7 - MDPs and Value Iteration

知识点 & 题目

CLASSICAL PLANNING	MDPs
Set of states S	Set of states S
Initial state s_0	Initial state s_0
Actions $A(s)$	Actions $A(s)$
Transition function $s' = f(a, s)$	Transition probabilities $P_a(s' \mid s)$
Goals $S_G \subseteq S$	Reward function $r(s, a, s')$ positive or negative of transitioning from state s to state s'
Action costs $c(a, s)$	Discount factor $0 \leq \gamma \leq 1$

Policies: Deterministic vs. Stochastic

Deterministic: pi(s) -> A. Given state s, the policy pi is a function that maps states to actions.

- It specifies which action to choose in every possible state.
- Thus, if we are in state s, our agent should choose the action defined by π(s).

Stochastic: pi(s, a) S * A -> R. Given a state s and action a, returns the probability that action a will be selected in s. Intuitively, π(s, a) specifies the probability that action a should be executed in state s.

Optimal solutions to MDPs: The Bellman Equation (Discounted-Reward MDPs)

$$V(s) = \max_{a \in A(s)} \sum_{s' \in S} P_a(s' | s) \; [r(s, a, s') + \gamma \; V(s')]$$

Solving MDPs with Dynamic Programming: Value Iteration

$$V_{i+1}(s) := \max_{a \in A(s)} \sum_{s' \in S} P_a(s' | s) \; [r(s, a, s') + \gamma \; V_i(s')]$$

Algorithm – Value iteration

Input: MDP $M = \langle S, s_0, A, P_a(s' \mid s), r(s,a,s') \rangle$

Output: Value function V

Set V to arbitrary value function; e.g., $V(s) = 0$ for all s

Repeat

$\quad \Delta \leftarrow 0$

$\quad$ For each $s \in S$

$$\underbrace{V'(s) \leftarrow \max_{a \in A(s)} \sum_{s' \in S} P_a(s' \mid s)\ [r(s,a,s') + \gamma\, V(s')]}_{\text{Bellman equation}}$$

$\quad\quad \Delta \leftarrow \max(\Delta, |V'(s) - V(s)|)$

$\quad V \leftarrow V'$

Until $\Delta \leq \theta$

O(|S|^2 |A| n) L7 P16

Policy extraction:

$$\pi(s) = \underset{a \in A(s)}{\operatorname{argmax}} \sum_{s' \in S} P_a(s'|s)\ [r(s,a,s') + \gamma\ V(s')]$$

Summary

- We covered Markov Decision Processes (MDPs). They differ from classical planning in that **actions can have more than one possible outcome**. Each outcome has an associated probability.
- The optimal policy can be computed through value iteration, which is based on dynamic programming. Specifically, it uses the Bellman equations to iteratively improve on a non-optimal solution.
- We looked at how to extract policies from value functions derived by value iteration.

题目

Quiz

Question 1

1 / 1 pts

You want to buy a new guitar. There are three options: Maton, Fender, and Martin; but you are worried about the dreaded 'buyers remorse'.

If you buy a Maton (your dream acoustic guitar!), you think there is an 80% chance that you will feel +100 better (your reward/return); but because it is so expensive, there is a 20% chance of buyer's remorse, which will make you feel -100 (that's a *negative* reward)

If you buy a Fender, you think there is an 70% chance that you will feel +70 better; and a 30% you feel -100.

If you buy a Martin, you think there is an 60% chance that you will feel +100 better; a 20% you feel -40; and a 20% that you can sell it to your idiot brother whose name is Martin and buys anything that bears his name, which makes you slightly happy (feel +10)

What is the expected return of the Maton?

60

60 (with margin: 0)

The expected return is calculated as:

0.8 x 100 + 0.2 x -100
= 80 - 20
= 60

Question 2

1 / 1 pts

What is the expected return of the Fender?

19

19 (with margin: 0)

The expected return is calculated as:

0.7 x 70 + 0.3 * -100
= 49 - 30
= 19

Question 3

1 / 1 pts

What is the expected return of the Martin?

54

54 (with margin: 0)

The expected return is calculated as:

0.6 x 100 + 0.2 x -40 + 0.2 * 10
= 60 - 8 + 2
= 54

Question 5

3 / 3 pts

Consider the following abstract MDP with three states, s, t, and u and two actions a and b.

The transition probabilities are as follows:

P_a (t | s) = 0.6
P_a (s | s) = 0.4
P_b (u | s) = 1.0
P_b (u | t) = 1.0

Any probabilities not listed above have probability of 0.

The reward function has the following:

r(s, a, t) = 2
r(s, b, u) = 5
r(t, b, u) = 5

Assuming V(s) = V(t) = V(u) = 0, and a discount factor of 0.9, calculate the V for the first iteration to one decimal place.

V(s) = 5

V(t) = 5

V(u) = 0

For V(s):

Q(s, a) = P_a (t | s) * [r(s, a, t) + γV(t)] + P_a (s | s) * [r(s, a, s) + γV(s)]
= 0.6 * [2 + 0.9*0] + 0.4 * [0 + 0.9*0]
= 1.2
Q(s, b) = P_b (u | s) * [r(s, b, u) + γV(u)]
= 1.0 * [5 + 0.9*0]
= 5
max((Q(s,a), Q(s,b)) = 5
Therefore, V(s) = 5

For V(t):

Q(t, b) = P_b (u | t) * [r(t, b, u) + γV(u)]
= 1.0 * [5 + 0.9*0]
= 5
Action b is the only action, therefore V(t) = 5

For V(u):

There are no actions from u, so the value is just 0.

Question 6

1 / 1 pts

Take the same example from the previous question. Assume that we run value iteration and terminate after some fixed number of iterations. The resulting value function is:

V(s) = 12
V(t) = 10
V(u) = 0

In state s, which action should be taken: a or b?

- (selected) a
- b

For policy extraction, we just calculate the expected reward of each action:

Q(s,a) = P_a (t | s) * [r(s, a, t) + γV(t)] + P_a (s | s) * [r(s, a, s) + γV(s)]
= 0.6 * [2 + 0.9*10] + 0.4 * [0 + 0.9*12]
= 6.6 + 4.32
= 10.92

Q(s, b) = P_b (u | s) * [r(s, b, u) + γV(u)]
= 1.0 * [5 + 0.9*0]
= 5

The argmax of these two is action a, so this is what we select.