Teach a Quadcopter How to Fly

REVIEW

Meets Specifications

Hello,

You've done a great work by implementing the DDPG and to teach quadcopter how to fly. I have provided some feedback on your agent code and the reward function. Please go through it. I am including this resourse which is about DDPG algorithm. It would be nice if you can have a look at it.

Since you have completed this project, Now if you want to know how to apply these techniques to real-world problems. For that, I would advise you to go through Deep Reinforcement Learning for Self Driving Car by MIT. So that you can know more about reinforcement learning algorithms in a broader and real-world perspective. As you asked for more feedback, I have provided them in Suggestion section.

Define the Task, Define the Agent, and Train Your Agent!

✓

The `agent.py` file contains a functional implementation of a reinforcement learning algorithm.

Awesome

- Successful implementation of the DDPG algorithm.
- Actor and critic network are correctly implemented.
- It was good to use target network for both actor and critic network as suggested by the author of DDPG Paper.
- Further using soft updates for the target network was a good choice.

✓

The `Quadcopter_Project.ipynb` notebook includes code to train the agent.

Plot the Rewards

✓

A plot of rewards per episode is used to illustrate how the agent learns over time.

Reflections

✓

The submission describes the task and reward function, and the description lines up with the implementation in `task.py`. It is clear how the reward function can be used to guide the agent to accomplish the task.

Awesome

- Task and Reward function are explained properly.

Suggestion

Try using a small number of units and compare the performance. For example:

- [128, 128, 128]
- [128, 64, 128]
- [64, 128, 256]

Use batch normalization for all the layers. Deep networks are prone to gradient vanishing and exploding problems.

✓

The submission provides a detailed description of the agent in `agent.py`.

Awesome

- Good job by choosing DDPG algorithm for continuous state space problem.
- Good job by including hyperparameters and network architecture in the notebook.

Suggestion

Exploration parameters are very important for a good learning. Please refer to this Link to know more about the exploration strategies in reinforcement learning.

✓

The submission discusses the rewards plot. Ideally, the plot shows that the agent has learned (with episode rewards that are gradually increasing). If not, the submission describes in detail various attempted settings (hyperparameters and architectures, etc) that were tested to teach the agent.

Awesome

- It is quite common to see agent not learning ideally in `Quadcopter` project since it is not an easy project. And you have done a good job so far.
- Good job providing the details of various hyperparameters and reward settings.

Suggestion

- You can add z-axis velocity in reward function to encourage quadcopter to fly towards the target.

- You can subtract angular velocity from the reward to make sure quadcopter flies straight up.
- You can include some large bonus and penalty rewards also. Such as a bonus on achieving the target height and a penalty on crashing.
- Clip your final reward between (-1, 1). It will definitely help in better performance.

✓

A brief overall summary of the experience working on the project is provided, with ideas for further improving the project.