

Bellman Equation

Dynamic Programming

Bellman Equation

- Reinforcement learning aims to find the optimal policy, which selects the correct action in each state to maximize the agent's return and achieve its goal.
- This class introduces two classic reinforcement learning algorithms: value iteration and policy iteration, used to find the optimal policy.
- Before delving into these algorithms, we will first learn about the Bellman equation, a fundamental concept in reinforcement learning.
- The Bellman equation is crucial for finding the optimal value and Q functions, which are essential for determining the optimal policy in reinforcement learning.

The Bellman equation

- The Bellman equation, named after Richard Bellman, is essential for solving Markov decision processes (MDPs) and finding the optimal policy in reinforcement learning.
- It is widely used to recursively compute the optimal value and Q functions, which can then be used to derive the optimal policy.
- We will explore how the Bellman equation can be used to compute the optimal value and Q functions.

The Bellman equation of the value function

- An Introduction to the Bellman Equation and Value Function
- Here's a quick overview of the Bellman equation and how it relates to the value of a state. The equation suggests that the value of a state can be calculated by combining the immediate reward and the discounted value of the next state. For example, if you take action a in state s and move to state s' to receive reward r , the Bellman equation for the value function is expressed as follows:

$$V(s) = R(s, a, s') + \gamma V(s')$$

- $R(s, a, s')$ represents the immediate reward obtained from performing an action a in state s and transitioning to the next state s' .
- γ is the discount factor
- $V(s')$ implies the value of the next state

Return and discount factor

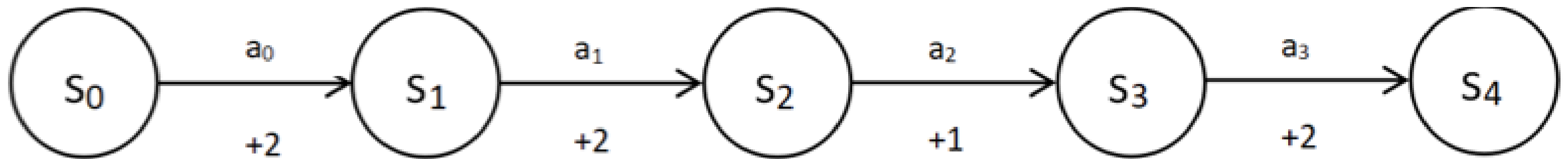
A return can be defined as the sum of the rewards obtained by the agent in an episode. The return is often denoted by R . Say the agent starts from the initial state at time step $t = 0$ and reaches the final state at time step T , then the return obtained by the agent is given as:

$$R(\tau) = r_0 + r_1 + r_2 + \dots + r_T$$

$$R(\tau) = \sum_{t=0}^T r_t$$

Return and discount factor

Let's understand this with an example; consider the trajectory (episode) τ :



$$R(\tau) = 2 + 2 + 1 + 2 = 7.$$

- The agent's goal in reinforcement learning is to maximize the return, which is the sum of rewards obtained over an episode.
- Maximizing the return requires performing the correct action in each state.
- Achieving the correct actions in each state is facilitated by following the optimal policy.
- The optimal policy is the one that enables the agent to attain the maximum return by selecting the correct actions in each state.

The Bellman equation of the value function

Let us define the return for continuous tasks?

$$R(\tau) = r_0 + r_1 + r_2 + \dots + r_\infty \qquad R(\tau) = \sum_{t=0}^{\infty} \gamma^t r_t$$

How can we maximize the return that just sums to infinity?

how is this discount factor helping us?

Discount Factor

- The discount factor in reinforcement learning prevents the return from reaching infinity by balancing the importance of future rewards and immediate rewards.
- Its value ranges from 0 to 1, where lower values prioritize immediate rewards, and higher values prioritize future rewards.
- A small discount factor (close to 0) emphasizes immediate rewards, while a high discount factor (close to 1) emphasizes future rewards.
- An example with different discount factor values can illustrate how this balance impacts decision-making in reinforcement learning.

Small vs. large discount factor

Let's set the discount factor to a small value, say 0.2

$$\begin{aligned} R &= (\gamma)^0 r_0 + (\gamma)^1 r_1 + (\gamma)^2 r_2 + \dots \\ &= (0.2)^0 r_0 + (0.2)^1 r_1 + (0.2)^2 r_2 + \dots \\ &= (1)r_0 + (0.2)r_1 + (0.04)r_2 + \dots \end{aligned}$$

When we set the discount factor to 0, that is , it implies that we consider only the immediate reward r_0 and not the reward obtained from the future time steps.

$$\begin{aligned} R &= (\gamma)^0 r_0 + (\gamma)^1 r_1 + (\gamma)^2 r_2 + \dots \\ &= (0)^0 r_0 + (0)^1 r_1 + (0)^2 r_2 + \dots \\ &= (1)r_0 + (0)r_1 + (0)r_2 + \dots \\ &= r_0 \end{aligned}$$

$$(x + 7)(5x + 6)$$

$$9x^3y^4(7x^{-2} + 5y^4)$$

$$\frac{3x^2 + 5x - 7}{x + 9}$$

Example of The Bellman
equation of the value
function

$$15x + 4$$

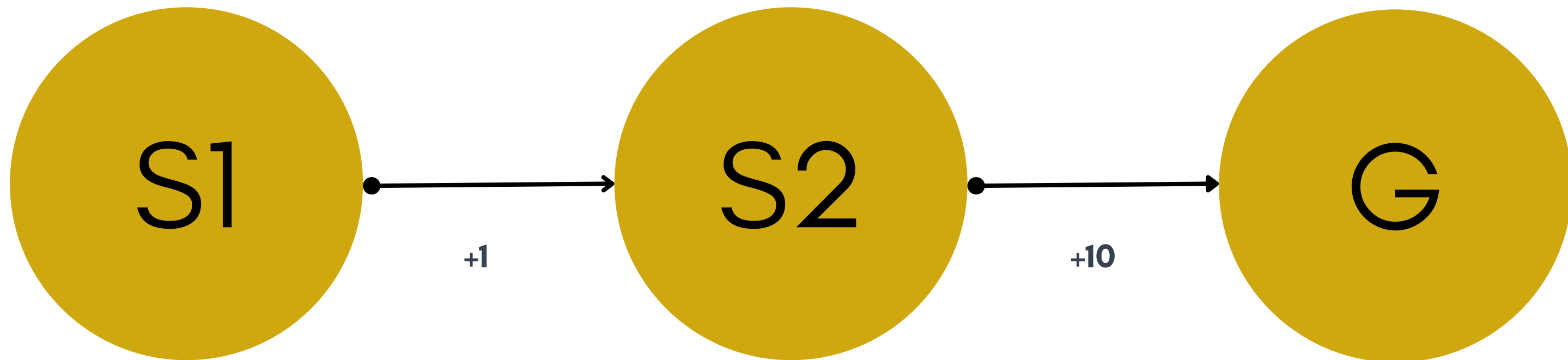
$$\frac{(x - 5)}{7} = \frac{(x + 2)}{9}$$

Let's consider a simple grid world environment with three states (S1, S2, and the goal state G) and two actions (left and right).

The agent receives a reward of -1 for each step taken and a reward of +10 for reaching the goal state. The discount factor γ is set to 0.9.

The Bellman equation for the value function $V(s)$ in this case can be expressed as:

$$V(s) = \max_a \left(R(s, a) + \gamma \sum_{s'} P(s'|s, a) V(s') \right)$$



Example 1: The Bellman equation of the value function

Let's calculate the value of state S1 using the Bellman equation. Assuming the agent is in state S1 and takes action left, it moves to state S2 with a reward of -1. Using the Bellman equation:

$$V(S1) = \max(-1 + 0.9 \times V(S2), -1 + 0.9 \times V(S2))$$

Since both actions lead to the same state S2, we can simplify the equation:

$$V(S1) = -1 + 0.9 \times V(S2)$$

Similarly, for state S2, the agent receives a reward of -1 for each action, and both actions lead to the goal state G with a reward of +10. Using the Bellman equation:

$$V(S2) = \max(-1 + 0.9 \times V(G), -1 + 0.9 \times V(G))$$

Again, since both actions lead to the same state G, we simplify the equation:

$$V(S2) = -1 + 0.9 \times V(G)$$

Finally, for the goal state G, the value is simply the reward:

$$V(G) = 10$$

Now, we can substitute the value of $V(G)$ into the equation for $V(S2)$, and then substitute the value of $V(S2)$ into the equation for $V(S1)$ to find the value of $V(S1)$:

$$V(S2) = -1 + 0.9 \times 10 = -1 + 9 = 8$$

$$V(S1) = -1 + 0.9 \times 8 = -1 + 7.2 = 6.2$$

Therefore, the value of state S1 is 6.2.

The Bellman equation of the Q function

In the equation, the following applies:

$R(s, a, s')$ implies the immediate reward obtained while performing an action a in state s and moving to the next state s' .
 γ is the discount factor, which determines the importance of future rewards relative to immediate rewards.

$Q(s', a')$ is the Q value of the next state-action pair"

- The Bellman equation expresses the Q-value of a state-action pair as the sum of the immediate reward and the discounted maximum Q-value of the next state. This reflects the agent's goal of maximizing cumulative rewards over time.
- By iteratively updating Q-values using the Bellman equation, an agent can learn an optimal policy that maximizes its expected cumulative reward in the long run. The optimal policy selects actions with the highest Q-values for each state.
- The Bellman equation is fundamental in reinforcement learning algorithms, such as Q-learning, which use it to iteratively update Q-values based on observed rewards and transitions. This process allows the agent to learn an optimal policy through trial and error.

$$Q(s, a) = R(s, a, s') + \gamma Q(s', a')$$

EXAMPLE 2: THE BELLMAN EQUATION OF THE Q FUNCTION

ADD MORE TEXT

Consider a simple grid world where an agent can move left, right, up, or down. The grid has a reward of -1 for each step, and the agent receives a reward of +10 for reaching the goal state. The discount factor γ is set to 0.9.

Let's consider a specific example where the agent is in state S and can take actions to move left, right, up, or down. The rewards for each action are as follows:

- **Moving left or right: -1**
- **Moving up or down: -1**

The goal state (state G) has a reward of +10. Since the grid is deterministic, the agent moves to the desired state with probability 1.

We can calculate the Q-values for each state-action pair using the Bellman equation and the given rewards. Let's start with the initial Q-values:

$$Q(S, \text{left}) = 0 \quad Q(S, \text{right}) = 0 \quad Q(S, \text{up}) = 0 \quad Q(S, \text{down}) = 0$$

To update the Q-values, we apply the Bellman equation for each state-action pair. For example, to update $Q(S, \text{left})$:

$$\begin{aligned} Q(S, \text{left}) &= -1 + 0.9 \times \max(Q(\text{next state}, \text{all actions})) \\ &= -1 + 0.9 \times \max(Q(S, \text{left}), Q(S, \text{right}), Q(S, \text{up}), Q(S, \text{down})) = -1 + 0.9 \times \max(0, 0, 0, 0) \\ &= -1 \end{aligned}$$

Similarly, we can update $Q(S, \text{right})$, $Q(S, \text{up})$, and $Q(S, \text{down})$. After updating, the Q-values become:

$$Q(S, \text{left}) = -1 \quad Q(S, \text{right}) = -1 \quad Q(S, \text{up}) = -1 \quad Q(S, \text{down}) = -1$$

These updated Q-values reflect the expected cumulative rewards the agent can achieve from each state-action pair following an optimal policy in the grid world environment.

$$V^*(s) = \max_a \left(R(s, a) + \gamma \sum_{s'} P(s'|s, a) V^*(s') \right)$$

THE BELLMAN OPTIMALITY EQUATION

The Bellman optimality equation is a key concept in reinforcement learning that describes the relationship between the optimal value function and the optimal policy in a Markov decision process (MDP). It defines the optimal value of a state as the maximum expected return achievable by following an optimal policy from that state onward. The Bellman optimality equation for the optimal value function $V^*(s)$ of a state s in an MDP is given by:

γ is the discount factor, which determines the importance of future rewards relative to immediate rewards.

$R(s,a)$ is the immediate reward received after taking action a in state s .
 $P(s'|s,a)$ is the probability of transitioning to state s' from state s after taking action a .

\max denotes the maximum over all possible actions a in state s .

BELLMAN OPTIMALITY EQUATION

- The Bellman optimality equation states that the optimal value of a state is equal to the maximum expected return that can be achieved by taking the best action a in that state and then following the optimal policy thereafter. It provides a recursive way to compute the optimal value function, which represents the maximum expected return from each state under the optimal policy.
- The optimal policy π^* can be derived from the optimal value function V^* by selecting the action a in each state s that maximizes the expression inside the max operator in the Bellman optimality equation. This policy is optimal in the sense that it maximizes the expected cumulative reward over time in the MDP.
- The Bellman optimality equation is fundamental in reinforcement learning because it provides a formal way to compute the optimal value function and derive the optimal policy in MDPs. Algorithms like value iteration and policy iteration use the Bellman optimality equation to iteratively improve the value function and converge to the optimal policy.

Dynamic programming

Dynamic programming (DP) is a problem-solving technique that breaks down complex problems into smaller, simpler sub-problems. For each sub-problem, DP computes and stores the solution instead of solving the problem as a whole. When the same subproblem arises, DP uses the already computed solution instead of re-computing it, which significantly reduces computation time. DP is widely used in computer science, mathematics, bioinformatics, and other fields.

- Value iteration
- Policy iteration

Algorithm 1 Value Iteration

```
1: Initialize  $V(s) = 0$  for all states  $s$ 
2: Initialize  $\epsilon > 0$  as the convergence threshold
3: Initialize  $\Delta$  as a large value
4: Initialize iteration counter  $k = 0$ 
5: while  $\Delta > \epsilon$  do
6:    $\Delta \leftarrow 0$ 
7:   for each state  $s$  do
8:      $v \leftarrow V(s)$ 
9:      $V(s) \leftarrow \max_a (R(s, a) + \gamma \sum_{s'} P(s'|s, a)V(s'))$ 
10:     $\Delta \leftarrow \max(\Delta, |v - V(s)|)$ 
11:   end for
12:   Increment iteration counter:  $k \leftarrow k + 1$ 
13:   Output current value function  $V$  and iteration number  $k$ 
14: end while
```

Value iteration

In order to determine the best course of action for an agent in each state, the value iteration method focuses on identifying the optimal policy. To do so, we first need to calculate the optimal value function, which can then be used to derive the optimal policy. So how do we calculate the optimal value function? The answer lies in the optimal Bellman equation of the value function. According to this equation, the optimal value function can be computed as follows:

Thank you!