2.2 Markov Decision Processes

Motivation

The previous section introduced the fundamental idea that robot actions have uncertain effects, and showed how these effects can be modeled using conditional probability distributions. This led to a sampling algorithm that can be used to simulate robot behavior. In this section, rather than simulating robot behavior, we consider the problem of planning robot behavior. In particular, we consider the problem of deriving optimal strategies that can be executed by a robot that is able to measure its state at each moment in time. To accomplish this, it will be necessary to introduce a few tools from probability theory, leading to the introduction of a Markov decision process, one of the most powerful computational models available for robots whose actions are subject to uncertainties during execution.

2.2.1 A Circular World

H	A	B
G		C
F	E	D

Figure 2.19: A simple building layout that contains eight rooms (rooms $A, B, \ldots H$) arranged in a cyclic corridor.

We start by giving an another example of an agent, inhabiting a world in which actions have probabilistic outcomes. Figure 2.19 shows a simple environment consisting of a circular corridor of rooms. Consider a robot in this environment that may choose one of two actions:

R : move in a counterclockwise direction

L : move in a clockwise direction

As before, we model the effects of these actions as probabilistic. When the robot executes an action, either R or L, it moves a random number of rooms D_k in the corresponding direction. More formally, D_k is the random variable that denotes the number of rooms traversed by the robot at time k. For this example, suppose that probability distribution for D_k is given by:

$$P(D_k = d_k) = \begin{cases} 0.25 & : \quad d_k = 1 \\ 0.5 & : \quad d_k = 2 \\ 0.25 & : \quad d_k = 3 \end{cases}$$

For example, if the robot is in state $X_k = A$ at time k and executes the action $A_k = L$, we have the following conditional probabilities for the state X_{k+1}

$$P(X_{k+1} = x_{k+1} \mid X_k = A, A_k = L) = \begin{cases} 0.25 & : \quad x_{k+1} = B \\ 0.5 & : \quad x_{k+1} = C \\ 0.25 & : \quad x_{k+1} = D \end{cases}$$

Proceeding in this manner for each possible state and both possible actions, we can construct the entire conditional probability table, such as shown in Figure 2.10.

2.2.2 Review of Concepts from Probability Theory

Three useful concepts from probability theory that we need below are the Markov property, the chain rule, and the notion of taking expectations.

The **Markov property** formalizes the idea from the last chapter that in a Markov chain, anything that happened prior to time k is irrelevant when determining the outcome X_{k+1}. This can be written in general terms as:

$$P(X_{K+1} = x_{k+1} \mid X_k = x_k, \ldots X_0 = x_0) = P(X_{K+1} = x_{k+1} \mid X_k = x_k)$$

Simply stated, if a system satisfies the Markov property, future states depend only on the current state and action.

We also need the notion of the **chain rule**. This rule says that any *joint probability* $P(X, Y)$ can be rewritten as the product of a prior probability and a conditional:

$$P(X, Y) = P(X)P(Y|X)$$

The chain rule is unaffected if we condition on other information Z:

$$P(X, Y|Z) = P(X|Z)P(Y|X, Z).$$

The condition simply applies to both factors above.

Finally, the notion of **expectation** gives a prediction of the average behavior of functions of probabilistic events. More precisely, if X is a discrete random variable that takes on values in the set $\{x_1, \ldots, x_n\}$, the *expected value* of a function $f(X)$ of X is defined as

$$E_{P(X)}[f(X)] = \sum_{i=0}^{n} f(x_i)P(X = x_i) \tag{2.1}$$

A simple example to illustrate this concept is that of rolling a single fair, six-sided die. In this case, we have six possible events E_i, and let $f(E_i) = i$ be the number of spots on the top face of the die after it is cast. If the die is fair, we have $P(E_i) = 1/6$ for each $i \in \{1, 2, 3, 4, 5, 6\}$. Applying (2.1), we arrive immediately to

$$E_{P(E_i)}[f(E_i)] = \sum_{i=0}^{n} \frac{i}{6} = 3.5.$$

At first glance this may seem odd; certainly no one literally *expects* to roll a die and observe 3.5 dots on the top face.

In probability theory, the meaning of *expectation* is not the usual meaning regarding what we expect to see as the result of one experience. Rather, the concept of expectation in probability theory is refers to what we expect for the *average* behavior when we observe many trials. For the example of rolling a die, let $\overline{f(X)}_N$ denote the average value obtained from N rolls of the die. We expect that this average will be close to 3.5, particularly as N

increases. In probability theory, this idea is captured by the law of large numbers, which, in its strong form essentially says that

$$\lim_{N\to\infty} \overline{f(X)}_N = E_{P(X)}[f(X)]$$

.

2.2.3 Computing the Probability of a Sequence of States

While sampling gives us a way to simulate, we need a more direct way to compute the probability distribution over future states. Given the world model above, we can apply the sampling algorithm described in Section 2.1.7 to generate typical sequences of states for specific action sequences. Unfortunately, this forward simulation algorithm does not give us a way to determine the probability that a given sequence will occur, and therefore it is not immediately useful for the problem planning an appropriate set of actions to achieve a goal.

To develop a method for computing the probability that a given sequence of actions will lead to a particular sequence of states, let us start with a simple two-step example. In particular, consider the case in which we begin at time $k = 0$ in state $X_0 = A$ and execute the action sequence $A_0 = L, A_1 = L$. What is the probability that the robot will visit a sequence of rooms $X_1 = B, X_2 = C$? The first step to solving this problem is constructing a precise description of the probability we wish to compute. In this case, we are *given* three pieces of information: $X_0 = A, A_0 = L, A_1 = L$; we wish to determine the probability of the outcome $X_1 = B, X_2 = C$. Therefore, the probability of interest is exactly expressed as

$$P(X_1 = B, X_2 = C \mid X_0 = A, A_0 = L, A_1 = L).$$

Given the chain rule, we can find the probability of a two-step sequence. In our example, $X \triangleq \{X_1 = B\}$, $Y \triangleq \{X_2 = C\}$, and and $Z \triangleq \{X_0 = A, A_0 = L, A_1 = L\}$. Substituting that into the conditional chain rule (2.2.2) leads to

$$\begin{aligned} &P(X_1 = B, X_2 = C \mid X_0 = A, A_0 = L, A_1 = L) = \\ &\quad P(X_1 = B \mid X_0 = A, A_0 = L, A_1 = L) P(X_2 = C \mid X_1 = B, X_0 = A, A_0 = L, A_1 = L) \end{aligned}$$

The expression on the right hand side can be simplified in two ways. First, the value of X_1 clearly does not depend upon the action take at time $k = 2$, and therefore

$$P(X_1 = B \mid X_0 = A, A_0 = L, A_1 = L) = P(X_1 = B \mid X_0 = A, A_0 = L)$$

Now consider the term $P(X_2 = C \mid X_1 = B, X_0 = A, A_0 = L, A_1 = L)$. For the model of uncertainty that we have defined above, in this example the results of the robot's action at stage k depends only on the state X_k and the action A_k, which is simply an instance of the Markov property. If we apply these observations to our example, we arrive to

$$\begin{aligned} &P(X_1 = B, X_2 = C \mid X_0 = A, A_0 = L, A_1 = L) = \\ &\qquad P(X_1 = B \mid X_0 = A, A_0 = L) P(X_2 = C \mid X_1 = B, A_1 = L) \end{aligned}$$

and we can easily determine that $P(X_1 = B, X_2 = C \mid X_0 = A, A_0 = L, A_1 = L) = 0.0625$ using the conditional probabilities for our actions.

We can extend the ideas above to evaluate the probability distribution on sequences of arbitrary length. This gives us the following general formula:

$$P(X_1 = x_1, \ldots, X_{k+1} = x_{k+1} \mid X_0 = x_0, A_0 = a_0, \ldots, A_{k+1} = a_{k+1}) = \\ P(X_1 = x_1 \mid X_0 = x_0, A_0 = a_0) \ldots P(X_{k+1} = x_{k+1} \mid X_k = x_k, A_k = a_k)$$

2.2.4 Reward and Expected h-stage Return

Now that we are able to compute the probability that a given sequence of actions will produce a particular sequence of states, let us turn our attention to the more interesting problem of choosing an optimal set of actions with respect to some desired performance objectives.

In many robotics problems, we specify performance objectives by assigning a specific **reward** to each state. Mathematically, if our set of states is denoted by $\mathcal{S}$, then a reward function is a mapping, $R : \mathcal{S} \to \mathbb{R}$, that specifies the reward value for each state. For our example above, suppose that our main objective is to keep the robot fully charged, and that the only charging station is in Room E. To capture this objective, we might assign a positive reward, say $+1$ to room E, and a small negative reward, say -0.2, to all other rooms.

Since we are generally interested in the accumulated reward over some period of execution, let us define the **h-stage return** for a sequence of states as

$$r_h(x_0, \ldots, x_h) = \sum_{i=0}^{h} R(x_i)$$

in which h denotes the *horizon*. For the room sequence A, B, C, the 2-stage reward can be easily computed to be $r_2(A, B, C) = R(A) + R(B) + R(C) = -0.6$.

However, we will need a better way to evaluate the return associated to a random sequence of states. In this example, it is fairly easy to see that given this reward structure there is really no better action sequence than L, L; any other actions will definitely move the robot away from room E. And yet, this return is the worst possible return we could have for a sequence of two actions. Clearly merely using the h-stage return for a sequence of actions is not the right way to evaluate the quality of the action sequence. What went wrong? The problem with using the deterministic sequence A, B, C to evaluate the action sequence L, L is our actions are *not* deterministic.

We need to take into account probabilities of outcomes. In fact, as we have seen above, the probability for this outcome is 0.0625, which means that this situation is not particularly likely. If we evaluate the probabilities and 2-stage returns for each possible sequence of states given actions L, L and initial state $X_0 = A$, we obtain

Sequence	Probability	2-stage return
A, B, C	0.0625	-0.6
A, B, D	0.125	-0.6
A, B, E	0.0625	0.6
A, C, D	0.0625	-0.6
A, C, E	0.25	0.6
A, C, F	0.0625	-0.6
A, D, E	0.0625	0.6
A, D, F	0.125	-0.6
A, D, G	0.0625	-0.6

and for any other sequence, the probability is zero. From this table, we can see that the sequence with the highest probability is A, C, E, which has the maximum return possible. Nevertheless, there are quite a few possibilities that lead to negative reward.

We can immediately apply this concept to our example, and compute the **expected 2-stage return**

$$E_{P(X_1,X_2)}[r_2(A, X_1, X_2)] = R(A) + \sum P(X_1, X_2)\{R(X_1) + R(X_2)\} = -0.075$$

in which the sum is taken over possible values for X_1 and X_2 as given in the table above. This result may not seem so pleasing (a negative return is rarely the desired result), but when compared with all other options, it is the best. To see this, compute $E[r_2(A, X_1, X_2)]$ for all other 2-action sequences: (R, R), (R, L), and (L, R). For each of these, $E[r_2(A, X_1, X_2)] = -0.6$, so the choice of (L, L) is indeed the best available action sequence from the initial state $X_0 = A$.

2.2.5 Discounted Rewards

Suppose we wish to maximize the expected return over the lifetime of a robot. In this case, the value of h could become quite large, even allowing $h \to \infty$, which corresponds to the so-called *infinite horizon* problem. If we merely apply the definition of expected h-stage return, there are two disadvantages that may arise. First, in the infinite horizon case, the expected return may diverge to infinity, making it impossible to discriminate between different behaviors. Second, it may be detrimental to give equal weight to rewards that may occur far into the future when planning near-term actions. One way to deal with these problems is to introduce a discounting factor, $\gamma \in (0, 1)$, to obtain the *discounted* h-stage return

$$r_h(x_0, \ldots, x_h) = \sum_{k=0}^{h} \gamma^k R(x_k)$$

Note that if there is a maximum value for the reward, i.e., if we have $R(x_i) \leq R_{\max}$ for all i, we can derive a bound on the discounted return as

$$\lim_{h\to\infty} r_h = \sum_{k=0}^{\infty} \gamma^k R(x_k) \leq \sum_{k=0}^{\infty} \gamma^k R_{\max} = \frac{R_{\max}}{1-\gamma}$$

Therefore, even in the infinite horizon case, the expected discounted return is finite. In general, infinite horizon problems are easier to solve, and they provide a good approximation for finite horizon problems, since the future rewards are severely discounted as k becomes large.

2.2.6 Policies

In real life, robots or agents take action based on the state they find themselves in. For a *given* sequence of actions, say $a_0, \ldots a_h$ we can compute the expected return as

$$E[r_h] = E\left[\sum_{k=0}^{h} \gamma^k R(X_k) \mid a_0, \ldots a_h\right] \tag{2.2}$$

In principle we could use (2.2) to directly compute the optimal sequence of actions: enumerate every sequence of actions and choose the sequence that maximizes (2.2). If the robot were going to execute the action sequence without ever making any observations about the world state, this seem to be a reasonable -if computationally heavy- approach. However, real robots rarely operate for long periods of time without using their sensors to observe the state of the world. If the robot is able to determine its state at time k, it makes little sense to commit to an action sequence that was computed before the robot began operating in the world.

To capture this, we can formally define the notion of a *policy*. As we know, at time k the future behavior of a Markov process depends on the current state X_k and the chosen action a_k, but the states for time $t < k$. Therefore, it makes sense that the robot should update its decisions depending on its state. This idea leads to the concept of a **policy**, $\pi : \mathcal{S} \to \mathcal{A}$, in which $\mathcal{S}$ is the set of states, and $\mathcal{A}$ is the set of possible actions. In short, a policy is merely a mapping from the current state, X_k to the action a_k to be executed.

In order to evaluate the quality of a policy, it is necessary to update the idea of h-stage return to allow for state-dependent action selection. We will denote the expected return for executing policy π starting in state x as

$$V^{\pi}(x) = E\left[\sum_{k=0}^{\infty} \gamma^k R(X_k) \mid \pi, X_0 = x\right]$$

We can a derive recursive formulation for the value function $V^{\pi}(x)$ that lends itself to efficient computation. It is not at all apparent how one could actually compute $V^{\pi}(x)$ from the expression (2.2.6) above. Luckily, we have can take some terms out of the expectation

and rewrite,

$$\begin{aligned} V^{\pi}(x) &= E\left[\sum_{k=0}^{\infty}\gamma^k R(X_k) \mid \pi, X_0 = x\right] \\ &= E\left[R(X_0) + \sum_{k=1}^{\infty}\gamma^k R(X_k) \mid \pi, X_0 = x\right] \\ &= R(x) + \gamma E\left[\sum_{k=1}^{\infty}\gamma^{k-1} R(X_k) \mid \pi\right] \\ &= R(x) + \gamma E\left[\sum_{j=0}^{\infty}\gamma^{j} R(X_{j+1}) \mid \pi\right] \end{aligned}$$

in which we have used the substitution $j = k-1, j+1 = k$ to obtain the last line. Now, notice that the expectation in this last line is nothing more than the expected value of $V^{\pi}(X')$ under policy π, but in this case the expectation is taken with respect to the random variable X', which denotes the next state when π is executed in state x, which can be written as

$$V^{\pi}(x) = R(x) + \gamma E\left[V^{\pi}(X') \mid \pi\right]$$

If we now directly apply the conditional expectation version of (2.1), we obtain

$$V^{\pi}(x) = R(x) + \gamma \sum_{x' \in \mathcal{S}} P(X' = x' \mid X = x, a = \pi(x)) V^{\pi}(x') \tag{2.3}$$

That is $V^{\pi}(x)$ is equal to the sum of (a) the reward in the current state, $R(x)$, and (b) the discounted expected value of $V^{\pi}(X')$, where X' denotes the (random) state obtained by applying the action $\pi(x)$ in state x.

2.2.7 The Value Function

Let us denote by π^* the **optimal policy** with respect to (2.3), i.e.

$$\pi^* = \arg\max_{\pi} V^{\pi}(x)$$

The function $V^{\pi^*}(x)$ gives the maximum possible expected return from state x. In this case, we denote the expected return as $V^* \triangleq V^{\pi^*}$, and we refer to V^* as the **value function**. If we have the value function for a problem, the optimal action at time k is easily computed using

$$\pi^*(x) = \arg\max_{a \in \mathcal{A}} \sum_{x' \in \mathcal{S}} P(X' = x' \mid X = x, a = \pi(x)) V^*(x') \tag{2.4}$$

For this reason, estimating the value function has long been a topic of research in the fields artificial intelligence and machine learning.

The value function V^* can also be expressed in a compact recursive form. Using (2.3) and (2.4) we can obtain

$$V^*(x) = R(x) + \gamma \max_{a \in \mathcal{A}} \sum_{x' \in \mathcal{S}} P(X' = x' \mid X = x, a = \pi(x)) V^*(x') \tag{2.5}$$

This is the famous **Bellman Equation**, one of the most important equations in fields including artificial intelligence, machine learning, stochastic optimal control.

2.2.8 Value Iteration

Besides its elegance, the Bellman Equation (2.5) also lends itself to computation using an iterative approximation scheme known as **value iteration**. The idea behind value iteratin is to compute a sequence of estimates V^l such that $V^l \to V^*$. The heart of the value iteration algorithm is the update equation

$$V^{l+1} = R(x) + \gamma \max_{a \in \mathcal{A}} \sum_{x' \in \mathcal{S}} P(X' = x' \mid X = x, a = \pi(x)) V^l(x')$$

This equation is known as a **Bellman update**. It can be shown that if $V^l = V^*$ for some value of l, then $V^{l'} = V^*$ for all $l' > l$, and that value iteration converges to the unique solution of the Bellman equation.

2.2.9 A Formal Description of MDPs

The development above has been largely driven by examples. Here, we collect the various concepts that have been introduced above to provide a formal definition of Markov decision processes.

A **Markov decision process** or MDP is defined by the following:

$\mathcal{S}$	the set of states
$\mathcal{A}$	the set of actions
$P : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$	transition probabilities
$R : \mathcal{S} \to \mathbb{R}$	reward function
$\gamma \in (0, 1)$	discount factor