Welcome to the Foundations Bootcamp

Session 1 Lecture Notes[*]

Leo Klenner[†] Henry Fung[†] Cory Combs[†]

1 The Learning Task

In this section, we formulate the problem of learning sequential control strategies more precisely.

In a MDP, the agent senses its current state s_t, chooses an action a_t from a set of actions A that it can performed in s_t, and performs it. In response, the environment gives the agent an immediate reward as a function of its state and action: $r_t = r(s_t, a_t)$, and produces the subsequent state: $s_{t+1} = \delta(s_t, a_t)$.

The agent does not necessarily know the reward function r and the state transition function δ. In other words, the agent might not know the consequence of its actions: the new state s_{t+1} that it will transition from its current state by performing action a_t, and the resulting reward r_t that it receives.

The task of the agent is to learn a policy π that dictates the agent's choice of its action a_t based on its current observed state s_t. Mathematically, this is expressed as:

$$\pi(s_t) = a_t \ \ \forall s_t \in S \tag{1}$$

What is the optimal policy that we want the agent to learn? One obvious approach is to require the optimal policy to produce the greatest possible cumulative reward for the agent over time. Mathematically, we express the cumulative reward for the agent as follows:

$$V^\pi(s_t) = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + ... \tag{2}$$

The sequence of rewards $(r_t, r_{t+1}, r_{t+2}, ...)$ in Equation 2 is generated by an agent that starts in s_t and repeatedly uses policy π to select its actions: $a_i = \pi(s_i)$

In Equation 2, γ is the "discounting factor" that specifies the importance of future rewards relative to immediate rewards. For example, $\gamma = 1$ implies that the agent values immediate reward as much as future rewards. On the other hand, if γ is close to 0, that means the agent only cares about immediate rewards and places very little value on future rewards. In many cases, it is reasonable to discount future rewards relative to immediate rewards because the agent prefers to obtain the reward sooner rather than later.

Armed with the above definitions and equations, we are now ready to precisely state the agent's learning task: we require the agent to learn a policy π that maximizes the cumulative discounted reward $V^\pi(s)$ for all s. We call such a policy the optimal policy π^*. The optimal policy is expressed mathematically

[*]Written using XƎLaTeX.
[†]Johns Hopkins University School of Advanced International Studies (SAIS), Washington, D.C.

as follows:

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, V^{\pi}(s) \tag{3}$$

The maximum possible cumulative reward that an agent can achieved by following the optimal policy π^* is denoted as $V^*(s)$. In the next section, we will illustrate the concepts that we presented so far in a six-grid world example.

2 A Six-Grid World Example

The six grids in Figure 1 represents the six possible states of an autonomous robot (the agent). Our task is to train the agent to move from an arbitrary start grid through the shortest path to the goal grid G. To start, let's specify the actions, immediate rewards, and the goal state of this problem:

- The goal state is labelled G on Figure 1.
- The arrows represent the possible actions that the agent can perform at each state. For example, if the agent is at state (grid) A in Figure 1, then as indicated by the arrows, it can move in the left, right, or up directions.
- In Figure 2, the numbers on the arrows are the immediate rewards $r(s, a)$ that the agent receives in the corresponding state-action pair. For example, if the agent starts in grids A or C, then it receives an immediate reward of 100 if it moves to G. The immediate reward for all other action-state pairs is zero. In other words, other than moving to the goal state G from states B or C, the agent receives no immediate reward.
- If the agent enters state G, it can only remain in G (hence, the looping arrow in state G). For this reason, G is also called the "absorbing state". Once the agent is in the absorbing state, the learning episode ends. Subsequently, a new training episode would start with the agent randomly positioned at an arbitrary state. The agent will continue to "play the game" of moving to G from an arbitrary starting grid (we call each game a "training episode") until it eventually learns an optimal policy by which it moves from a starting grid to G through the shortest path.

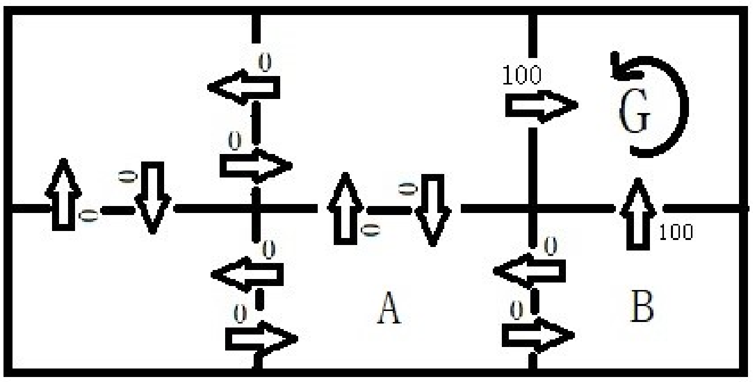

Figure 1: The six-grid world: immediate reward values.

The number labelled on each grid in Figure 2 is the largest possible cumulative reward that the agent (starting from that grid) can achieve if it follows the optimal policy $V^*(s)$. Let's illustrate this with an example. If the agent starts in state A in Figure 2, then its optimal policy (shortest path to G) is to

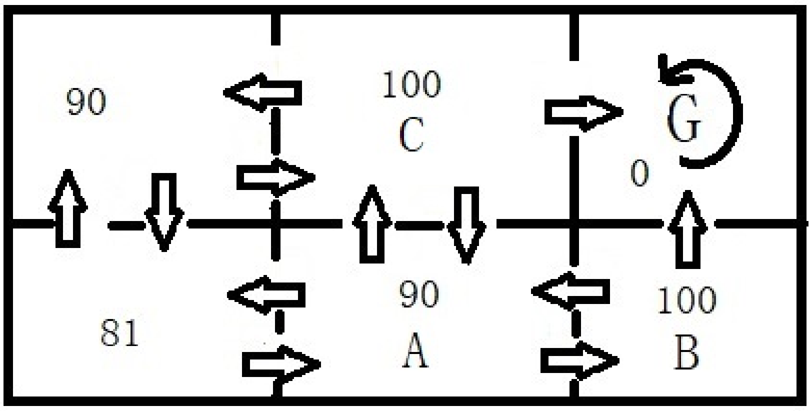

Figure 2: The six-grid world: V*(s) values.

move right to state B, and then move up to state G. By following the optimal policy, the agent receives $V^*(s = A)$ that is computed using Equation 3. Note that in this example, we assume the discount factor γ to be 0.9, and the immediate rewards specified according to Figure 1. The largest possible cumulative reward at state A is computed as follows:

$$V^*(s = A) = 0 + 0.9 * 100 = 90 \tag{4}$$

Thus, grid A in Figure 2 is labelled 90, the largest possible cumulative reward that can be achieved by an agent that starts in A.

The objective of the agent is to learn an optimal policy π^* that directs it to G through the shortest path from an arbitrary starting state. In general, it is difficult to learn π^* directly since the agent only observes a sequence of immediate rewards. One approach is to learn a numerical evaluation function that is defined over states and actions, then implement the optimal policy in terms of the evaluation function.

An obvious evaluation function that the agent can learn is $V^*(s)$. Note that the agent does not have prior knowledge of $V^*(s)$ (the values that are labelled on the grids in Figure 2); rather it must learn $V^*(s)$ by exploring the environment (ie: through repeated attempts of reaching G from an initial state in training episodes).

If the agent learns V*(s) (ie: it has good estimates of the grid values in Figure 2 through completing many training episodes), then it can formulate preferences for each state. For example, if $V^*(B) > V^*(A)$, then the agent prefers state B over state A because it can achieve a higher cumulative reward from state B than state A. In general, the optimal action a^* for the agent at state s, is the action that maximizes the sum of the immediate reward $r(s, a)$ plus the largest achievable cumulative reward at the immediate successor state, discounted by γ. This is expressed mathematically as follows:

$$\pi^*(s) = \underset{a}{\operatorname{argmax}}[r(s, a) + \gamma V^*(\delta(s, a))] \tag{5}$$

Recall that $\delta(s, a)$ denotes the successor state resulting from applying action a in current state s. From Equation 5, the agent can acquire the optimal policy $\pi^*(s)$ by learning V^*, provided that it has perfect knowledge of the immediate reward function r and the state transition function δ. In other words, the agent can perfectly predict the consequence of its actions in the six-grid world: the rewards that it will receive, and the subsequent state that it will transition by performing action a in state s. If r and δ

are not known, then we cannot use Equation 5 to acquire the optimal policy; rather we need to learn a different evaluation function called the Q function. The Q function will be presented in the next section.

3 The Q Function

Let us define the evaluation function $Q(s, a)$ as follows: the value Q is the immediate reward received by the agent after performing action a from state s, plus the value (discounted by γ) of following the optimal policy thereafter (V^*). This is expressed mathematically as follows:

$$Q(s, a) = r(s, a) + \gamma V^*(\delta(s, a)) \tag{6}$$

Going back to our six-grid world, the Q values for each state-action pair (s, a) is labelled in Figure 3. As an example, let's compute the Q value for the state action pair ($s = A, a = right$) (the arrow that is highlighted in red in Figure 3).

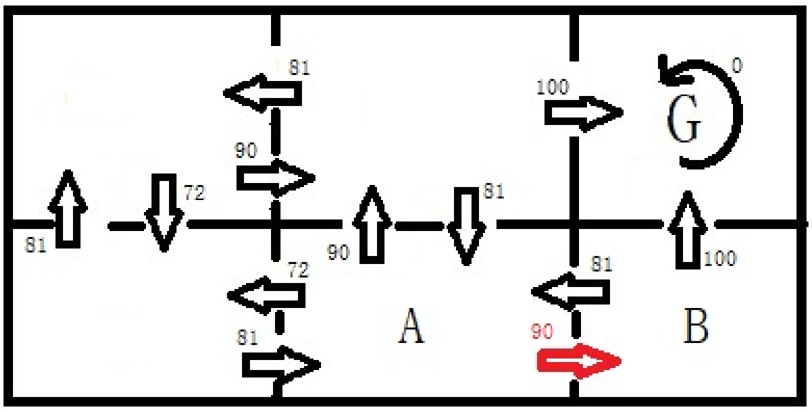

Figure 3: The six-grid world: Q values.

By moving to the right, we know that the agent will transition from state A to state B and it will receive an immediate reward of 0. Since the $V^*(s)$ of state B is 100 (see Figure 2); thus, the Q value of the state-action pair (s=state A, a=right) is computed from Equation 6:

$$Q(s = A, a = right) = 0 + 0.9 * 100 = 90 \tag{7}$$

To choose the optimal action a in state s, we can rewrite Equation 5 in terms of $Q(s, a)$ as follows:

$$\pi^*(s) = \underset{a}{\text{argmax}}\, Q(s, a) \tag{8}$$

Why are we rewriting Equation 5 and expressing $\pi^*(s)$ in items of Q in Equation 8? The main reason is that if the agent learns Q in Equation 8 instead of the V* in Equation 5, then it will be able to select the optimal action in state s even when if has no knowledge of the reward functions r and the state transition function δ. Instead, the agent only needs to consider the possible actions $a \in A$ that it can perform in state s, and choose the action that maximizes $Q(s, a)$. For example, in state B (Figure 3), the agent has two possible actions: up and left. Since the Q value for up is greater than the Q value for left (denoted by the values on the arrows), the agent chooses the action up in state B as per Equation 8. In summary, once the agent learns the Q values (all values in Figure 3), then it can use Equation 8 to choose the optimal action in state s.

4 The Bellman Equation

As demonstrated in the last example, learning the Q values corresponds to learning the optimal policy. This is because if all the Q values are known in the six-grid world (Figure 3), then we can use Equation 8 to find the optimal action in each state s. For this reason, our learning task in reduced to learning the Q values. This brings us to the vital question: how can Q be learned?

First, notice the close relationship between $Q(s, a)$ and $V^*(s)$:

$$V^*(s) = \max_{a'} Q(s, a') \tag{9}$$

We can show the relationship between $Q(s, a)$ and $V^*(s)$ as defined by Equation 9 with a simple example (Figure 4). Suppose the agent is in state A. From the top figure in Figure 4, $V^*(s)$ of state A is 90. This is also the maximum Q value that the agent can achieved in state A by moving right ($a' = right$) as shown by the red arrow in the bottom figure of Figure 4. Similarly, it can be shown that the relationship Equation 9 holds for all other grids.

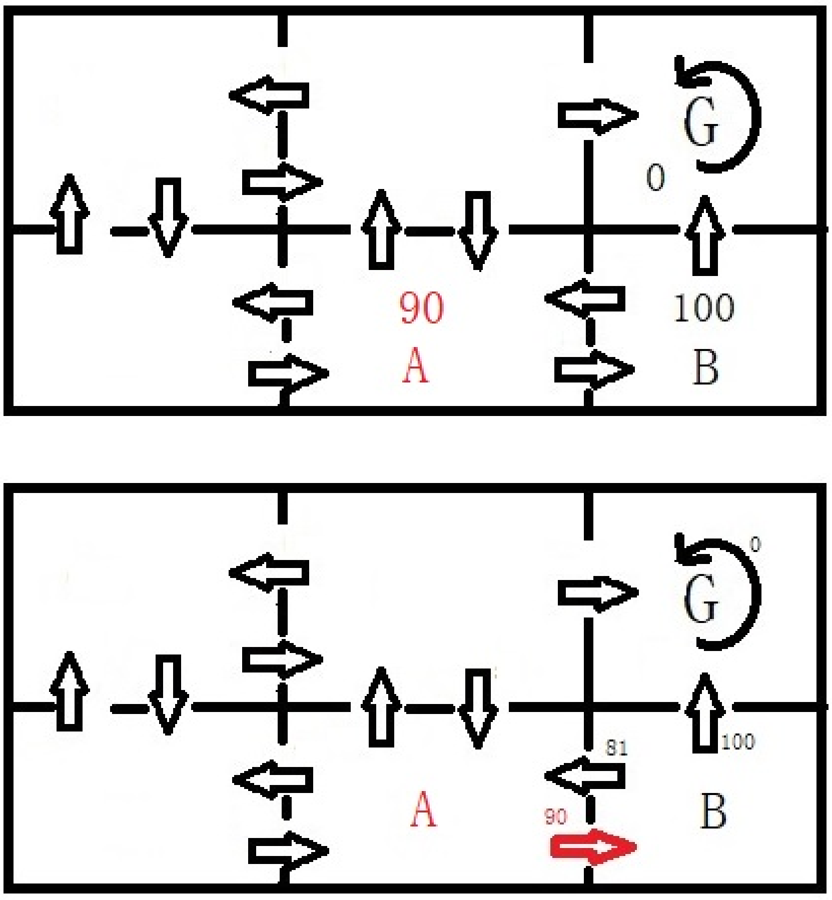

Figure 4: The relationship between V* and Q

Using Equation 9, we rewrite Equation 6 as:

$$Q(s,a) = r(s,a) + \gamma \max_{a'} Q(\delta(s,a), a') \tag{10}$$

Equation 10 is known as the Bellman equation. Armed with the Bellman equation, we can use an algorithm called the ”Value Iteration” (presented in class 3 of this workshop) to learn the Q values (the values in Figure 3). Once the agent learns the Q values, then it can use Equation 8 to obtain the optimal action at every s. In other words, by learning Q, the agent learns the optimal policy.