UDACITY

Return to Classroom

DISCUSS ON STUDENT HUB

Navigation

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

Awesome ⭐ ⭐ ⭐ ⭐ ⭐
You have done a great work and acquired all the concepts needed in this project. Congratulations 😃

Few useful resources:

- You can use PyTorch-summary to visualize your architecture.
- The current State-Of-The-Art (SOTA) RL algorithm for discrete action space is Rainbow. This is a great post explaining how to combine all of these improvements together.
- PyTorch implementation of DQN and its variants: RL Adventure
- Advanced DQNs: Playing Pac-man with Deep Reinforcement Learning
- Speeding up DQN on PyTorch: how to solve Pong in 30 minutes

Training Code

The repository (or zip file) includes functional, well-documented, and organized code for training the agent.
The code is well documented. Keep the good work. 👍

Some guidelines to make your repository standout:

- The Best of the Best Practices (BOBP) Guide for Python
- Python Best Practices: 5 Tips For Better Code

The code is written in PyTorch and Python 3.

The submission includes the saved model weights of the successful agent.

README

The GitHub (or zip file) submission includes a `README.md` file in the root of the repository.

Few Tips to improve readme:

Add specific sections like:

Environment details

- What is the environment like?
- What is the task of the agent?
- Is it a single agent or many agents?
- Is the state space continuous or discrete? In either case how it is represented
- What are the possible actions agent can take?
- How is the agent rewarded?
- And finally when is the environment considered solved.

Dependencies

The Readme is a place where you can let the user know about installation dependencies and downloading needed files, so include in the readme:

- How to download the project environment?
- Any additional installations you require for the project.
- Also remember to add the all the modules that you have imported along with their version as requirements (dependencies) of this project, so that anyone can reproduce your results.

How to use:

Finally in the Readme mention how to run your code, which files to run (in case of command line what commands to run) and in case of Jupyter Notebook which cells contain training the agent part and which

cells contain seeing the trained agent part.

The README describes the the project environment details (i.e., the state and action spaces, and when the environment is considered solved).

The README has instructions for installing dependencies or downloading needed files.

The README describes how to run the code in the repository, to train the agent. For additional resources on creating READMEs or using Markdown, see here and here.

Report

The submission includes a file in the root of the GitHub repository or zip file (one of `Report.md`, `Report.ipynb`, or `Report.pdf`) that provides a description of the implementation.

The report clearly describes the learning algorithm, along with the chosen hyperparameters. It also describes the model architectures for any neural networks.

Awesome! 😃 A very well written report. It includes the algorithm, model architecture and major hyperparameters. The report clearly demonstrates that you have understood the concepts of Q networks and how it can be used to approximate policy for continuous state space. 🏆

A plot of rewards per episode is included to illustrate that the agent is able to receive an average reward (over 100 episodes) of at least +13. The submission reports the number of episodes needed to solve the environment.

Great work in achieving an average reward (over 100 episodes) of at least +13. 😃

The submission has concrete future ideas for improving the agent's performance.

DOWNLOAD PROJECT

RETURN TO PATH