CSE474/574: Introduction to Machine Learning (Fall 2018)

Instructor: Sargur N. Srihari, Alina Vereshchaka, Nathan Margaglio

********** November 29, 2018**********

Commenting code?

Project 4: Tom and Jerry in Reinforcement learning

Due Date: Wednesday, Dec 5, 11.59pm

1 Overview

The project combines reinforcement learning and deep learning. Your task is to teach the agent to navigate in the grid-world environment. We have modeled Tom and Jerry cartoon, where Tom, a cat, is chasing Jerry, a mouse. In our modeled game the task for Tom (an agent) is to find the shortest path to Jerry (a goal), given that the initial positions of Tom and Jerry are deterministic. To solve the problem, we would apply deep reinforcement learning algorithm - DQN (Deep Q-Network), that was one of the first breakthrough successes in applying deep learning to reinforcement learning.

There are two main parts of the project:

1. Coding part - build neural network in Keras for the deep learning part and write code in Python for the reinforcement learning part.
2. Writing part - answer two questions.

This project does not require GPU and is supported by Windows/Linux/MacOS, though you are welcome to use Google Colab (Jupyter notebook environment that requires no setup and allows to use GPU for free). The grading will be based on the mean reward.

1.1 Coding tasks [max 60 points total + max 20 bonus points]

All coding tasks has to be implemented within a Jupyter Notebook (can be downloaded at UBlearns). Refer to "Project 4: Supporting material" for more details.

1. Build a 3-layer neural network using Keras library [20 points]
2. Implement exponential-decay formula for epsilon [10 points]
3. Implement Q-function [10 points]
4. Report [20 points]
   At minimum there should be a description of your code-snippets implementations:
   - What parts have you implemented?
   - What is their role in training the agent?
   - Can these snippets be improved and how it will influence the training the agent?
   - How quickly your agent were able to learn?

We would also suggest you tune the hyperparameters, like *epsilon max/min*, *number of episodes*, *gamma*, etc. to see influence the total time of training and the mean reward.

5. Bonus [20 points]
   Implement DQN using at least TWO environments from OpenAI's Gym library (one of them has to be from Atari). You can use Stable Baselines implementation of DQN, which provides a detailed documentation. In the report for each implemented environment reason how quickly your agent were able to learn using pre-implemented DQN function, plot graphs of the rewards, etc.

1.2 Writing tasks [max 40 points total]

The writing part has to be submitted in pdf-version only. You are welcome to use NIPS template.

1. 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. [20 points]

2. Calculate Q-value for the given states and provide all the calculation steps. [20 points]

   Consider a deterministic environment which is a 3x3 grid, where one space of the grid is occupied by the agent (green square) and another is occupied by a goal (yellow square). The agent's action space consists of 4 actions: UP, DOWN, LEFT, and RIGHT. The goal is to have the agent move onto the space that the goal is occupying in as little moves as possible.

   Initially, the agent is set to be in the upper-left corner and the goal is in the lower-right corner. The agent receives a reward of:

1	when it moves closer to the goal
-1	when it moves away from the goal
0	when it does not move at all (e.g., tries to move into an edge)

   Consider the following possible optimal set of actions and their resulting states, that reach the goal in the smallest number of steps:

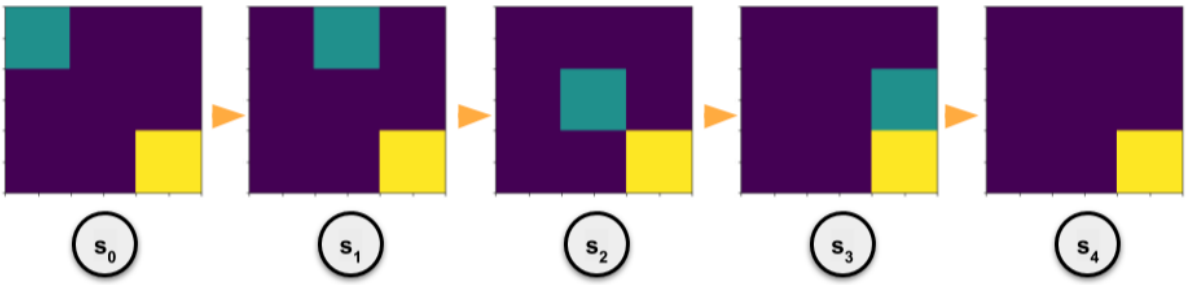

Figure 1: One of the possible optimal actions sequence

In Figure 1, 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.

$$Q(s_t, a_t) = r_t + \gamma * max_a Q(s_{t+1}, a) \tag{1}$$

Table 1: Q-Table (for the states in Figure 1)

	ACTIONS			
STATE	UP	DOWN	LEFT	RIGHT
0				
1				
2				
3				
4				

2 Deliverables

You need to submit only one zip-file *proj4.zip*, that will contain the following:

- Jupyter Notebook (*main.ipynb*), that will contain your code for missing parts.
- Report and answers for two writing questions (*report.pdf*) should be done in electronic version and submitted as pdf.
- For bonus points, you need to submit two fully commented Jupyter Notebooks (*bonus1.ipynb, bonus2.ipynb*) and report (*bonus.pdf*). In your report, reason about the performance of the inbuilt DQN function and how it behave on different environments.

Submission:

Submit the PDF on a CSE student server with the following script:
`submit_cse474 proj4.pdf` for undergraduates
`submit_cse574 proj4.pdf` for graduates

3 Scoring Rubric

1. Code part [60 points + 20 bonus points]
   (a) Implement Neural Network [20 points]
   (b) Implement Epsilon [10 points]
   (c) Implement Q-function [10 points]
   (d) Report [20 points]
   (e) Bonus + report [20 points]
2. Writing part [40 points]
   (a) Question #1 [20 points]
   (b) Question #2 [20 points]

4 Due Date and Time

The due date is **11:59PM, Dec 5**.