Albert Richie (A98407956)

CSE 190 – Final Project

arichie@ucsd.edu

For the final project, I decided to implement q-learning algorithm which is model free reinforcement learning in extension of programming assignment 3.

Objective

Find an optimal action-selection policy of grid when MDP model and its environment are not available by using different exploration vs exploitation functions (particularly ε-greedy and softmax function).

Method

With Q learning algorithm, our robot has to learn its optimal action-selection policy by interacting with its environment. Therefore, for this project each available grid has four Q value (expected utility) need to be stored; one for each available action that the robot can do. These values are calculated by:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\,(R_{t+1} + \gamma \max Q(s_{t+1}, a) - Q(s_t, a_t))$$

Q learning will eventually converge to optimal policy if all the states and actions in those states are explored enough. The convergence depends on the complexity of the environment and how the algorithm will choose explore vs exploit. In my project, I utilized several methods for the algorithm to choose which state the robot will transition into while learning Q value of the environment. The methods I will presents are the ε-greedy and softmax algorithm.

With ε-greedy, the robot will select an action with fixed probability depending on ε parameter where $0 \le \varepsilon \le 1$. To accomplish this, I used random module in python to choose a random value. If this random value is smaller than ε, then the robot will choose random available action from particular state. Otherwise, the robot will choose the action with the highest Q value.

The second method is a softmax function which calculate the probabilities of choosing an action a:

$$P(s_t, a_t) = \frac{e^{\frac{Q(s_t, a_t)}{\tau}}}{\sum_a e^{\frac{Q(s_t, a_t)}{\tau}}}$$

Where τ is a temperature that decides how the action will be chosen. If the temperature is high, the actions will be chosen almost randomly with the same probability. If the temperature is low, the action will be chosen similar to those with ε-greedy. In this project, the temperature is decreased constantly in every iteration.

Result

Q learning should converge into optimal policy that reflects the expected utility of given state. In my result, the time it takes for the algorithm to converge relies on the complexity of the environment. In programming assignment 3, using A star and MDP value iteration should converge faster since the robot is knows the model of the environment to generate a policy for the robot. Also, in order to reduce the convergence time, I added negative reward for hitting the wall.

In Q learning, using constant ε will treat explore vs exploit balance constantly for every iteration. To simulate the real behavior of robot in the environment, I chose a model which will treat exploration higher in the beginning and decayed slowly toward the end of the simulation. In comparison to the result with MDP value iteration policy, the algorithm has more policy to optimize the number of steps taken to the goal. In programming assignment 3, it is quite obvious that some policies are chosen in order to avoid the penalty of hitting the wall or falling into the pit.

Using softmax action, the robot will treats the action with higher Q value better than those with lower Q value. This allows the current highest Q value to be tested thoroughly. This creates problem as the environment get larger as some grid will not be explored as much and lengthen the time it takes for the algorithm to converge. In conclusion, softmax performs better when the environment is smaller or when the right policy is more obvious or closer to the goal but it is having a harder time choosing the right policy when the goal is farther away and/or multiple path because the optimal action might have lower current Q value but not explored well enough by the robot.

Github

https://github.com/arichie/CSE190Q-Learning

Youtube

https://youtu.be/X0LLOEvy6Hs