Report for Project 3 – Collaboration and Competition

Problem

In this project, the goal is to train two agents to control rackets to bounce a ball over a net. The agent receives a reward of +0.1 if it hits the ball over the net. Otherwise, if an agent lets a ball hit the ground or hits the ball out of bounds, it receives a reward of -0.01. Each agent will try to keep the ball in play. The observation space consists of 8 variables corresponding to the position and velocity of the ball and racket. Each agent receives its own, local observation. Two continuous actions are available, corresponding to movement toward (or away from) the net, and jumping. The criteria for solving the problem is to achieve +0.5 scores averaged over the latest 100 episodes, after taking the maximum score over both agents.

Approach

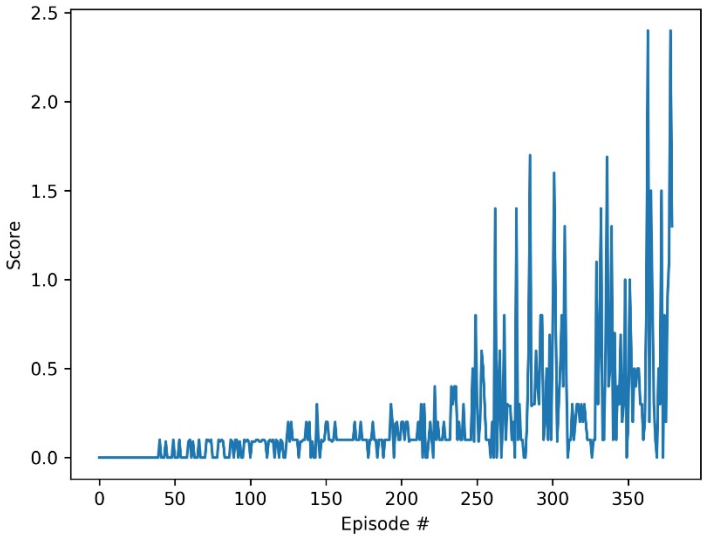
To solve the problem, I adapted the Deep Deterministic Policy Gradient (DDPG) method from the paper [1] to train two agents that collaborate and compete with each other. The experiences collected by both the agents are pushed to a common replay buffer, which is used for the update for both the agents. The algorithm implements two networks, namely the actor and critic networks.

The actor neural network consists of 2 hidden layers that have 512 and 256 neurons respectively. For each hidden layer, it is followed by the Relu nonlinearity. The final output layer has two neurons 2 output neurons and uses tranh as the activation function. To encourage exploration, the output value of the action is added by an Ornstein-Uhlenbeck noise. On the other hand, the critic neural network has two hidden layers and one output layer. The first hidden layer has 512 neurons and takes the state as input. Then the second hidden layer takes as input from both the actor network and the first hidden layer. The activation function for each hidden layer is Relu.

After tuning the hyper-parameters, the final set of parameters I choose that achieves the best performance is as follows. The learning rate is 1e-4 for the actor network and 3e-4 for the critic network. The reward decay factor is 1 (without discounting). The weights of the actor and critic networks are soft-updated 10 times every 5 time-steps. The soft-update coefficient is 1e-1. The weight decay is 0 (no regularization). The batch-size is 512 and the replay buffer size is 1e5. The maximum time steps for each episode is 1500.

Result

By implementing the algorithm described above, I was able to solve the problem after ~380 episodes. The scores are plotted as a function of the episodes, as shown in the figure below. The training takes ~1 hour to finish on one NVIDIA GTX 1080 Ti GPU. It can be seen that the reward for each episodes stuck at zero during the initial stage of training and fluctuate for some time in the middle of training. Towards the end of training, the scores suddenly increase, but still have large variances.



Future directions

In the future, the following approaches can be tried to further improve the performance of the learning algorithm and the agent.

First, we can try the recent Distributed Distributional Deterministic Policy Gradients algorithm (D4PG) that demonstrates the state-of-the-art performance in continuous control tasks [2]. Second, we can use priority experience replay instead of purely random experience replay. By doing this, we can increase the sampling probability of important experiences, which can hopefully boost the learning process.

Reference

[1] Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D. and Wierstra, D., 2015. Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971.

[2] Barth-Maron, G., Hoffman, M.W., Budden, D., Dabney, W., Horgan, D., Muldal, A., Heess, N. and Lillicrap, T., 2018. Distributed Distributional Deterministic Policy Gradients. arXiv preprint arXiv:1804.08617.