COMP 150 RL: Homework 4

Billy Witrock

November 19, 2019

1 Introduction

This homework is about laying out the baseline for the final project. The goal is to find an algorithm that works on the desired environment. To understand this homework, first we must understand the final project. The final project is to tests how switching back and forth between harder and easier tasks can help train the agent in a faster matter. The algorithm the agent uses is not as important as finding an algorithm that can solve the problem to a resonable extent.

2 Methodology

First, I let me discuss the ideal enviroment. The ideal environments are the "BipedalWalker-v2" and "BipedalWalkerHardcore-v2" from Openai's gym. These environments are very similar which leads us to believe it is a perfect candidate. The goal in both is to get the simulated robot to move forward without hitting the ground. Positive reward is given as the robot moves forward, and negative reward is given if it moves backwards. A maximum of 300 reward is given if it makes it to the end, and a reward of -100 is given if it hits the ground. The observation is also the same in both cases. The robot receives information on the location, and speed of its body parts, as well as, some sensor information that allows the robot to see in front of itself. Since, the observations, and rewards are the same, it makes it the best environments for this experiment.

For this homework the main goal was to create an algorithm to score high reward in the easy base case. To do this, we created a sarsa agent, with linear function approximation. However, since the action is an array of 4 floats between -1, and 1, we needed to put a spin on how we used sarsa. In the algorithm we created, we have a separate linear approximation for each spot in the action array. Then we also scaled back the choices in power for each action from changing the possible values from any value in the range of -1 to 1, to any increment of 0.1 between -1 and 1. This allowed us to create a discrete set of actions. So, for each of our 4 different linear function approximations, we had 21 output nodes corresponding to the 21 different values it could be. Then, we selected the one that had the greatest value. Also, the Q function value estimator for each action was the average of each actions value based on its approximation. Then 1800 episodes with a max of 500 steps per episode were run to train the agent.

3 Experiment and Results

The results were not good of this experiment. The agent was unable to gather any knowledge. Most of the time the agent found a policy in which it stood straight up and did not move forward at all. This policy minimized negative reward in the short run but did not achieve any long term reward. Also, almost every trail would end with the agent either falling backwards, or forwards both which causes great negative reward.

A possible reason for this failure could be the hyperparameters chosen. Even though more than one set of hyperparameters were tested, none of the trails led to success. Another possible reason for failure is the fact that even though all the actions are related, each one was chosen independently of the rest. Also, there was no memory. This means that even if an action was not bad, but was forced into failure because of a previous action, it would be the good action that was penalized.

4 Conclusion

From this homework it is clear a lot more work is needed. First, a new algorithm needs to be selected. Possibly deep Q-learning or a policy network. Also, the newly selected policy should try to work with continuous action spaces. The fact that we are reducing the amount of available actions could be a problem if optimal policies involve actions being cut. Also, have an action replay system or some sort of n-step return to account for the actions in the past having an effect is critical.

Once, a better algorithm for this problem is selected, tested, and ran we will be able to move on the rest of the project. This homework was a good awakening that an optimal algorithm could be very difficult to find but is needed before conducting any other experiments. In this upcoming weeks I will try to find the algorithm and then run baseline tests. All code is on github at https://github.com/BillyWitrock/Reinforcement_Learning_final.