UDACITY

Back to Machine Learning Engineer Nanodegree

# Teach a Quadcopter How to Fly

REVIEW

CODE REVIEW

HISTORY

SHARE YOUR ACCOMPLISHMENT!

## Meets Specifications

Dear Udacian,

I have to say that the project is a fairly complicated one and you have done a great job. The task of coming up with the task and choosing the appropriate algorithm to implement the agent is in itself a challenge. Then, formulating a suitable reward function is a very innovative challenge and requires a lot of experimentation.
I have to say that it's not uncommon to see the agent not learning properly even after a lot of experimentation. Glad that you gave your best and made the project work successfully. Overall, you have done a great job!

I suggest you to check this blog post. It's very good.

I would suggest you to go through Deep Reinforcement Learning for Self Driving Car by MIT. You'd get to know more about reinforcement learning algorithms in broader and real-world perspective and, more importantly, how to apply these techniques to real-world problems.

All the best for the future projects. ✨✨

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

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

#### Awesome

The `agent.py` contains a functional implementation of a reinforcement learning algorithm.
The algorithm implemented here is Deep Deterministic Policy Gradient which is found to perform very well in case of continuous action spaces.

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

#### Awesome

Good job implementing the code to train the agent for the task `take off`. It's been very neatly presented in the notebook `Quadcopter_Project.ipynb`.

### Plot the Rewards

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

#### Awesome

Thanks for providing the plot of rewards for the task `take off` 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

Good job implementing the task `take off`. The notebook contains detailed description of the task and the formulated reward function.
Good job experimenting with the reward function. The reward function you have used makes sense.

## Suggestions

The performance of the agent is very sensitive to the reward function. I would like to suggest you to try the following reward function (explained with comments) also.

```
Zdist = abs(self.target_pos[2]-init_pose[2])

# We want minimum change in direction of x and y and making z close to the target z
# so we punish for moving in directions x and y and give rewards to direction in z
punish_x = abs(self.sim.pose[0] - self.target_pos[0])
punish_y = abs(self.sim.pose[1] - self.target_pos[1])
reward_z = 1.0 - (self.target_pos[2] - self.sim.pose[2])/self.Zdist
punish_rot1 = abs(self.sim.pose[3])
punish_rot2 = abs(self.sim.pose[4])
punish_rot3 = abs(self.sim.pose[5])
reward = reward_z - 0.1 * (punish_x + punish_y) - 0.1 * (punish_rot1 + punish_rot2 + punish_rot3)
return reward
```

Request you to try this. 🌟

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

## Awesome

Great job providing the details of the implemented agent. The answer describes the learning agent by specifying the details of the learning algorithm used, hyper-parameters and architectural information of the deep learning model used.

**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

- A great deal of information has been provided.
- I have to say that the project is a fairly complicated one. It's not uncommon to see the agent not learning gradually even after a lot of experimentation.
- The reward plot shows improvement and seems to be quite good. Plot shows sudden improvement with an "aha moment". Thanks for putting in the effort to make it work.

## Further work possible

I would like to encourage you to try and implement the `hover` task also where the objective would be to make the agent learn to fly at the desired height for a few seconds.

If you try the `hover` task, the following suggestion should be very useful.
To make the agent `hover` it could mean to make the quadcopter fly close to the `target height`. Positive reward could be given when the agent flies within a range of, let's say, `2 meters` above and below the `target height`. And negative reward if the agent crosses that limit. This should help the agent to fly near the `target height`.

Request you to try this. 🌟

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

## Awesome

- Thanks for providing the details.
- The project is indeed a hard one and requires the learning and combining of a lot of concepts.
- A good reward function is important to train the agent properly. Good job working on the reward function.

- It is requested to check: Deep Reinforcement Learning Models: Tips & Tricks for Writing Reward Functions.

DOWNLOAD PROJECT

RETURN TO PATH

Student FAQ