Report of
Project 3 : Collaboration & Competition

Author : Saoussen Chaabnia

Project Description :

For this project we will use Multi-Agent Deep Reinforcement Learning to teach two agents control rackets to bounce a ball over a net.

In this environment, two agents control rackets to bounce a ball over a net. If an agent hits the ball over the net, it receives a reward of +0.1. If an agent lets a ball hit the ground or hits the ball out of bounds, it receives a reward of -0.01. Thus, the goal of each agent is 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 task is episodic, and in order to solve the environment, your agents must get an average score of +0.5 (over 100 consecutive episodes, after taking the maximum over both agents). Specifically,

After each episode, we add up the rewards that each agent received (without discounting), to get a score for each agent. This yields 2 (potentially different) scores. We then take the maximum of these 2 scores.

This yields a single **score** for each episode.
The environment is considered solved, when the average (over 100 episodes) of those **scores** is at least +0.5.

Implementation

Actor-Critic method is used to solve this problem. Actor-Critic method is at the intersection between Policy-Based-Methods and Value-Based-Methods. An Actor-Critic uses function approximation to learn a policy and a value function. This achieved by using two neural networks , one for the actor and one for the critic. The actor is a neural network which updates the policy and the critic is another neural network which evaluates the policy being learned which is, in turn, used to train the actor.

For our solution, we adopted the **Multi Agent DDPG (Multi-Agent Deep Deterministic Policy Gradient (MADDPG))** algorithm is our implementation. The actor is used to approximate the optimal policy deterministically, outputting the best action for any given state. The critic learns to evaluate the optimal action-value function by using the actor best believed action.

A framework of centralized training with decentralized execution is at the heart of this solution. Thus, we allow the policies to use extra information to ease training, so long as this information is not used at test time. In this algorithm, the critic is augmented with extra information about the policies of other agents. Such observation could be states observed and actions taken by all other regions. There is one actor for each agent. Each actor has access to only its agent observation and actions. During execution time, only the actors are present. Learning critic for each agent allows for the use of different reward structure for each agent.

Some techniques contributed significantly towards stabilizing the training :

Experience Replay: In Experience Replay, we maintain a Replay Buffer of fixed size in which we store some experience tuples as we interact with the environment. After a fixed number of iterations, we sample a few experiences from this replay buffer and use that to calculate the loss and eventually update the parameters. Sampling randomly this way breaks the sequential nature of experiences and stabilizes learning. It also helps us learn from an experience multiple times and recall rare occurrences. Using Experience replay, the value based learning of the reinforcement learning problem is reduced to a supervised learning problem.

Fixed Targets: The idea behind fixed q-targets is to decouples the target from the parameters by fixing the parameters w used to generate the target that we call w-. w− are the weights of a separate target network that are not changed during the learning step. This helps stabilize training. Both the actor and critic are neural networks in which the fixed target is used.

Soft Updates: The target network are updated by mixing the 0.01% of the local network weights with the target network weights that are retrained during each update step.

Hyperparameters:

- **n_episodes** : 2000

- **m_tax** : 1000

- **BUFFER_SIZE** = int(1e6)

- **BATCH_SIZE** = 128

- **LR_ACTOR** = 1e-3

- **LR_CRITIC** = 1e-3

- **WEIGHT_DECAY** = 0

- **LEARN_NUM** = 5

- **GAMMA** = 0.99

- **TAU** = 8e-3

- **OU_SIGMA** = 0.2

- **OU_THETA** = 0.15

- **EPS_START** = 5.0

- **EPS_EP_END** = 300

- **EPS_FINAL** = 0

The Deep Learning Model:

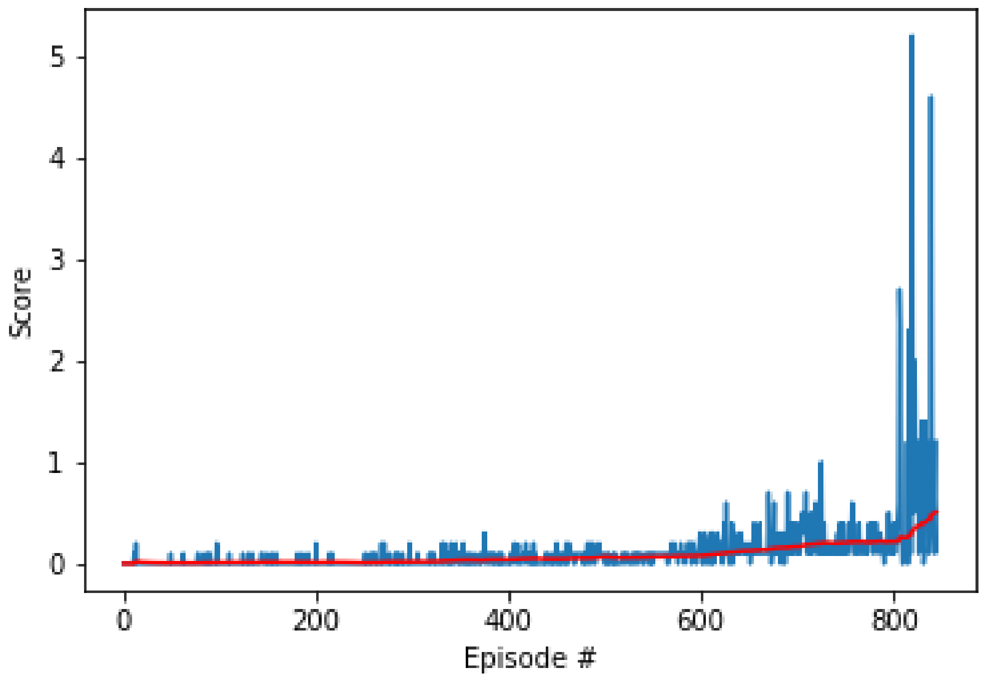
Actor :

Our actor model constitutes of a two-hidden layers neural network. Layer 1 has 256 hidden units and layer two has 128 hidden units, with ReLu activation function applied after each fully-connected layer. A tanh function is applied to output layer since the entry action are between -1 and 1.

Critic :
The critic model has 2 hidden layers of 256, 128 hidden units respectively. ReLu is applied to each of them.

Result:

The environment was solved in **745** episodes using the **Multi Agent DDPG** solution.



Improvement :

The results can be improved by using the following methods:

- More hyper-parameters Tuning
- Using the Prioritized Experience Replay Learning Algorithm
- Using Asynchornous Actor Critic Agent Learning Algorithm