Solving a Collaboration-Competition Environment by Multi-Agent Deep Deterministic Policy Gradient

Xiongwen Qian

1. Problem

The problem is about two agents (players) bouncing a ball over a net. If one agent bounces the ball over the net, it receives a reward of +0.1. If the ball is hit to the ground or out of bounds, it gets a reward of -0.01. Thus, each agents tries to keep the ball in play so as to get a high reward.

2. Learning Algorithm

A Multi-Agent Deep Deterministic Gradient (MADDPG) algorithm is used to solve the problem.

a) Algorithm Overview

MADDPG can be viewed as an extension of the DDPG algorithm which is used for the single agent case. The Tennis environment involves two agents (players). Each agent has an actor network and a critic network. The input to the actor network is the agent's perceived state which includes the position and velocity of the ball and the agent's racket. The input of the critic network consists of not only the agent's (player's) state and action, but also the other agent's (player's) state and predicted action. In other words, an agent sees (knows) the state of the other agent, predicts the other agent's action by its own actor network, and finally evaluates the Q-value of the situation (state-action pair) based on both agents' states and (predicted) actions. In the program, an MADDPG agent is designed to coordinate the saving of the experience (both agents' states and actions) and the training of the networks.

b) Neural Network Structure

For the actor's neural network, the input is a state vector of 24 units, and 2 hidden layers are involved. The first hidden layer has 400 units, and the second has 300 units. The output layer has 2 units corresponding to the values of the 2 actions. Batch normalization is used after the input goes through the first hidden layer. It transforms the output of the first hidden layer into standard ranges. The activation function is Relu except that the last layer's activation function is tanh.

For the critic's neural network, the input is a concatenation of both agents' states and actions. So the input vector contains 2*24+2*2=52 units. Batch normalization is also used. The first hidden layer contains 400 units and the second hidden layer contains 300 units. The output layer has one unit, i.e. the final Q-value.

c) Hyperparameters

Max number of episodes = 10000

Max number of steps per episodes = 1000

Number of episodes between learning process = 4

Number of multiple learning process performed in a row = 3

Replay buffer size = 10000

Minibatch size = 200

Learning rate of the actor = 0.0004

Learning rate of the critic = 0.001

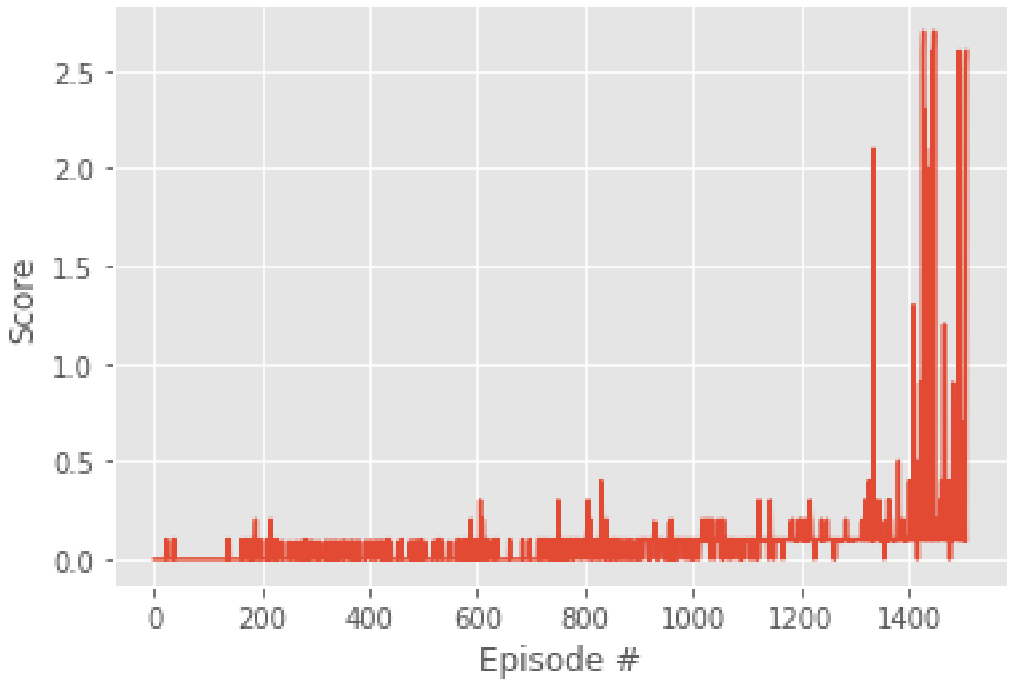
Discount factor γ = 0.995

Parameter for soft update = 0.001

Noise = 0

3. Result

The environment is solved in 1507 episodes.



4. Ideas for Future Work

After some experiments, it turns out that when noise is not added to the action, the whole training process takes shorter. However, the generalization capability of the trained network seems to be weaker. Sometimes agents do not achieve great result in the test after the networks are trained. In future, the hyperparameter Noise can be further optimized by grid search.