Artificial Intelligence Week 11: Revision

Edmond Ho
University of Glasgow, Glasgow, UK

December 2, 2024

Markov Decision Processes (MDP) and Bellman equations

Recap: Bellman (optimaility) equations

The utility of a state is the immediate reward for that state plus the expected discounted sum of rewards of the next state,

$$U^*(s) = R(s) + \gamma \, max_a \sum_{s'} p(s'|s,a) U^*(s')$$

This results in a set of coupled (non-linear) equations which we would need to solve to find the utilities to allow us to do the planning.
Note: We usually leave out the * notation and simply use U(s).

Sample Question 1

Sometimes MDPs are formulated with a reward function $R(s, a)$ that depends on the action taken or with a reward function $R(s, a, s')$ that also depends on the outcome state.

Task: Write the Bellman equations for these formulations.

Sample Question 1

Sometimes MDPs are formulated with a reward function $R(s, a)$ that depends on the action taken or with a reward function $R(s, a, s')$ that also depends on the outcome state.

Task: Write the Bellman equations for these formulations.

The key here is to get the max and summation in the right place. For $R(s, a)$ we have

$$U(s) = \max_a [R(s, a) + \gamma \sum_{s'} p(s'|s, a) U(s')]$$

and for $R(s, a, s')$ we have

$$U(s) = \max_a \sum_{s'} p(s'|s, a)[R(s, a, s') + \gamma U(s')]$$

Sample Question 2a

Consider an undiscounted MDP having three states, $(1, 2, 3)$, with rewards $-1, -2, 0$, respectively. State 3 is a terminal state. In states 1 and 2 there are two possible actions: *a* and *b*. The transition model is as follows:

- In state 1, action *a* moves the agent to state 2 with probability 0.8 and makes the agent stay put with probability 0.2.
- In state 2, action *a* moves the agent to state 1 with probability 0.8 and makes the agent stay put with probability 0.2.
- In either state 1 or state 2, action *b* moves the agent to state 3 with probability 0.1 and makes the agent stay put with probability 0.9.

What can be determined qualitatively about the optimal policy in states 1 and 2?

Sample Question 2a

What can be determined qualitatively about the optimal policy in states 1 and 2?

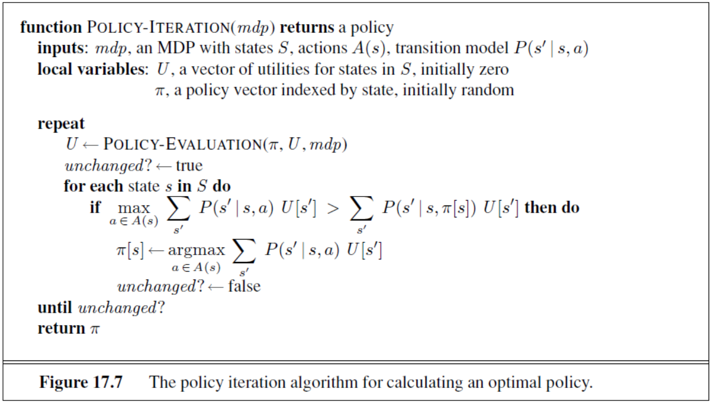
Intuitively, the agent wants to get to state 3 as soon as possible, because it will pay a cost for each time step it spends in states 1 and 2. However, the only action that reaches state 3 (action b) succeeds with low probability, so the agent should minimize the cost it incurs while trying to reach the terminal state.

This suggests that the agent should definitely try action b in state 1; in state 2, it might be better to try action a to get to state 1 (which is the better place to wait for admission to state 3), rather than aiming directly for state 3. The decision in state 2 involves a numerical tradeoff.

Sample Question 2b

Apply policy iteration, showing each step in full, to determine the optimal policy and the values of states 1 and 2. Assume that the initial policy has action *b* in both states.

(... a quick recap...)



```
function POLICY-ITERATION($mdp$) returns a policy
  inputs: $mdp$, an MDP with states $S$, actions $A(s)$, transition model $P(s' \mid s, a)$
  local variables: $U$, a vector of utilities for states in $S$, initially zero
                   $\pi$, a policy vector indexed by state, initially random

  repeat
     $U \leftarrow$ POLICY-EVALUATION($\pi, U, mdp$)
     $unchanged? \leftarrow$ true
     for each state $s$ in $S$ do
         if $\max_{a \in A(s)} \sum_{s'} P(s' \mid s, a)\ U[s'] > \sum_{s'} P(s' \mid s, \pi[s])\ U[s']$ then do
             $\pi[s] \leftarrow \operatorname{argmax}_{a \in A(s)} \sum_{s'} P(s' \mid s, a)\ U[s']$
             $unchanged? \leftarrow$ false
  until $unchanged?$
  return $\pi$
```

Figure 17.7 The policy iteration algorithm for calculating an optimal policy.

Sample Question 2b

Apply policy iteration, showing each step in full, to determine the optimal policy and the values of states 1 and 2. Assume that the initial policy has action *b* in both states.

Initialization: $U \leftarrow \langle -1, -2, 0\rangle$, $P \leftarrow \langle b, b\rangle$
Value determination:

$$u_1 = -1 + 0.1u_3 + 0.9u_1$$

$$u_2 = -2 + 0.1u_3 + 0.9u_2$$

$$u_3 = 0$$

That is, $u_1 = -1.9$ and $u_2 = -3.8$.
Policy update: In state 1,

$$\sum_j P(s_1, a, s_j)u_j = 0.8 \times -3.8 + 0.2 \times -1.9 = -3.42$$

while

$$\sum_j P(s_1, b, s_j)u_j = 0.1 \times 0 + 0.9 \times -1.9 = -1.71$$

so action b is still preferred for state 1.

Policy update: In state 2,

$$\sum_j P(s_2, a, s_j)u_j = 0.8 \times -1.9 + 0.2 \times -3.8 = -2.28$$

while

$$\sum_j P(s_2, b, s_j)u_j = 0.1 \times 0 + 0.9 \times -3.8 = -3.42$$

so action a is preferred for state 2. We set *unchanged?* to *false* and proceed.

(remember we updated $U \leftarrow \langle -1.9, -3.8, 0\rangle$, $P \leftarrow \langle b, a\rangle$, see next page for more...)

Sample Question 2b (cont.)

Value determination:

$$u_1 = -1.9 + 0.1u_3 + 0.9u_1$$

$$u_2 = -3.8 + 0.8u_1 + 0.2u_3$$

$$u_3 = 0$$

Now $u_1 = -3.61$ and $u_2 = -5.32$.
Policy update: In state 1,

$$\sum_j P(s_1, a, s_j)u_j = 0.8 \times -5.32 + 0.2 \times -3.61 = -4.978$$

while

$$\sum_j P(s_1, b, s_j)u_j = 0.1 \times 0 + 0.9 \times -3.61 = -3.249$$

so action b is still preferred for state 1.

Policy update: In state 2,

$$\sum_j P(s_2, a, s_j)u_j = 0.8 \times -3.61 + 0.2 \times -5.32 = -3.952$$

while

$$\sum_j P(s_2, b, s_j)u_j = 0.1 \times 0 + 0.9 \times -5.32 = -4.788$$

so action a is still preferred for state 2. *unchanged?* remains *true* and we terminate.

Note that the resulting policy matches our intuition: when in state 2, try to move to state 1, and when in state 1, try to move to state 3.

Markov Decision Processes (MDP) and Reinforcement Learning

Sample Question 3 - Past Exam Paper

In this simple 4 $\times$ 3 grid problem, the agent is trying to move from the initial state at grid cell (3,0) (i.e. the bottom-right cell) to the goal state (0,1). There are two terminal states (0,0) and (0,1) and the rewards are -1 and +1, respectively. The rest of the states have a reward of -0.04 each as shown in Table 1.

The agent executes a set of trials in the environment using its policy (Table 3). With a discounting factor γ=0.9, the utility values for the states are estimated based on the observations (Table 2). The agent takes action (i.e. up, down, left, or right) based on the policy.

-0.04	-0.04	-0.04	-0.04
+1	-0.04	-0.04	-0.04
-1	-0.04	-0.04	-0.04

Table: 1. The rewards for the states.

0.790	0.639	0.503	0.382
1.0	0.742	0.554	0.406
-1.0	0.410	0.403	0.315

Table: 2. Utility values for the states.

↓	←	←	←
·	←	←	←
·	↑	↑	←

Table: 3. The policy.

Sample Question 3a [6 marks]

Given the problem stated above, Temporal-difference (TD) learning can be used to update the utility values shown in Table 2.

Explain why the utilities updated using TD optimal are not necessarily optimal based on the fixed policy in Table 3.

Suggest a solution to estimate the optimal utilities and explain what additional information will be needed.

Sample Question 3a (cont.)

- Utilities calculated from temporal differencing (TD) based on a fixed policy are not necessarily optimal.
- In this problem, with a fixed policy, TD learning updates its estimates based on the policy being followed. Here, it is unclear if the policy shown in Table 8 is the optimal one or not. As a result, the utilities updated using TD are not necessarily optimal. [2]
- **Policy iteration** algorithm can be used to find the optimal policy. [1]
- The policy iteration algorithm alternates the following two steps, beginning from some initial policy:
  - Policy evaluation: given a policy, calculate the utility of each state if the given policy were to be executed [1]
  - Policy improvement: Calculate a new MEU policy, using one-step look-ahead based on the current utilities [1]
- In order to use the Policy iteration algorithm in this problem, we will need the transition probabilities [1].

Marking: Partial marks will be given as shown in the marks breakdown above. Other suggested methods will be considered and marks will be given accordingly.

Sample Question 3b [4 marks]

Although executing a set of trials in the environment enables us to train the agent to move from the initial state to the goal as shown above, the computation costs can be high.

Suggest how the value of a state or an action can be approximated. Explain your answer.

Sample Question 3b (cont.)

Function Approximation can be employed. [1] In this grid world problem, the 2D coordinates for each state can be used as the input to train an approximation function:

$$U(s) = w_0 + w_1 a_0 + w_2 a_1 \qquad [1]$$

where a_0 and a_1 are the x and y coordinates of the grid cell (i.e. state), and $w = [w_0, w_1, w_2]$ are the eights to be learned to approximate the utility function. [1]

Based on the observations, we can fit the data into this approximation function to find the weights w using optimisation such as gradient descent. [1]

Marking: There are other approaches using function approximation and marks will be given to reasonable answers.