LECTURE 2

Markov Decision Processes, Part II

Markov Decision Processes

In the previous section, we defined a markov reward process that gives us a way to simulate an agent being moved through an environment, while the agent accumulates reward. However, you may have noticed that the agent has no say in what state it will be moved to. The agent has no way to take *actions*. The agent is just moved by the environment to a new state according to the environment's transition probabilities. By adding a *finite set of actions for each state* to the markov reward process, we get a *markov decision process.*

MDP = <States, **Actions**, Probability, Reward, Discount>

Now, when you are in a state, you have the option of *choosing from a set of actions, each of which will lead to a different probability distribution of transitions to the next set of states*. (See Example). This means we need to modify the definition of the state transition probability matrix.

The state transition probability matrix must now define a probability distribution of next states for each *state-action pair,* instead of just for each state. This means the magnitude of our probability matrix will increase by a scalar factor.

We will also redefine the reward process such that a reward is defined for each *state-action pair*, so that the environment provides a reward each time an agent takes an action.

MDP = <States, Actions, **Probability, Reward**, Discount>

- State transition probabilities are now defined per state and per action
- Reward function is now defined per state and per action

Note that while some environments have the same set of actions for each state, like our Pong example from the previous lecture, environments can also have different sets of actions for each state.

(See Slides for an example MDP)

Policy

To formally define an agent's behavior, or *decision-making process*, we will define the policy.

Policy. A policy is a distribution over actions given some set of states. This fully describes the behavior of an agent in some environment.

You can imagine this as a table, where actions are labelled along the columns, states are labelled along the rows, and each cell is the probability that that the agent will take the corresponding action at that state. An agent can then use this table to probabilistically determine which action to take from each state.

Take care not to confuse this *distribution over actions* with the distribution over future states defined by the state transition probability matrix described in the previous section. To put everything together, imagine the following scenario: an agent A has some behavior B and is in state S1; behavior B says that when A is in S1, A has a 90% likelihood of taking the walk action, and a 10% likelihood of taking the sit action. Randomly, A takes the walk action, which has a 70% chance of leading A to McDonalds and a 30% chance of leading A to Burger King.

Note that policies that are distributions are called stochastic policies, whereas policies with a one to one action-state mappings are deterministic policies. In this section, we will mainly refer to stochastic policies, the more general and powerful case- but we will use deterministic policies in later lectures about optimal policy for the sake of simplicity.

Optimal Policy. An optimal policy is a policy that would maximize the amount of expected cumulative reward an agent receives. Finding an optimal policy is equivalent to solving a reinforcement learning problem. More on this in the value-function and optimality sections below.

Policy Dependencies

Policies only depend on the current state, not the history (as a result of the markov property). This means that policies are *independent* of time.

Recovery of Markov Reward Processes and Markov Chains

Choosing a specific policy for a markov decision process gives us a markov reward process. More formally, if we remove the MDP's set of actions, and define its state probability transition matrix and reward function for a single policy, we will get a markov reward process.

Previously we could only imagine the transitions present in a markov reward process as the *environment randomly pushing* us around into different states. Now, we have an example of a markov reward process where the transitions are actually described by the policy from its “parent” markov decision process!

Also worth noting is that if we only look at the states of an MDP and define its transitions for a single policy, we will get back a markov chain.

These insights are not important for later sections, but are worthwhile to learn to gain a deeper understanding of the nature of markov decision processes.

Value Functions

In the previous section, we formally defined policies, which fully describe an agent’s behavior. Choosing a deterministic policy for an agent is akin to ‘setting your destiny in stone’. This means that once we choose a policy, we have some ability to look *into the future* of an agent, since we know which states the agent’s future actions will lead to, and which actions the agent will take at those new states (the definition of Policy).

Return. And if we look far enough ahead, we can even figure out all the *cumulative reward* the agent is expected to receive in its lifetime (episode).

This is of course only half-true. There will still be a degree of randomness involved after the agent takes an action, since the next state after a given action can be non-deterministic, since state transitions are probability distributions. And with a stochastic policy, there is even a degree of randomness in the actions an agent can take.

Expected Value. In spite of this randomness, we can still find the expected value (mean) of the cumulative reward of an agent, by summing the cumulative rewards across all possible paths and multiplying each path’s rewards by their probabilities.

We now have some notion of a function that can ascertain the *return (cumulative reward)* that is expected to be received by an agent.

State-Value Function

The state-value function is defined as the expected return from following a certain policy pi beginning at some state s.

Informally, this gives us a sense of how “good” it is to be in a certain state.

In the previous section, we described an iterative method of computing the state-value function for a given policy. In the next lecture, we will find a closed-form solution for the state-value function.

Action-Value Function

The action-value function is defined as the expected return from following a certain policy pi and taking a certain action a from some state s.

Informally, this gives us a sense of how "good" it is to take a certain action from some state.

There are a few more things to note about action-value and state-value functions:

- Action-value functions and state-value functions can easily be expressed in terms of one another. We will explore this in the next lecture.
- In practice, computing value functions can be infeasible, so we will spend much of the course exploring ways to *approximate* the value function using much faster methods.

Optimality

In the previous section, we defined the "goodness" of a state, given some policy. If we search through the space of all policies to find the policy that best maximizes the "goodness" of the initial state, we have an optimal policy.

Optimal State-Value Function. This is defined as the maximum state-value function over all policies (equivalent to the state-value function that uses the optimal policy).

Optimal Action-Value Function. This is defined as the maximum action-value function over all policies (equivalent to the action-value function that uses the optimal policy).

If we find any of these functions, we have solved our MDP. We can find these functions by either finding the optimal policy first, or somehow directly finding the optimal value function. We will talk about the former first.

Optimal Policy Theorem. There always exists an optimal policy that is better than every other policy. All optimal policies achieve optimal state-value and action-value function.

Some Policies Are Better Than Others

One way to find the best policy is by starting with some random "seed" policy and then iteratively find better and better policies than the previous one.

In order to find a better policy, we must first define what it means for one policy to be better than another.

Policy Ordering. A policy A is better than another policy B when the value expected from following policy A is greater than the value expected than following policy B at *every state* in the MDP.

Finding An Optimal Policy by Action-Value Function

We can also find an optimal policy by picking the actions that result in maximizing the optimal action-value function. This gives us a deterministic optimal policy for any MDP.

Thus, knowing the optimal action-value function is enough to find the optimal policy.

We will recover this section in further detail, and will talk more about finding optimal policies in later lectures.

Bellman Equation

We can define the value functions in terms of one another.

The state-value function can be defined as a function of the action-value function.
The action-value function can be defined as a function of the state-value function.

This implies that the state-value function can be defined recursively in terms of itself. The same goes with action-value functions.

The next lecture will cover Bellman expectation and optimality equations in depth.

NOTE: If you are a student of the Spring 2018 offering of this class at UMD, we will be covering this lecture material multiple times: once in a fast-paced theoretical lecture (for those familiar with the mathematics), and once in an example-based lecture.