'|s=10,a=\text{UP})$ of all 12 s' always equals 1.

Stochastic means our world is not *deterministic* as we assumed above. From a current state, if we choose the same action, we are not guaranteed to move into the same next state. In our game, we can think of this as our robot somehow has some uncertainty. If it decides to go up in current state, it has a high possibility to actually go up. Nevertheless, there is still a small possibility, no matter how small it may be, that it goes wild and actually moves along other directions. In our game, following the example from the book, the probability of actually moving in the intended direction is 0.8. There is a 0.1 probability of moving 90 degree left to the intended direction and another 0.1 probability of moving 90 degrees right to the intended direction. For example, if we are at $s=10$ and we choose $a=\text{UP}$, we have a probability of 0.8 to move into the intended new state $s'=6$. We also have a probability of 0.1 arriving at $s'=9$ and a probability of 0.1 arriving at $s'=11$. This is written as

$$\begin{aligned}
P(s'=0|s=10,a=\text{UP})&=0\\
P(s'=1|s=10,a=\text{UP})&=0\\
P(s'=2|s=10,a=\text{UP})&=0\\
&\dots\\
P(s'=6|s=10,a=\text{UP})&=0.8\\
P(s'=7|s=10,a=\text{UP})&=0\\
P(s'=8|s=10,a=\text{UP})&=0\\
&\dots\\
P(s'=9|s=10,a=\text{UP})&=0.1\\
P(s'=10|s=10,a=\text{UP})&=0\\
P(s'=11|s=10,a=\text{UP})&=0.1
\end{aligned} \tag{2}$$

This is the transition model when $s=10$ and $a=\text{UP}$. Now for every pair

of current state s and action s, we can write out the probability of arriving at each new state like this. Combine all these together, we get the transition model P. We have 12 possible current state and 4 possible actions. Therefore, the dimension of our transitional model P is $12 \times 4 \times 12$, with a total of 576 probability values.

Fully observable means our agent somehow knows what the world looks like and where itself currently is in that world. In some sense, we can say our agent has a God's eye view. It has access to all the knowledge needed to find the optimal decision during each move. All the knowledge here refers to our reward function $R(s)$ and transitional model $P(s'|s, a)$

Sequential means our current situation is affected by previous decisions. By the same token, all future situations are affected by our current decision. For example, if we start from $s = 8$ and decide to move up as our first move, then we arrive at $s = 4$. There is no way we can reach $s = 10$ in next move. However if we choose move right as our first move when $s = 8$, then we arrive at $s = 9$. Now we can reach $s = 10$ by moving right. In other words, whether we can reach $s = 10$ in the second move depends on what we choose in the first move.

Markovian means our world is memoryless. This may seem in contrary to our definition of *sequential*, but they have completely different meanings. *Sequential* means whether we can reach $s = 10$ in our second move depends the choice we made in our first move. What does memoryless mean? It means no matter how we reached state $s = 10$, no matter whether we reached it from $s = 9$ or $s = 11$ or $s = 6$, *once we are in that state*, we always face the same set of possible actions. If we make a decision, what will happen does not depend on what happened in the past. In other words, it just means $P(s'|s, a)$ do not change during the entire game regardless of what you did in the game. When we reach $s = 10$ for the 100th time, we face the same choices of actions as if we reach $s = 10$ for the first time. If we choose a certain action at this state, the outcome always follows the same probability distribution. If a certain action is the optimal action for state $s = 10$ when we reach this state for the first time, then this action is always the optimal action for this state.

3 Reward function

As we explained above, reward function $R(s)$ only depends on state s. We use the following code to plot the reward function. In our csv file representing the map of our world, we use 0 as white passable squares, 1 as green squares, 2 as red squares, and 3 as black squares. As such, the reward of each type of squares is represented by the dictionary `reward`. The function `get_reward_table` generates the results of the reward function $R(s)$ for each s.

The reward function can then be plotted using the following function:

4 Transitional model

Let's recap the rules in our game. If we move into a wall, we are bounced back and stay where we were. If we want to move up, we have 0.8 probability to actually go up, 0.1 probability to go left, and 0.1 probability to go right. Similarly, if we want to move left, we have 0.8 probability to actually go left, 0.1 probability to go up, and 0.1 probability to go down. We can get our transitional model using the following code.

We can plot the transitional model using the following function:

5 Policy

Policy refers to the instructions of what action to take in each state. It is usually denoted as π and it is a function of s. For example, if $s = 2$ and $\pi(s) = \pi(2) =$ RIGHT, then whenever out agent is at state $s = 2$, it checks its policy and the policy says it should choose the action to go right. Therefore, the agent will execute the action of RIGHT every single time it is at state $s = 2$. Whether it can actually go to the square on the right is another story which depend on the probability in transition model. The point is, regardless of the outcome, as long as the policy says you should go right, our agent will choose to go right every single time.

5.1 Random policy

The starting point of looking at policy is to simply generate a random policy.

Using this random policy, we then execute it and check how many points we received in this game.

For example, if we have this random policy, we achieved a final points of The history of our state looks like this: [8, 9, 9, 9, 9, 10, 10, 6, 10, 11, 7]

Of course this is just a single run and we are facing a stochastic enviroment. So let us run it 10000 times and see how it does.