ECSE5552025 Assignment 1

README

Alex Pi
alex.pi@mail.mcgill.ca

February 19, 2025

Contents

1 Overview

This repository contains the implementation of several value-based reinforcement learning algorithms applied to two classic control tasks provided by OpenAI Gym:

- **CartPole-v1**: A task in which the agent must balance a pole on a cart.
- **MountainCar-v0**: A task where the agent must drive a car up a steep hill.

The implemented algorithms include:

1. **Q-Learning**
2. **SARSA**
3. **Deep Q-Network (DQN)** (a deep reinforcement learning approach)

The objectives of this assignment are to:

- Gain familiarity with the OpenAI Gym environment.
- Implement and analyze Q-Learning and SARSA.
- Experiment with a deep RL algorithm (DQN) on classic control tasks.
- Understand and discuss differences among various reinforcement learning concepts such as off-policy versus on-policy, model-based versus model-free, etc.

2 Repository Structure

The repository is organized as follows:

```
.project/
 CartPole-v1/
    cartpole_qlearning.py    # Q-Learning implementation for CartPole-v1
    cartpole_sarsa.py        # SARSA implementation for CartPole-v1
    cartpole_dqn.py          # DQN implementation for CartPole-v1
 MountainCar-v0/
    mountaincar_qlearning.py # Q-Learning implementation for MountainCar-v0
    mountaincar_sarsa.py     # SARSA implementation for MountainCar-v0
    mountaincar_dqn.py       # DQN implementation for MountainCar-v0
 README.pdf                   # Project documentation
 ECSE555 Report.pdf           # Assignment Report
```

If your file organization is different, please adjust the references accordingly.

3 Dependencies and Requirements

The project is implemented in Python (version 3.7 or higher). The following packages, with their specified versions, are required:

- **numpy** version: `1.24.3`
- **matplotlib** version: `3.7.2`
- **gym** version: `0.26.2`
- **torch** version: `2.4.1`

4 How to Run the Code

Each algorithm/environment combination is implemented in its own Python script. The scripts print out intermediate training information (such as episode rewards and current epsilon values), evaluate the learned policies, and generate plots of training and evaluation rewards.

4.1 MountainCar Environment

Q-Learning

Run the following command in your terminal:

```
python mountaincar_qlearning.py
```

This script:

- Discretizes the continuous state space using a binning approach.
- Implements the Q-Learning update rule.
- Evaluates the learned policy and plots training/evaluation rewards.

SARSA

Run:

```
python mountaincar_sarsa.py
```

The SARSA script uses an epsilon-greedy policy and the SARSA update rule. It follows a similar structure to the Q-Learning script but selects the next action based on the current policy.

DQN (Deep Q-Network)

Run:

```
python mountaincar_dqn.py
```

This script implements:

- A neural network approximator for the Q-value function.
- A replay buffer for experience replay.
- A target network to stabilize training.

4.2 CartPole Environment

Q-Learning

Run:

```
python cartpole_qlearning.py
```

This implementation discretizes the continuous state space of CartPole and applies the Q-Learning update rule.

SARSA

Run:

```
python cartpole_sarsa.py
```

This script implements SARSA for CartPole, utilizing a similar discretization strategy as the Q-Learning version.

DQN (Deep Q-Network)

Run:

```
python cartpole_dqn.py
```

The DQN implementation for CartPole uses neural networks and experience replay to learn a policy that maximizes the reward.

5 Hyperparameter Tuning and Experimentation

Within each script, hyperparameters such as learning rate, discount factor, epsilon (and its decay parameters), and batch sizes (for DQN) are defined. You are encouraged to experiment with different values to observe the effect on learning performance.

For example, in the DQN script, you might see lines like:

```
GAMMA = 0.99
LR = 1e-3
BATCH_SIZE = 64
MEMORY_SIZE = 10000
TARGET_UPDATE = 20
EPSILON_START = 1.0
EPSILON_END = 0.01
EPSILON_DECAY = 500
```

You can modify these parameters and re-run the scripts to compare learning curves.