UDACITY

Return to "Deep Reinforcement Learning Nanodegree" in the classroom

Collaboration and Competition

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

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

Some good resources to refer:

- The Coopetition Dilemma: Building Reinforcement Learning Agents that Learn to Collaborate and Compete at the Same Time
- Competition and collaboration among fuzzy reinforcement learning agents
- Learning to Cooperate, compete and communicate
- A list of best deep learning books to help you in your further study
- Another list of Hands-on books on AI-ML-DL by an Udacity Alumni which featured in #Udacity_EdTalk

Some frontiers in RL

- Distributional RL
- Model-based RL
- UNREAL Agent

Training Code

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

Well documented code.  Some good resources to make it further better:

- Documenting Python Code: A Complete Guide
- Google Python Style Guide

The code is written in PyTorch and Python 3.

Now that you have mastered PyTorch you can learn about tensorFlow:

- **Sebastian Thrun on TensorFlow**
- TensorFlow Vs PyTorch: Which Framework Is Better For Implementing Deep Learning Models?
- PyTorch vs Google TensorFlow – Which AI will take over the world?

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

Saved model weights are included.

README

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

`Readme.md` is included.

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

 It is great that you described the environment details in your `Readme.md`. You have mentioned all the important points identifying the environment viz:

- **State Space**: continuous or discrete; state variables/size.
- **Action space**: Continuous/discrete; action variables/size
- Agent/s Goal
- When the environment is considered solved.
- Is the environment episodic or not?

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

Well documented `Readme.md` you have given clear instructions for downloading the relevant environment files. it would be great if you also include dependencies in the `Readme.md`.

Some resources to make your Readme world class:

- Readme-template
- How and why to write a good Readme

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.

The instructions to train the network are included. Good job 👏

Suggestion

It would be good if you also add the code to test the trained agent. It will very similar to the random agent playing, but this time agent weights will be that of the trained one.

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.

`Report.md` is included.

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

You have mentioned the algorithm, hyperparameters and model architecture of Actor and Critic. I really liked your approach of using reward value ti fill the replay buffer. I would suggest you to explore it further, it might be helpful in training the agents in general. 🏆

You can further improve your report by adding the following points:

- Why you choose the MADDPG algorithm?
- Did you used replay buffer, epsilon-greedy etc approaches? (Also are both agents sharing same experience buffer or different)
- Is the noise added to agents?
- Why you have chosen these particular hyperparameters (like you took from the paper, or result of your own experimentation)?
- Why you chose the particular model architecture for Actor/Critic?

A plot of rewards per episode is included to illustrate that the agents get an average score of +0.5 (over 100 consecutive episodes, after taking the maximum over both agents).

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

Great work in achieving an average score of +0.5 (over 100 consecutive episodes, after taking the maximum over both agents).

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

Great idea, indeed changing reward value has a significant impact on agent training.
Few more future ideas to explore:

- Use parameter space noise rather than noise on action. https://vimeo.com/252185862https://github.com/jvmancuso/ParamNoise
- We can use prioritised experience buffer. https://github.com/Damcy/prioritized-experience-replay
- Different replay buffer for actor/critic
- Try adding dropouts in critic network
- Turn off OU noise and use random noise

DOWNLOAD PROJECT

RETURN TO PATH