Machine Learning
Reinforcement Q-Learning
Muhammad Husain | 1301153626 | IF39-INT

1. Problem Analysis

Given a set of matrix, by using Q-Learning we have to create an agent that able to move from it's initial state to it's goal state.

2. Design

a. Initialization

First, initializing the Q and R matrix. Q matrix is initialized as a 2 dimensional array or matrix, with all the value assigned to 0. While R matrix is from file 'DataTugasML.txt' where it is a representational of each state rewards.
Second, we need to initialize gamma, in this case i set gamma as 0.8. Gamma will affect how much the Q will consider the future actions, the greater the gamma, it will consider more future actions, while less gamma will less consider future actions.
Also, the action of the agent will be defined as follow:
Up means **y-1**
Down means **y+1**
Left means **x-1**
Right means **x+1**
Please **be aware** of the agent's action definition above, as it is quite different from what we usually use.

b. Training

Model Training is done by start to move the agent from initial state to final/goal state. The agent movement will be randomized until the agent has reach the goal state.
This process of moving from initial to goal by choosing action randomly is done by **n**(which i set to 20 episode in my program) times of episode. The purpose is to learn the environment reward and considering its future action.

c. Testing

After 20 episode of training, the agent will have a Q matrix that represent the memory or experience through randomly exploring the environment. By using this Q matrix, we will

try to reach the goal, unlike in training phase, we will choose the action by choosing the highest reward action, until reaching the goal state.

3. Experiment Result Evaluation

The first experiment of the program shown that, on testing phase, the agent will move back to its previous position since the score for the previous state is lower compared to other action. This resulting a forever loop (i.e Up, down, up , down, …) . I solved this by creating a rule that an agent may not go back to its previous position. And it works better compared to the first experiment. The result of my second experiment is as follow:

0 8 | move up
1 8 | move right
1 7 | move up
2 7 | move right
3 7 | move right
3 6 | move up
4 6 | move right
5 6 | move right
5 5 | move up
5 4 | move up
6 4 | move right
7 4 | move right
7 3 | move up
7 2 | move up
8 2 | move right
8 1 | move up
8 0 | move up

Total reward to reach goal: -98.4

Unfortunately the loop were stopped for some reason which i do not know, even though the agent almost reach the goal.