How long should I expect my agent to train?

Depending on the random initialization of parameters, sometimes your agent may learn a task in the first 20-50 episodes, but you can expect most algorithms to be able to learn these tasks in 500-1000 episodes. It is also possible for them to get stuck in local minima, and never make it out (or make it out after a long time). It's possible for your training algorithm to take longer, e.g. depending on the learning rate parameter you choose.

If you see no significant progress by 1500-2000 episodes, then go back to the drawing board. First make sure your agent is able to solve a simpler continuous action problem, like Mountain Car or Pendulum (yes, your model sizes will be different based on the state/action space sizes, but this will at least verify your algorithm implementation is working).

Next, tweak the state and/or action space definitions in the task to make sure you're giving the agent all the information it needs (but not too much!). Finally (this is the part that takes the most number of iterations), try to change the reward function to give a better and more distinctive indication of what you want the agent to achieve. Play with the weights of the individual metrics, add in an extra bonus / penalty, try a non-linear mapping of the metrics (e.g. squared or log distance), etc. You may also want to scale or clip the rewards to some standard range, like -1.0 to +1.0. This avoids instability in training due to exploding gradients.

My agent seems to be learning, but even after a lot of episodes, it still doesn't impress me. What should I do?

Getting a reinforcement learning agent to learn what you *actually* want it to learn can be hard, and very time consuming. It'll try to learn the optimal policy according to the reward function you've specified, but that may not address all aspects of the behavior desired by you. E.g. in the Takeoff task, if you don't penalize the agent for twisting around, and only care how high it's going, then it's perfectly fine for it to whirl around while going up!

Sometimes, your algorithm might not be tuned correctly for the environment. E.g. some environments may need more exploration or noise, some may need less. Try changing the these variables, and see if you get any improvement. But even if you

This is why we ask you to plot the episode rewards. **As long as it shows that the average reward per episode is generally increasing (even with some noise), that's okay. The final behavior of the agent doesn't need to be perfect.**

Am I allowed to use a Deep Q Network to solve this problem?

Yes! But ... note that a Deep Q Network (DQN) is meant to solve discrete action space problems, whereas the quadcopter control problem has a continuous action space. So a discrete space algorithm is not advised here, but you could still use one by mapping the final output of your neural network model to the continuous action space appropriately. This may make it harder for the network to understand how its actions are related to each other, but it may also simplify the learning algorithm needed to train it.

NEXT