CSCI 397: Deep Q-Network (DQN) for CartPole

Due Date: Saturday, November 18th @ 11:59 PM

Introduction

In this assignment, you will implement and train a Deep Q-Network (DQN) to balance a pole on a moving cart in the CartPole environment from the OpenAI Gym library. This is a classic problem in the field of reinforcement learning where an agent learns to perform a task of balancing a pole that is hinged to a cart by applying forces to the cart's base.

The environment is a simulation that includes a cart that moves along a frictionless track with a pole hinged on top of it. The agent has to decide between two actions at each time step: move the cart left or right. The goal is to keep the pole balanced upright as long as possible.

Tasks

1. Environment Setup:

- ☐ Instantiate the CartPole environment.
- ☐ Familiarize yourself with the state space and action space of the environment.

2. Implement DQN:

- ☐ Define the architecture of the DQN with at least two hidden layers.
- ☐ Initialize two networks, the policy network and the target network.
- ☐ Implement the epsilon-greedy strategy for action selection.
- ☐ Define the replay buffer to store transitions.

3. Training Process:

- ☐ Implement the function to optimize the model using samples from the replay buffer.
- ☐ Implement the logic for updating the target network with the policy network's weights at regular intervals.
- ☐ Define the criteria for the environment's termination state and how to reset the environment.

4. Monitor Learning:

- Plot the duration of each episode (number of time steps the pole was balanced) over time.
- Calculate the moving average of the duration over the last 100 episodes and plot it as well.

5. Hyperparameter Tuning:

- Experiment with different values for hyperparameters like learning rate, discount factor, epsilon decay, etc.

6. Evaluate Performance:

- After training, evaluate the performance of the DQN agent over a set number of episodes and compute the average duration the agent kept the pole balanced.

7. Analysis:

- Discuss how the DQN performed.
- Explain the impact of hyperparameters on learning.
- Include plots to visualize the agent's learning progress.

Deliverables

1. A Python script (.py file) containing the implementation of the DQN.
2. A report (PDF) discussing:
    - The architecture of the neural network used.
    - The final hyperparameters chosen.
    - Training progression plots.
    - An evaluation of the trained DQN agent's performance.
    - Insights and difficulties encountered during the implementation.
3. Ensure that the code is well-commented and the report clearly communicates your process and findings. Submit both the code and the report through the assignment portal before the due date. Late submissions will be penalized according to the course's late submission policy.

Good luck!