**Homework#2: Q-Learning for Maze Traversal**

**Name : 임용성 (Lim Yong Sung)**

**Student ID : 2019310649**

**[목차]**

**Ⅰ. Background**

* **Q-Learning Algorithm**

**Ⅱ. Results and Analysis**

**Ⅱ. Code Description**

**Ⅲ. Visualization**

**<Summary>**

텍스트, 도표, 패턴이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트, 도표이(가) 표시된 사진

자동 생성된 설명텍스트, 도표, 라인, 스크린샷이(가) 표시된 사진

자동 생성된 설명

**Ⅰ. Background**

According to the wiki, Q-Learning is a model-free reinforcement learning technique used to find the optimal policy for a given finite Markov decision process (MDP). In an MDP, the agent interacts with the environment and learns the optimal policy by receiving feedback in the form of rewards.

An MDP follows the Markov Property, which means that the current state completely characterizes the state of the world, and the future is only concerned with the present, not the past. It is defined by a tuple (S, A, R, P, r), where:

* S is the set of states in the environment.
* A is the set of actions that the agent can take.
* R is the reward function that provides the immediate reward when an action is taken in a particular state.
* P is the state transition probability function, which defines the probability of transitioning to a new state when an action is taken in a current state.
* r is the discount factor that determines the importance of future rewards.

An MRP (Markov Reward Process) is a Markov chain augmented with rewards. The goal in an MRP is to maximize the return, which is a measure of the cumulative rewards obtained over time. The return is influenced by the discount rate, denoted as γ.

The discount rate serves two purposes. First, it helps avoid infinite returns in cyclic Markov processes by assigning less weight to future rewards as the time horizon extends. This prevents the return from growing unbounded and ensures convergence. Second, the discount rate reflects the degree of uncertainty about future rewards. A smaller discount rate values immediate rewards more and considers future rewards with less weight, indicating a more myopic decision-making approach. Conversely, a larger discount rate values future rewards more and takes a more forward-looking perspective.

The return in an MRP is computed by summing the discounted rewards over time. The formula for the return is:

폰트, 텍스트, 화이트, 라인이(가) 표시된 사진

자동 생성된 설명

Here, the discount rate determines the relative importance of immediate rewards versus future rewards. It allows for trade-offs between immediate gains and long-term benefits, reflecting the time preference of the decision-maker.

By maximizing the return in an MRP, we aim to find policies that achieve the highest cumulative rewards over time, considering both the immediate rewards and the potential future rewards discounted by γ.

The Bellman equation for an MRP expresses the relationship between the value function of the current state and the value function of the next state.  
텍스트, 폰트, 친필, 화이트이(가) 표시된 사진

자동 생성된 설명

In an MDP, which is an extension of MRP, actions to be taken by agents are included. The value function of an MDP is expressed as follows:

텍스트, 스크린샷, 폰트, 대수학이(가) 표시된 사진

자동 생성된 설명

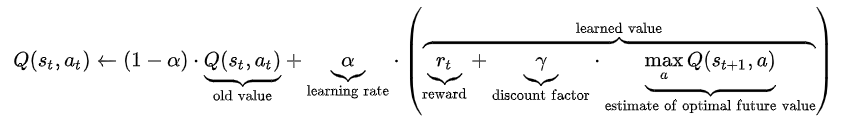
To maximize the return in an MDP, Q-Learning aims to find the optimal policy by maximizing the Q-value function Q(s, a), which represents the expected cumulative reward when taking action a in state s. The Bellman Optimality Equation provides the theoretical foundation for Q-Learning and is given by:

폰트, 텍스트, 화이트, 서예이(가) 표시된 사진

자동 생성된 설명

Here, Q\* represents the optimal Q-value function, R(s, a) is the immediate reward, γ is the discount factor, P(s' | s, a) is the transition probability, and max Q\*(s', a') represents the maximum Q-value over all possible actions in the next state.

The Q-Learning algorithm updates the Q-values iteratively based on the observed rewards and transitions. The Q-Learning update equation is:



Here, Q(s, a) is the Q-value for state s and action a, α is the learning rate that determines the weight given to new information, R(s, a) is the immediate reward, γ is the discount factor, and max Q(s', a') represents the maximum Q-value over all possible actions in the next state.

By applying Q-Learning iteratively, the agent can learn the optimal Q-values for each state-action pair and ultimately determine the optimal policy for

**Ⅱ. Result and Analysis**

Let's examine the progression of the Episode by referring to the following image.

텍스트, 도표, 패턴이(가) 표시된 사진

자동 생성된 설명

Let me analyze the picture above.

First of all, since the Exploration Rate is set high in the early stages of low Epsiode and all Q\_Table are similar in initialization state, the Agent updates the Q\_Table as it explores most of the space. An episode ends when an agent randomly accidentally arrives at the destination. And as Epsiode progresses, the agent's Exploration Rate gradually decreases, so it tends to rely on Q\_Table (dependent on experience). For reference, as I will explain later, in the code I made, if there are multiple same Max values when walking on the street based on Q\_Table, it is set to go in a random direction among the directions that are max values. Anyway, as Episode progresses, the agent updates the Q\_Table and becomes increasingly dependent on experience.

There is an important point here. In fact, after a long period of code analysis and research, the agent was observed to move back and forth (Ex, the agent moves left and right indefinitely), and as a result of checking the Q\_Table in this case, the same Left and Right existed at the same value (for example, -0.166 or -0.125). To understand this reason, let's look at the equation of Q-Learning.

텍스트, 폰트, 라인, 화이트이(가) 표시된 사진

자동 생성된 설명

Notice the inside of the right parenthesis in the above equation.

As explained in Background, Q-Learning gradually updates the Q\_table of the final destination's Reward in the form of Backpropagation, and the final destination's Reward affects the way to the final destination, increasing its value. However, if the Discount Rate is very low, it converges to zero of the impact of Reward at the final destination, and most of the values converge to a specific value. The value is the value of zero inside the right bracket in the above equation.

Therefore, if the maze is long from the start position to the end position, the smaller the Discount Rate, the smaller the impact on the final destination, so that most Q\_Table around the start position converges to the same value and repeatedly moves back and forth in the space!

As a way to solve this problem, it was possible to solve the problem by significantly increasing the Discount Rate.

Now let's take a closer look at the picture above. As shown in the figure above, it can be seen that there is less tendency to pierce walls well in the beginning, and the closer to the final destination, the greater the tendency to pierce walls.

This is due to the fact that when Reward reaches its final destination, +100, -1 when drilling a wall, and -0.1 when just moving, the closer it is to the final destination, the less Discount multiplied, so even if the wall is pierced and reaches the final destination, it has a significant impact on the update of the Q-Table. Therefore, this trend becomes stronger toward the second half, so the tendency is written in the form of a compensation in which the updated value of the Q-Table breaks through a large wall (-1) rather than going back along the road. In the vicinity of the Start position, the effect of the final reward +100 has become very small because the Discount Rate has been multiplied several times, so the Reward (-1) when the wall is drilled is operated larger. The following is part of the Q-Table.

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

As you can see in the figure, there is a lot of -1.15. Also, interestingly, if it's not -1.15, you can see a difference of -0.25, i.e. when it's barely pierced the wall (-1 + 0.1 = 0.9).

For this reason, it can be seen that less wall drilling is performed around the starting position, and the tendency to wall drilling becomes stronger as we reach the final destination.

For a maze, let's observe what happens when you sweep the Learning Rate and Discount Rate.

텍스트, 스크린샷, 도표, 폰트이(가) 표시된 사진

자동 생성된 설명

As shown in the figure above, the smaller the learning\_rate(a), the smaller the size of the Q\_table is, the smaller the size of the Q\_table's update, so even if the final result is reached and Reward (100), it eventually converges to a specific value as Episode is repeated.

So we can see that 0.2 or 0.4 where a is very small does not reach the final destination.

Because the above maze problem is difficult (because the distance is long from the start position to the target position) it is difficult to learn with these small learning rates.(The update size is small, so the final reward cannot affect and eventually converges all around the starting position, resulting in the Agent repeating only the same position) Note that most cases , it will repeat about 5 spaces or less from the starting position, but in my code, it is designed to behave randomly.

Now let's understand the impact of the Discover Rate. I explained the impact of Discount above, and it is the same. While the Learning Rate governs the overall size of updates, the Discount Rate more directly affects the reduction of final rewards. Smaller Discount Rate converges faster, so you become shortsighted, and the larger the Discount Rate, the more far you can see. Therefore, in order not to face the same converging problem, the Discount Rate must be set according to the tendency of the maze to some extent.

**Ⅲ. Code Description**

**Ⅳ. Visualization**

**TASK 1~4**

텍스트, 도표, 패턴이(가) 표시된 사진

자동 생성된 설명

**TASK 5**

텍스트, 스크린샷, 도표, 폰트이(가) 표시된 사진

자동 생성된 설명

텍스트, 도표, 라인, 스크린샷이(가) 표시된 사진

자동 생성된 설명