Overview

In this assignment, you will gain experience working with OpenAI Gym, which is a set of problems that can be explored with different reinforcement learning algorithms. This assignment is designed to help you apply the concepts you have been learning about Q-learning algorithms to the “cartpole” problem, a common reinforcement learning problem.

Note: The original code referenced in this assignment was written in Python 2.x. You have been given a zipped folder containing an updated Python 3 version of the code that will work in the Apporto environment. To make this code work, some lines have been commented out. Please leave these as comments.

Reference: Surma, G. (2018). *Cartpole*. Github repository. Retrieved from https://github.com/gsurma/cartpole.

Prompt

Access the Virtual Lab (Apporto) by using the link in the **Virtual Lab Access** module. It is recommended that you use the Chrome browser to access the Virtual Lab. If prompted to allow the Virtual Lab access to your clipboard, click “Yes”, as this will allow you to copy text from your desktop into applications in the Virtual Lab environment.

1. Review the following reading: Cartpole: Introduction to Reinforcement Learning. In order to run the code, upload the Cartpole.zip folder into the Virtual Lab (Apporto). Unzip the folder, then upload the unzipped folder into your Documents folder in Apporto. Refer to the Jupyter Notebook in Apporto (Virtual Lab) Tutorial to help with these tasks.

   Note: The Cartpole folder contains the Cartpole.ipynb file (Jupyter Notebook) and a scores folder containing score_logger.py (Python file). It is *very* important to keep the score_logger.py file in the scores folder (directory).
2. Open Jupyter Notebook and open up the Cartpole.ipynb and score_logger.py files. Be sure to review the code in both of these files. Rename the Cartpole.ipynb file using the following naming convention:

   <YourLastName>_<YourFirstName>_Assignment5.ipynb

   Thus, if your name is Jane Doe, please name the submission file “Doe_Jane_Assignment5.ipynb”.
3. Next, run the code in Cartpole.ipynb. The code will take several minutes to run and you should see a stream of output while the file runs. When you see the following output, the program is complete:

Solved in _ runs, _ total runs.

Note: If you receive the error “NameError: name ‘exit’ is not defined” after the above line, you can ignore it.

4. **Modify the values for the exploration factor, discount factor, and learning rates in the code** to understand how those values affect the performance of the algorithm. Be sure to place each experiment in a different code block so that your instructor can view all of your changes.

   Note: Discount factor = GAMMA, learning rate = LEARNING_RATE, exploration factor = combination of EXPLORATION_MAX, EXPLORATION_MIN, and EXPLORATION_DECAY.
5. Create a Markdown cell in your Jupyter Notebook after the code and its outputs. In this cell, you will be asked to analyze the code and relate it to the concepts from your readings. You are expected to include resources to support your answers, and must include **citations** for those resources.

   Specifically, you must address the following rubric criteria:
   - **Explain how reinforcement learning concepts apply to the cartpole problem.**
     - What is the goal of the agent in this case?
       1. The agent is tasked with balancing the pole on the cart by moving the cart side to side. This action is meant to keep the pole upright.
     - What are the various state values?
       1. The agent has been successful if it can achieve an average score of 195 or better
       2. Each frame that the pole is balanced the score is increased by +1
       3. The pole is “balanced” so long as it is less than 15 degrees from center, either left or right
       4. The cart should not move more than 2.4 units from center
       5. If the pole tips 15 degrees left or right OR the cart moves more than 2.4 units left or right, the agent has failed and the environment is reset for the next test / trial
     - What are the possible actions that can be performed?
       1. The cart can move left or right
     - What reinforcement algorithm is used for this problem?
       1. The agent is given feedback after each action it takes.
       2. The feedback is the reward (increase in score) and a change to the next state of the environment.
   - **Analyze how experience replay is applied to the cartpole problem.**
     - How does experience replay work in this algorithm?
       1. A random sample of “experiences” is taken from the batch during each run. These experiences are used to train the agent in small batches, which prevents the agent from having to train from

scratch each time. This process also reduces correlation between subsequent actions, and ensures the generated Q-values are of highest quality.

- What is the effect of introducing a discount factor for calculating the future rewards?
    1. A discount factor helps the agent perform better in the long-term. This is done by elongating the “time horizon”, essentially giving the agent “future-thinking”. What this really translates to is giving the agent concern for distant future rewards as opposed to short-term rewards. An agent will consider the sum total of all future rewards forthcoming when evaluating its actions, if the gamma is equal to 1 (or very close to it); this keeps the agent “working” to always improve its own total score. A gamma of 0 will make an agent that is only concerned with actions that produce an immediate reward.

- **Analyze how neural networks are used in deep Q-learning.**
    - Explain the neural network architecture that is used in the cartpole problem.
        1. The architecture of the neural network is such that states of the environment act as input while pairs of actions and Q-values are the outputs. The best possible action for the given state is represented by the output with the highest Q-value.
    - How does the neural network make the Q-learning algorithm more efficient?
        1. Standard Q-Learning maps a state-action pair to a q-value manually, whereas in Deep Q-Learning (DQN) a neural network is used to approximate the values of the Q-table. WIth standard Q-learning the Q-value for all possible state-action pairs is manually placed in the Q-table, but with larger amounts of state-actions pairs (think thousands or even millions) this becomes infeasible. Instead, a neural network is used to approximate the Q-values in the Q-table, and thus a DQN is created.
    - What difference do you see in the algorithm performance when you increase or decrease the learning rate?
        1. In order to update the weights in the neural network after each run, we use the learning rate variable. Changing the LR will affect the rate of change in the weights within the model. A low learning rate typically results in more reliable training but slower optimization. Higher learning rates beget higher losses during training. Since the goal is ultimately a decrease in loss over time, lower learning rates are preferred.

Guidelines for Submission

Please submit your completed IPYNB file. Make sure that your file is named as specified above, and that you have addressed all rubric criteria in your response. Sources should be cited in APA style.

Resources:

Beysolow, T. (2019). Chapter 3. In *Applied Reinforcement Learning with python: With Openai Gym, tensorflow, and keras*. essay, Apress.

Gulli, A., & Pal, S. (2017). Chapter 8: AI Game Playing. In *Deep learning with keras: Implement neural networks with Keras on Theano and tensorflow* (pp. 271–274). essay, Packt.

Gupta, A. (2022, May 13). *Deep Q-learning*. GeeksforGeeks. Retrieved September 23, 2022, from https://www.geeksforgeeks.org/deep-q-learning/

Surma, G. (2019, November 10). *Cartpole - introduction to reinforcement learning (DQN - deep Q-learning)*. Medium. Retrieved September 23, 2022, from https://gsurma.medium.com/cartpole-introduction-to-reinforcement-learning-ed0eb5b58288

Surmenok, P. (2021, April 19). *Estimating an optimal learning rate for a deep neural network*. Medium. Retrieved September 23, 2022, from https://towardsdatascience.com/estimating-optimal-learning-rate-for-a-deep-neural-network-ce32f2556ce0

Wang, M. (2021, October 3). *Deep Q-learning tutorial: Mindqn*. Medium. Retrieved September 23, 2022, from https://towardsdatascience.com/deep-q-learning-tutorial-mindqn-2a4c855abffc