UDACITY

‹ Return to "Deep Reinforcement Learning Nanodegree" in the classroom

DISCUSS ON STUDENT HUB

Continuous Control

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

Training Code

The repository includes functional, well-documented, and organized code for training the agent.

The code is written in PyTorch and Python 3.

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

README

The GitHub submission includes a `README.md` file in the root of the repository.

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.

Well documented `Readme.md` you have given clear instructions for downloading the relevant files, details about the project environment etc. Great work. 😃

However, it is a convention to include dependencies too in `Readme.md` you can add in the readme that you need Python 3.5, PyTorch, Numpy... etc so that anyone accessing your repo, knows what are the modules required. And also how to install them. You can use the `requirements.txt` and installation instructions.

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 (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.

A plot of rewards per episode is included to illustrate that either:

- *[version 1]* the agent receives an average reward (over 100 episodes) of at least +30, or
- *[version 2]* the agent is able to receive an average reward (over 100 episodes, and over all 20 agents) of at least +30.

The submission reports the number of episodes needed to solve the environment.

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

DOWNLOAD PROJECT

RETURN TO PATH