Project 4

Reinforcement Learning

Muhammad A. Masood
Department of Computer Science
University at Buffalo
State University of New York
UB person# 50291164
mmasood@buffalo.edu

Abstract

The purpose of the project 3, was to combine reinforcement learning with deep learning. For this purpose, a 3 layered neural network was implemented on Keras which served QDN network for reinforcement learning. The game of Tom&Jerry was implemented in which agent was trained to navigate in grid.

1 Problem Statement

The environment consisted of 5x5 grid with 25 possible states (0,0), (0,1), (0,2) and so on. The agent can take one of four actions. Our objective was to train the agent to reach its goal via the shortest possible path.

2 Code Implementation

In the code, it was required to fill in the code for three parts naming the neural network implementation, exponential decay formula for epsilon and Q-function.

2.1 Neural Network

Keras was used for the neural network implementation of this task. This neural network or DQN served as the brain of the agent. It took a stack of six tuples as input. The model consisted of two feed-forward sequential hidden layers. The hidden layers consisted of 128 nodes and for both nodes 'relu' was used as the activation function. The activation function of the output layer was linear.

2.2 Exponential decay formula for epsilon

The exponential decay formula for epsilon can be written as:

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

This function implements the exploration rate of the agent.

2.3 Q-Function

Q-Function was also implemented in the code and it was an indication of goodness of an action. It chose the best path based on the knowledge of states, actions and their rewards.

2.4 Results

The hyperparameters were changed for all the values and their effect on the performance of the agent was checked. Changing the minimum epsilon and maximum epsilon values in a small range did not make any significant difference. The number of episodes was changed first to 5000, which sped up the training process but had an adverse effect on the agent learning. The best performance was achieved when the number of episodes was decreased to 8000. It sped up the process while also did not affect the performance.

3 Effect of always choosing the max reward Q-value:

When the agent always chooses the action that maximizes the Q-value, it doesn't learn anything new and whenever it is faced by an unknown environment, it is prone to make more mistakes. It is the classical exploration vs exploitation dilemma. It can be solved by following methods:

- Agent can be modified in a way such that it takes a random action after a certain period of time. This can be done through probability generation.
- Model-based approaches that compute a choice of action based on its expected reward and the model's uncertainty about that reward

4 Q-Table

STATE	UP	DOWN	LEFT	RIGHT
S0	3.9	3.94	3.9	3.94
S1	2.94	2.97	2.9	2.97
S2	1.94	1.99	1.94	1.99
S3	0.97	1	0.97	0.99
S4	0	0	0	0

Starting from S4 all values are 0

For S3:

*Up: -1+0.99*1.99=0.97*

*Down: 1+0.99*0=1*

*Left: -1+0.99*1.99=0.97*

*Right: 0+0.99*1= 0.99*

For S2:

*Up: -1 + 0.99 *2.97 = 1.94*

*Down: 1 + 0.99 * 1 = 1.99*

*Left: -1 + 0.99 *2.97 = 1.94*

*Right: 1 + 0.99 * 1 = 1.99*

For S1:

*Up: 0.99 *2.97 = 2.94*

*Down: = 1+ 0.99*1.99 = 2.97*

*Left: -1 + .99*3.94 = 2.90*

*Right: + 0.99*1.99 = 2.97*

For S0:

*Up: 0 + 0.99 * 3.94 = 3.90*

*Down: 1+ 0.99*2.97 = 3.94*

*Left: 0 + 0.99 * 3.94 = 3.90*

*Right: 1+ 0.99*2.97 = 3.94*