Multi-Agents Reinforcement Learning for Tennis Game

Abstract

A DDPG algorithm was used to solve the “Tennis” game which was simulated in Unity. The agent contains two parts: actor network and critic network and can attain +1.0 scores on average after hundreds of episodes. In this game, two agents were required to collaborate to bounce the ball as long as possible, which is preventing the ball fall on the ground or out of boundaries.

The DDPG contains two parts: the actor network and the critic network. The actor network was built up with two hidden layers, 128, 128 neurons respectively and each hidden layer was followed by a batch-normalization layer. The critic network was built up with two hidden layers, 128, 128 neurons respectively.

From the implementation, we found that agents learn quite slowly at first, and start to improve drastically after hundreds of episodes. It might be an interesting issue to explore how to make agents learn efficiently. Our final result can attain +1.0 scores after around 500 – 700 episodes, which may depends on different random seed that affect weights initialization and the game initialization.

Project goal

In this project, we’re going to use policy-based or mix way, or namely the DDPG, to train agents due to deep-Q network is unable to solve continuous action space

problems. The goal of agents were to collaborate to bounce the ball as long as possible.

Method and experiments

Agent information

- State: 24 dimensions include the position and velocity of the agent's racket and the ball. Since each agent only has its only local observation, the total state size is 48.
- Action: each agent has two continuous actions, one is the movement toward/away the net, and the other one is jumping. The value of each action is ranged from -1 to +1.
- Reward:
  - bounce the ball: +0.1
  - the ball fall on the ground or agent hits the ball out of bounds: -0.01

Model

Main structure: multi-layer perceptron network.

- Actor network
  - 128 → 128 → 2
- Critic network
  - 128 → 128 → 1
- Batch size: 128

- Optimizer: Adam, learning rate: 1e-3 for both the actor and the critic network.
- Update per action. Soft-update (switch local net parameters to target net parameters) per 20 update. The TAU was set on 1e-3.

Result

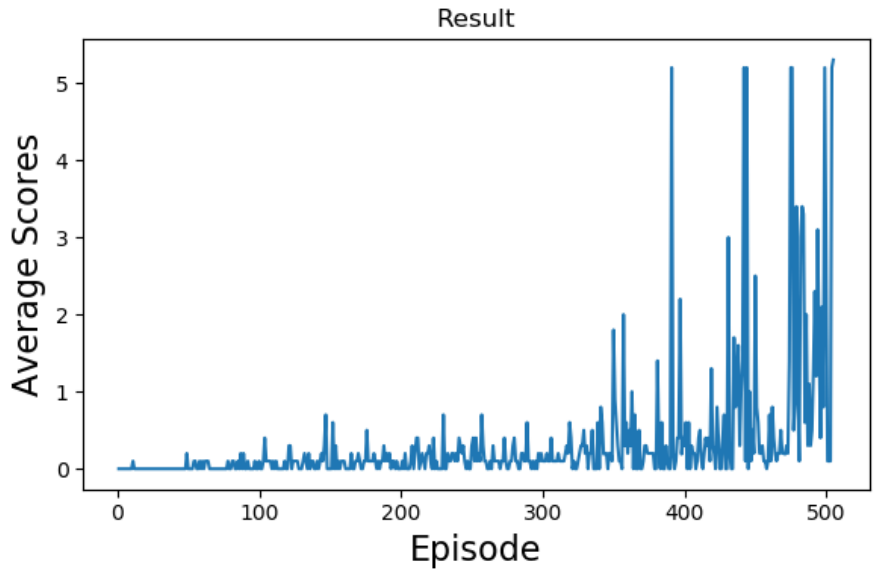

Figure1. Rewards of the solution, 2 layers.

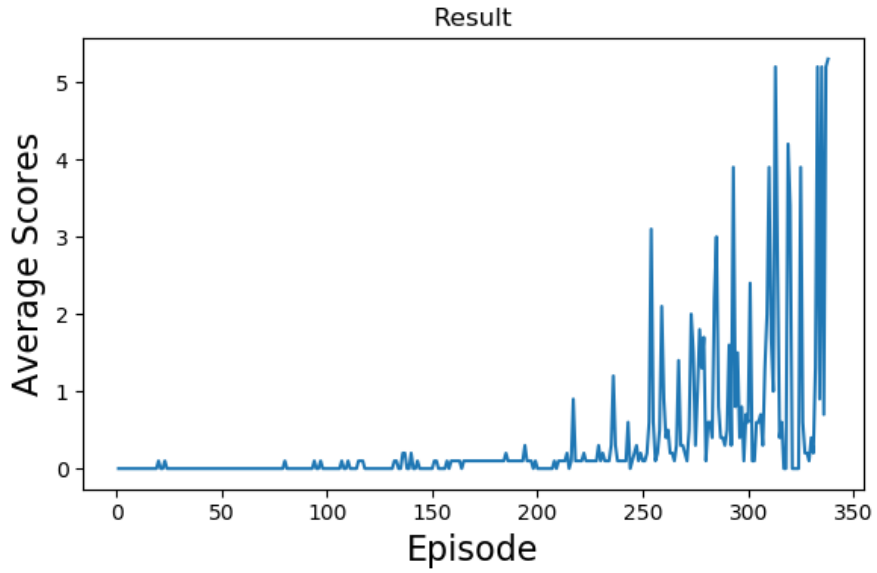

Figure2. Rewards of the solution, 4 layers.

Discussion (and future ideas)

The DDPG algorithm was used to solve the Unity tennis game. According to the training process and results, the agents can learn to collaborate in a not very efficient manner. It showed that the scores at begin growth slowly but gain large improvement after hundreds of episodes. Interestingly, agents with deeper hidden layers in both actor and critic networks seem to make the agent “wake up” earlier, suggesting that the two layers network haven’t captured all non-linearity of the system. However, stacking more layers will harm the agents’ performance, it is intrigue to know how many layers is required to optimize different environments/systems. Besides, the weights initialization is a critic issue for agents which strongly affect the learning efficiency in all deep neural networks. From previous experience however, we found that the difference between he_norm and ranom_init has little difference in the performance. We’re intrigue to know whether there is another way to initialize weights of layers that fit the deep reinforcement network.