Reinforcement learning

Yang Fei

August 24, 2024

1 Bellman equation

First, we need to explain the difference between the Markov process and the Markov decision process:

1. **Markov process:** The Markov process mainly describes the conditional probability of the system state propagating from the current status of S_t to the upcoming future stage of S_{t+1}.

2. **Markov decision process:** An additional word of 'decision' is added, indicating that there should be an action $a(k)$ that manipulates the system state. Of course, the result is not determined, and there is a possibility of getting S_{t+1} when you make action A_t when the system state is S_t.

Then a single step of the Markov decision process is

$$S_t \xrightarrow{A_t} \{R_{t+1}, S_{t+1}\} \tag{1.1}$$

where S_t is the system state at time step t, A_t is the action made and R_{t+1} is the action's reward. Some may find R_{t+1} confusing because the action is made at t. One way to understand it is that your action cannot lead to a deterministic result. Hence, it is the next step's state that determines your reward for making A_t.

There are some important items to mention:

1. $\pi(A_t = a|S_t = s)$: the possibility of making the action of $A_t = a$ when the current state is $S_t = s$. (Policy)

2. $p(R_{t+1} = r|A_t = a, S_t = s)$: the possibility of obtaining the reward of $R_{t+1} = r$ when the current state is $S_t = s$ and the action to make is $A_t = a$. (Reward)

3. $p(S_{t+1} = s'|A_t = a, S_t = s)$: the possibility of having a new state of $S_{t+1} = s'$ when the current state is $S_t = s$ and the action to make is $A_t = a$. (State transition or model)

Define $\gamma \in [0, 1)$ as the discounted rate, then the important discounted return is described as

$$G_t = R_{t+1} + \gamma R_{t+2} + \cdots = \sum_{i=1}^{\infty} \gamma^{i-1} R_{t+i} \tag{1.2}$$

The expectation of G_t is defined as the state value function (or the state value):

$$v_\pi(s) = \mathbb{E}(G_t|S_t = s) \tag{1.3}$$

We can see that the state value is correlated with the current system state S_t and the action policy π. Note that the return value is correlated with one single trajectory, while the state value is correlated with all the possible trajectories initiating from $S_t = s$.

Now, consider a random trajectory

$$S_t \xrightarrow{A_t} \{R_{t+1}, S_{t+1}\} \xrightarrow{A_{t+1}} \{R_{t+2}, S_{t+2}\} \xrightarrow{A_{t+2}} \{R_{t+3}, S_{t+3}\} \tag{1.4}$$

Then the trajectory return has the following form:

$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned}$$

which leads to the recursive format of the state value:

$$\begin{aligned} v_\pi(s) &= \mathbb{E}(G_t|S_t = s) \\ &= \mathbb{E}(R_{t+1}|S_t = s) + \gamma \mathbb{E}(G_{t+1}|S_t = s) \end{aligned}$$

For the first element, according to the property of the Markov process, it involves the following two parts:

1. When $S_t = s$, what action should we take? According to our policy, the possibility of making action $A_t = a$ is $\pi(A_t = a|S_t = s)$, which is written in the abbreviated form of $\pi(a|s)$.

2. After taking the action of $A_t = a$, what will the next system state be? According to our reward function, the possibility of obtaining R_{t+1} when $S_t = t$ and $A_t = a$ is $p(R_{t+1} = r|A_t = a, S_t = s)$, which is rewritten as $p(r|a, s)$.

Hence, the expected reward for making the action of $A_t = a$ when $S_t = s$ should be

$$R_d(a, s) = \sum_r p(r|a, s) r \tag{1.5}$$

Because there are many possible actions to take, the first item should then be rewritten as

$$\mathbb{E}(R_{t+1}|S_t = s) = \sum_a \pi(a|s) \sum_r p(r|a, s) r \tag{1.6}$$

Regarding the second item in the state value, it is directly correlated with the current state value S_{t+1}. Of course, the next state value is not determined as well, leading to the possibility of having $S_{t+1} = s'$ when $S_t = s$ in the form of $p(s'|s)$. For $S_{t+1} = s'$, its expected state value is $v_\pi(s') = \mathbb{E}(G_{t+1}|S_{t+1} = s')$, which leads to

$$\mathbb{E}(G_{t+1}|S_t = s) = \sum_{s'} v_\pi(s') p(s'|s)$$

Note that $p(s'|s)$ has not yet been defined, but could be expressed by existing functions. Its value is correlated with the policy (the possibility of making certain action a) and the possibility of action a takes the system to $S_{t+1} = s'$, then we have

$$\mathbb{E}(G_{t+1}|S_t = s) = \sum_{s'} v_\pi(s') \sum_a p(s'|a, s) \pi(a|s)$$

Combine the above two parts, then we have

$$\begin{aligned} v_\pi(s) &= \sum_a \pi(a|s) \sum_r p(r|a,s)r + \gamma \sum_{s'} v_\pi(s') \sum_a p(s'|a,s)\pi(a|s) \\ &= \sum_a \pi(a|s) \Big[\sum_r p(r|a,s)r + \gamma \sum_{s'} p(s'|a,s) v_\pi(s') \Big] \end{aligned}$$

For the second part (mean of the future state value), we can see there are some adjustments in the order of two summations. In the original format, the traversal is made regarding the possible future states, and the value is given in the format of value multiple the value's possibility. We can also make the traversal regarding the possible actions, in which case one action can lead to different possible states. Hence, the state value of each action is given as

$$\sum_{s'} p(s'|a,s) v_\pi(s')$$

Then we need to consider all the possible actions, which further leads to

$$\sum_{s'} v_\pi(s') \sum_a p(s'|a,s)\pi(a|s) = \sum_a \pi(a|s) \sum_{s'} p(s'|a,s) v_\pi(s') \tag{1.7}$$

The above Bellman equation is given in the element-wise format, and it can be transformed into a matrix-vector format for simplicity:

$$\begin{aligned} v_\pi(s) &= r_\pi(s) + \gamma \sum_{s'} p_\pi(s'|s) v_\pi(s') \\ r_\pi(s) &= \sum_a \pi(a|s) \sum_r p(r|a,s) r \\ p_\pi(s'|s) &= \sum_a p(s'|a,s)\pi(a|s) \end{aligned}$$

where $r_\pi(s)$ indicates the expected immediate reward and $p_\pi(s'|s)$ is the possibility for state transaction. To make it subtle, we need to include the specific states to have

$$v_\pi(s_i) = r_\pi(s_i) + \gamma \sum_{s_j} p_\pi(s_j|s_i) v_\pi(s_j) \tag{1.8}$$

which further leads to

$$\begin{bmatrix} v_\pi(s_1) \\ v_\pi(s_2) \\ \vdots \\ v_\pi(s_n) \end{bmatrix} = \begin{bmatrix} r_\pi(s_1) \\ r_\pi(s_2) \\ \vdots \\ r_\pi(s_n) \end{bmatrix} + \gamma \begin{bmatrix} p_\pi(s_1|s_1) & p_\pi(s_2|s_1) & \dots & p_\pi(s_n|s_1) \\ p_\pi(s_1|s_2) & p_\pi(s_2|s_2) & \dots & p_\pi(s_n|s_2) \\ \vdots & \vdots & \vdots & \vdots \\ p_\pi(s_1|s_n) & p_\pi(s_2|s_n) & \dots & p_\pi(s_n|s_n) \end{bmatrix} \begin{bmatrix} v_\pi(s_1) \\ v_\pi(s_2) \\ \vdots \\ v_\pi(s_n) \end{bmatrix}$$

which has the simplified version of $v_\pi = r_\pi + \gamma P_\pi v_\pi$. The reason for calculating the Bellman equation is to evaluate the performance of the employed policy. In theory, we can solve the equation by having $v_\pi = (I_n - \gamma P_\pi)^{-1} r_\pi$, but such a method involves the calculation of the matrix inverse, which is not friendly for large environments. Instead, we can have an iterative way to approach the actual value of v_π. Assign arbitrary value to v_0, then perform $v_{k+1} = r_\pi + \gamma P_\pi v_k$ until the value converges and $v_\pi = v_k$ when $k \to +\infty$.

Note that in the definition of state values, the actions are predefined. Hence, it is essential to figure out the relationship between action and reward, which leads to the concept of action value:

$$q_\pi = \mathbb{E}(G_t|s_t = s, A_t = a) \tag{1.9}$$

From the equation, q_π can be interpreted into the expectation of state value when you take a certain action with a certain current state. We know that when $S_t = s$, the chance of taking $A_t = a$ is $\pi(a|s)$ due to our policy. Hence, we have

$$v_\pi(s) = \mathbb{E}(G_t|S_t = s) = \sum_a \mathbb{E}(G_t|s_t = s, A_t = a)\pi(a|s) = \sum_a q_\pi \pi(a|s)$$

From the formulation of $v_\pi(s)$, we have

$$q_\pi = \sum_r p(r|a, s)r + \gamma \sum_{s'} p(s'|a, s)v_\pi(s') \tag{1.10}$$

2 Optimal policy and Bellman optimality equation

The core concept of reinforcement learning is to obtain the optimal policy when you give a certain environment (how states propagate and how reward is offered). We usually use the action value to determine if a policy is good or not, and the optimal policy π^* satisfies $v_{\pi^*}(s) \geq v_\pi(s)$. There are a couple of questions to settle:

1. Does the optimal policy exist?
2. Is the optimal policy unique?
3. Is the optimal policy stochastic or deterministic?
4. How to obtain the optimal policy?

To answer the above questions, we need to introduce the Bellman optimality equation. Based on the previous Bellman equation, include an extra maximum function to obtain the maximum state value:

$$\begin{aligned} v_\pi(s) &= \max_\pi \sum_a \pi(a|s)\left[\sum_r p(r|a, s)r + \gamma \sum_{s'} p(s'|a, s)v_\pi(s')\right] \\ &= \max_\pi \sum_a \pi(a|s)q(s, a) \end{aligned}$$

Similarly, we have the following equation for the matrix-vector form of the Bellman optimality equation:

$$v = \max_\pi(r_\pi + \gamma P_\pi v)$$

The best state value v^* is given as

$$v^* = r_{\pi^*} + \gamma P_{\pi^*} v^*$$

By employing the greedy optimal policy, we have

$$\pi^*(a|s) = \begin{cases} 1, & a = a^*(s) \\ 0, & a \neq a^*(s) \end{cases} \tag{2.1}$$

where

$$\begin{aligned} a^*(s) &= \text{argmax}_a q^*(a, s) \\ q^*(s, a) &= \sum_r p(r|s, a)r + \gamma \sum_{s'} p(s'|s, a)v^*(s') \end{aligned}$$

3 Value iteration and policy iteration

3.1 Value update

Based on the contraction mapping theorem, we can reach the steady point of $x = f(x)$ if we iterate the variable value along the law of $x_{k+1} = f(x_k)$. Similarly, we can use the law of

$$v_{k+1} = f(v_k) = \max_\pi(r_\pi + \gamma P_\pi v_k)$$

The algorithm contains two steps:

1. **Policy update**: Set v_k to be static, and update the policy by

$$\pi_{k+1} = \text{argmax}_\pi(r_\pi + \gamma P_\pi v_k)$$

2. **Value update**: Set π_{k+1} to be static, and update the value by

$$v_{k+1} = r_{\pi_{k+1}} + \gamma P_{\pi_{k+1}} v_k$$

Although the matrix-vector form seems tidy, its performance is restricted by calculating the matrix inverse. Hence, we need to write the algorithm in the element-wise format. Define $j = 0, 1, 2, \ldots$ to represent the number of iteration, Then we have the following pseudo algorithm:

Note that the second step is called "value update", meaning that v_{k+1} is not the state value we mentioned. The real state value should be $v_k = \sum_a \pi(a|s)[\sum_r p(r|a, s)r + \gamma \sum_{s'} p(s'|a, s)v_k]$. Previously, we mentioned that we usually use the recursive way of

$$v_{k+1} = \sum_a \pi(a|s)[\sum_r p(r|a, s)r + \gamma \sum_{s'} p(s'|a, s)v_k]$$

to obtain the state value v_k when $k \rightarrow$. Instead of running it recursively, we only update the value of v_k with one single iteration. The v_{k+1} should differ from the actual state value, but it will not affect the convergence of the algorithm.

3.2 Policy update

Different from the value update process, policy update aims to obtain the true state value in each episode. Hence, for the second step, we need to solve the state value equation recursively, and the algorithm is given as

Algorithm 1: Example algorithm for value iteration

```
Set $k = 0$ and $D = +\infty$ ;
; Initialize the system to have a random policy $\pi_k$ ;
Initialize the state value vector as $v_0$ ;
Calculate the current state value by running
  $v_{k+1} = \sum_a \pi(a|s)[\sum_r p(r|a,s)r + \gamma \sum_{s'} p(s'|a,s)v_k(s')]$ recursively ;
while $D \geq D_{\min}$ do
    for Every state $s \in \mathcal{S}$ do
        Set the action space for the current state as $\mathcal{A}_s$ ;
        for Every action $a \in \mathcal{A}_s$ do
            Calculate the immediate reward and the next lumped future reward of the
              current action $a$ according to the q-value function
              $q_k(s,a) = \sum_r p(r|s,a)r + \gamma \sum_{s'} p(s'|s,a)v_k(s')$;
            Find the best action with $a_k^*(s) = \text{argmax}_a q_k(a,s)$ ;
        end
        Update the policy by $\pi_{k+1}(a|s) = 1$ if $a = a_k^*$, and $\pi(a|s) = 0$ otherwise ;
    end
    Update the state value by
      $v_{k+1}(s) = \sum_a \pi(a|s)[\sum_r p(r|a,s)r + \gamma \sum_{s'} p(s'|a,s)v_k(s')]$ ;
    $D = v_{k+1} - v_k$ $(D = v_{\pi_{k+1}} - v_{\pi_k})$ ;
end
```

Algorithm 2: Example algorithm for policy iteration

```
Set $k = 0$ and $D = +\infty$ ;
; Initialize the system to have a random policy $\pi_k$ ;
Initialize the state value vector as $v_0$ ;
Calculate the current state value by running
  $v_{k+1} = \sum_a \pi(a|s)[\sum_r p(r|a,s)r + \gamma \sum_{s'} p(s'|a,s)v_k(s')]$ recursively ;
while $D \geq D_{\min}$ do
    for Every state $s \in \mathcal{S}$ do
        Set the action space for the current state as $\mathcal{A}_s$ ;
        for Every action $a \in \mathcal{A}_s$ do
            Calculate the immediate reward and the next lumped future reward of the
              current action $a$ according to the q-value function
              $q_k(s,a) = \sum_r p(r|s,a)r + \gamma \sum_{s'} p(s'|s,a)v_k(s')$;
            Find the best action with $a_k^*(s) = \text{argmax}_a q_k(a,s)$ ;
        end
        Update the policy by $\pi_{k+1}(a|s) = 1$ if $a = a_k^*$, and $\pi(a|s) = 0$ otherwise ;
    end
    Run $v_{k+1}(s) = \sum_a \pi(a|s)[\sum_r p(r|a,s)r + \gamma \sum_{s'} p(s'|a,s)v_k(s')]$ recursively to solve
      the state value ;
    $D = v_{k+1} - v_k$ $(D = v_{\pi_{k+1}} - v_{\pi_k})$ ;
end
```

Because it might take quite a few iterations for us to obtain the actual state value, there is also a method called truncated policy update, where the recursive process of $v_{k+1}(s) = \sum_a \pi(a|s)[\sum_r p(r|a,s)r + \gamma \sum_{s'} p(s'|a,s)v_k(s')]$ is only run for a finite number of times per iteration. The above three methods can reach convergence, just with different speeds and computational burden per episode.

4 Monte Carlo learning

Compared to previous methods, Monte Carlo learning is a model-free method. Previously, the model is expressed as the known state transition probabilities. However, it is hard to obtain the precise probability distribution for some events. For example, the expectation of flipping a coin is stochastic. Although we usually assume that the result (head or tail) follows a uniform distribution, it is not precise.

Then we need to switch to a model-free scheme, where we run the test for certain times and calculate the expectation by averaging the results.

$$\mathbb{E}(x) \approx \bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

In the previous section, the action value is defined as

$$q_\pi = \sum_r p(r|a,s)r + \gamma \sum_{s'} p(s'|a,s)v_\pi(s')$$

which is a combination of immediate reward and future reward. If the possibility distribution is unknown, we need to switch back to the original form of

$$q_\pi = \mathbb{E}(G_t|s_t = s, A_t = a) \tag{4.1}$$

Hence, when the model cannot be obtained, we use data (or samples/experiences) to analyze how the system state propagates.