UDACITY

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

DISCUSS ON STUDENT HUB

Navigation

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

Congratulations in meeting the specifications of the project! This project had discrete action space. Next, you will work with continuous action control space of the environment.
I hope that you're enjoying this nanodegree and excited to move ahead! Good luck!

Training Code

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

You can refactor your codes into Python files and run it as a program from the command line terminal. From my experience, jupyter notebooks can be slow! You can also then add hyper-parameters arguments by the use of the argparser in order to easily tune the values of the hyper-parameters through command line. Check out this tutorial on argparse https://www.youtube.com/watch?v=rnatu3xxVQE

Check out this blog https://www.pyimagesearch.com/2018/03/12/python-argparse-command-line-arguments/

The code is written in PyTorch and Python 3.

Pytorch is a dynamic computationally while tensorflow has been static. But recently tensorflow has been updated with eager execution making it easier to use i.e. without building computational graph to run the tensorflow sessions. All the deep learning frameworks are turning into the same page. However Pytorch is still Udacity's favorite because Pytorch performs better than the tensorflow eager execution for rapid prototyping as Pytorch is easier to read and understand than Tensorflow. Check out this article https://medium.com/@yaroslavvb/tensorflow-meets-pytorch-with-eager-mode-714cce161e6c :)

There's a new Pytorch Udacity course that you may want to check out https://www.udacity.com/course/deep-learning-pytorch--ud188

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

Saved model weights enable to run the agent during inference without re-training.

README

The GitHub (or zip file) 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).

Nice work in stating whether the action space is discrete (and not continuous) as it is important to choose the learning algorithm depending on it.

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

These steps will help anyone who is interested in running your project. It is also important to document all these steps if you want to re-run your codes again in the later time let's say after a year or so and may forget these steps if these are not documented.

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.

Along with writing the report, I'd also suggest you to write a blog post on this project like on Medium https://medium.com/ which is easier to write on. It will not take you much longer into your write-up and also include the tips given in this review. :D This is a great way to contribute to the AI community by sharing the knowledge.

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.

Nice work in describing the Q-learning algorithm and the architectures. Here's another udacity course lesson video where overview of Q-learning is given out. Q-table is also given out

https://www.youtube.com/watch?time_continue=94&v=WQgdnzzhSLM

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

Nice work in providing the details to your experiments. 👏🏼 It is important to document how did you tuned the values of the hyper-parameters which helps to gain the intuition about how/which the hyper-parameters values influence the performance of the deep reinforcement learning agent given the environment.

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.

The reward curve is quite noisy. Nice work in smoothing out any fluctuations in the reward plot, displaying the general trend of the curve.

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

I'd also highly recommend to use Convolution Neural Nets in your Q-learning's neural network model when using the image pixel values as state. CNNs preserve the spatial information of the image. You can use the pixel based environment given in the classroom.

DOWNLOAD PROJECT

RETURN TO PATH

Rate this project