UDACITY

‹ Back to Deep Reinforcement Learning Nanodegree

Continuous Control

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

Dear Student,

Going through your code was a pleasure as it is really well written in a good and modular style.
You also seem to have a good hold on python and Deep Reinforcement learning.

You have handled all the edge cases elegantly, I am quite impressed!

I would also like to provide a suggestion here: You are sharing the same network across your 20 agents (which is a good thing to do). Now what happens in case when a few/some agents are in same state?

They will start behaving exactly the same for the coming time steps. Now, in this case we will not be able to explore efficiently. Now, there are two solutions:

- We have different networks for each agent: But this wouldn't be a very good idea because each of our agent are exactly the same, so having different networks wouldn't bring efficiency
- We share the same network but have different noise processes for each of our agent: This will ensure that we exploit the most by sharing data achieved from all the agents and at the same time explore the maximum by incubating different random noise processes.

I hope you got my point.

For further learning you might be interested in this.

Congratulations on successfully completing the project.

Cheers!

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 code is written in pyTorch and Python 3. Well Done.
You might be interested in this and this to understand differences between pytorch and tensorflow.

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).

Good Job! The state and action spaces has been described well in the Readme file.

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 (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.

Well Done! The learning algorithm, the hyper-parameters and the model architecture are clearly explained in the report.

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.

Good Job here, the agent was able to solve the environment in a decent number of episodes! However, you were supposed to attach the plot of rewards in the report itself. Since you have mentioned about the plot in the report and the plot is actually present in the `Continuous_Control.ipynb` file, I will give you a green flag here.

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

Good Job here! You have provided very good future ideas on how to improve the agent's performance. It would be really interesting to see how the agent performs on the suggested implementations. As a additional challenge, you should try them out and see which one performs better and maybe report them in your repo!

DOWNLOAD PROJECT

RETURN TO PATH

Rate this review