Project 4: Reinforcement Learning

Sagnik Ghosh
Person Number : 50289151
University at Buffalo - Department of Computer Science
sagnikgh@buffalo.edu

Abstract

Our main goal is to let our agent learn the shortest path to the goal. In the environment the agent controls a green square, and the goal is to navigate to the yellow square (reward +1), using the shortest path. At the start of each episode all squares are randomly placed within a 5x5 grid-world. The agent has 100 steps to achieve as large a reward as possible. They have the same position over each reset, thus the agent needs to learn a fixed optimal path.

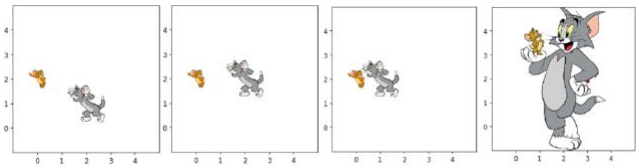

Figure 1:

1 Reinforcement learning:

Reinforcement learning is a direction in Machine Learning where an agent learn how to behave in a environment by performing actions and seeing the results. In reinforcement learning, an agent learns from trial-and-error feedback rewards from its environment, and results in a policy that maps states to actions to maximize the long-term total reward as a delayed supervision signal. Reinforcement learning combining with the neural networks has made great progress recently, including playing Atari games and beating world champions at the game of Go. It is also widely used in robotics.

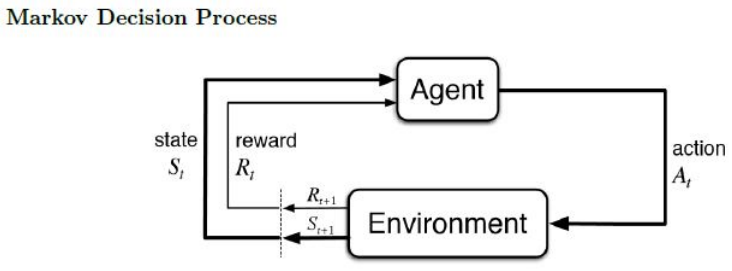

Figure 2:

Submitted to 32nd Conference on Neural Information Processing Systems (NIPS 2018). Do not distribute.

2 What parts have you implemented?

2.1 3-layer neural network using Keras library:

The DQN takes the coordinate of the agent (x1,y1) and the coordinate of the target (x2,y2) as input and covert them into a Q-value as output. It is passed through two hidden networks, and output a vector of Q-values for each action possible in the given state.

Example: Q(st,a1) - q-value for a given state s, if we choose action a1 We need to choose such an action, that will return the higest Q-value.

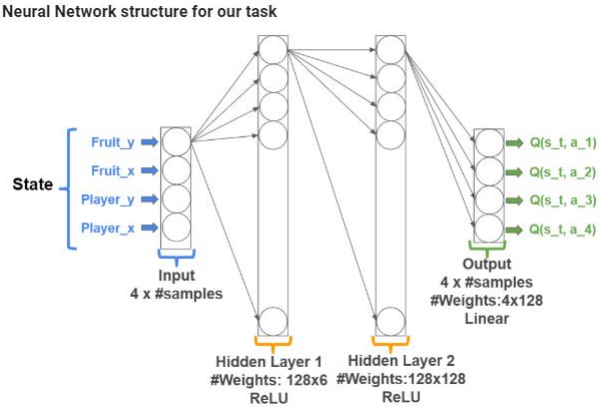

Figure 3:

- The DQN has two hidden layers.
- Activation function for the first and second hidden layers is 'relu'.
- Activation function for the final layers is 'relu'.
- Number of hidden nodes is 128.
- Number of the output is the same as the size of the action space, i.e. 4

The 'brain' of the agent is where the model is created and held. The agent learns to navigate through the environment and its experiences are stored in its memory. In the beginning, the agent does really badly. But over time, it begins to associate states with best actions to do by updating the q values.

2.2 Implement exponential-decay formula for Epsilon :

The agent will randomly select its action at first by a certain percentage, called 'exploration rate' or 'epsilon'. Initially, the value of Epsilon will be high. This is because at first, the agent will try all kinds of things before it starts to see the patterns. When it is not deciding the action randomly, the agent will predict the reward value based on the current state and pick the action that will give the highest reward. We want our agent to decrease the number of random action, as it goes, so we introduce an exponential-decay epsilon, that eventually will allow our agent to explore the environment.

- Exponential-decay formula for epsilon:

$$\epsilon = \epsilon_{min} + (\epsilon_{max} - \epsilon_{min}) * e^{-}\lambda.|S|$$

where, min,max[0,1] - hyperparameter for epsilon |S| - total number of steps

2.3 Implement Q-function :

Q-function was implemented to predict the action based on the q value of the next state:
A Q-value is an array of four element which contains the probability of the four direction in which the agent can move. The highest probability signifies that that moving in that direction in the next state, will gain the agent the maximum reward.
the Q-function was implemented based on the following formula:
if episode terminates at step t+1;

$$\mathrm{Q}_t = r_t,$$

otherwise,

$$\mathrm{Q}_t = r_t + \gamma max_a Q(s_t, a_t; \theta)$$

3 Can these snippets be improved and how it will influence the training the agent?

- By changing the number of hidden layers and by changing the number of nodes in the hidden layer the performance of the DQN.
- the lambda value is also a hyperparameter and it can be tuned to increase the performance of epsilon.

4 How quickly your agent were able to learn?

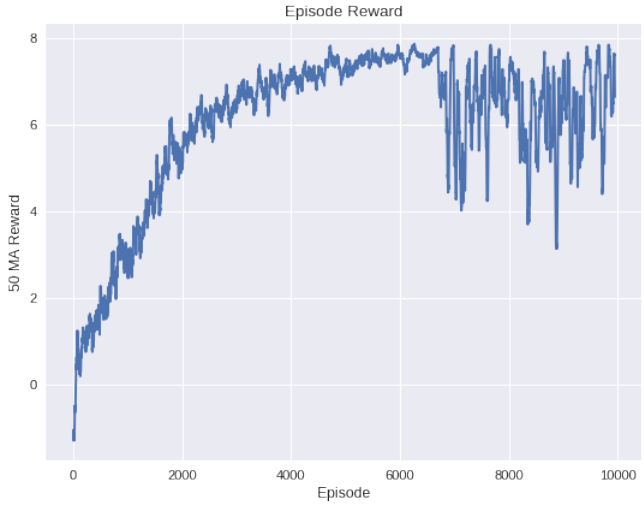

Figure 4:

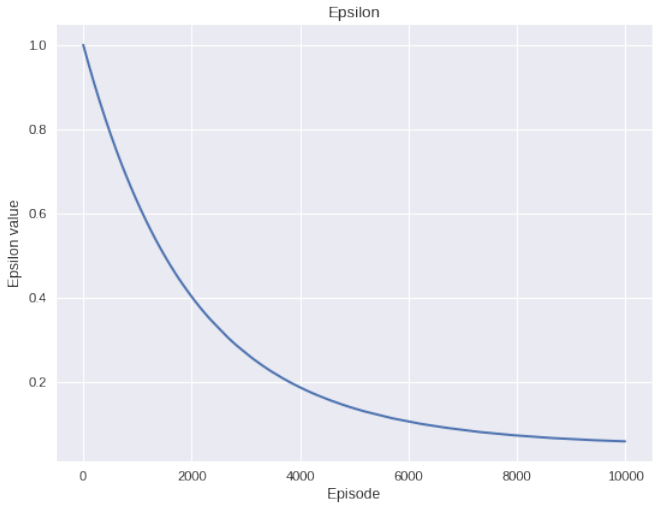

Figure 5:

5 Explain what happens in reinforcement learning if the agent always chooses the action that maximizes the Q-value. Suggest two ways to force the agent to explore:

If the agent will always choose the action to maximize Q. The agent will not explore enough to find the best possible action from each of the state. Hence, the agent will get stuck in non-optimal policies.
The two ways by which agent we can force the agent to explore is :
1. The first method is to make the agent pick random values of the Epsilon, intermittently, so that it starts exploring again.
2. The sond way is to set the initial value of the "Exploration rate". If the initial values are high the unexplored region will be explored quickly.

6 Calculate Q-value for the given states and provide all the calculation steps.

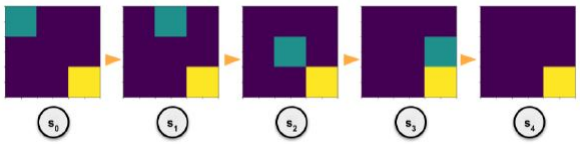

Figure 6:

In the above figure, the agent takes the following sequence of actions: RIGHT ! DOWN ! RIGHT ! DOWN.
It is an optimal path for the agent to take to reach the goal (although this is not the only possible optimal path).
Your task is to fill out the Q-Table for the above states, where

$$\gamma = 0 : 99.$$

Hint: Start calculating Q-function from the last state.

Please find the answer to this question in the other pdf.