Environment Introduction

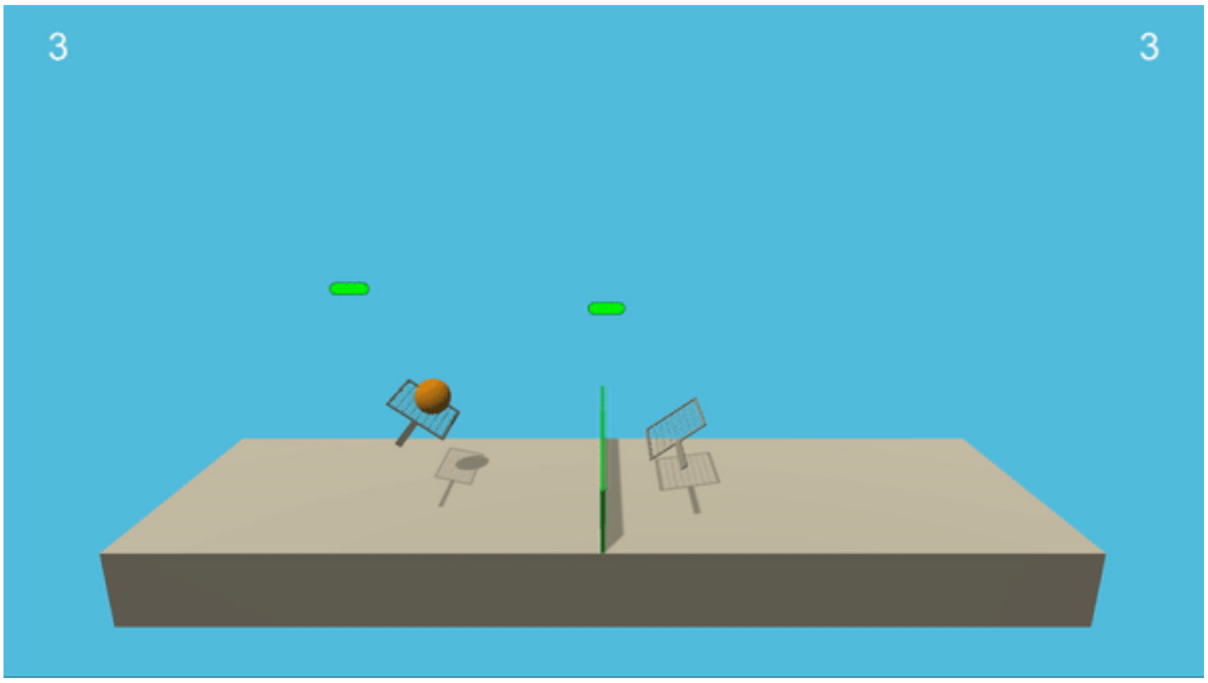
In this environment, two agents control rackets to bounce a ball over a net. If an agent hits the ball over the net, it receives a reward of +0.1. If an agent lets a ball hit the ground or hits the ball out of bounds, it receives a reward of -0.01. Thus, the goal of each agent is to keep the ball in 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.



Solving the Environment

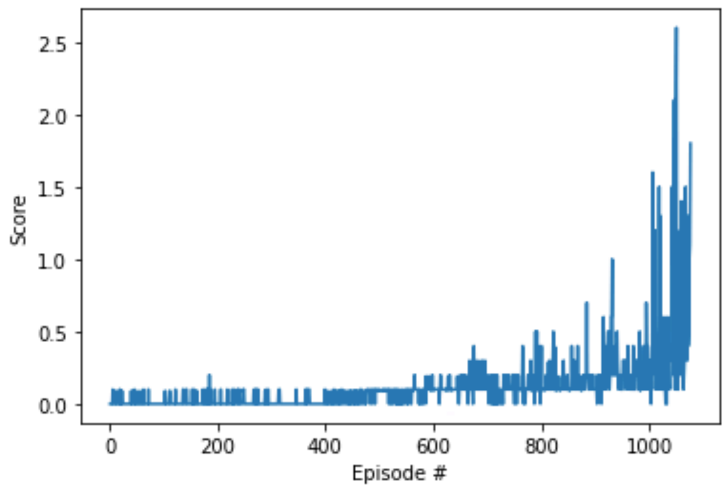
The algorithm used here is a Multiagent Deep Deterministic Policy Gradient (MADDPG). A DDPG is composed of two networks: one actor and one critic. During a step, the actor is used to estimate the best action, the critic then use this value as in a DDQN to evaluate the optimal action value function. Both actor and the critic are composed of two networks. On local network and one target network. For the MADDPG the critic network is working on set of environment states from all agent as well on join action space from all agent. The critic network is only used during the training phase.

The first trial was done on the agent and model used in p2_continuous-control project with modification to support multiagent. A new class MultiAgents has been introduced which instantiate two separate agent for corresponding player. As mentioned critic network is trained on combined states and actions from the both agent. Beside gradient clipping, batch normalization is also introduced in critic and actor model to stabilize the learning process. The Ornstein-Uhlenbeck process is also modified by additional hyperparameter NOISE_DECAY which decays noise addition over time. Moreover, by considering the initial environment steps in each episode correspond to really small number, the learning process was not initiated until sufficient experiences were filled in the replay buffer. Random actions were performed during the RANDOM_ACT_TILL episodes which helps to capture experiences better and to start training the agent efficiently. With the different combinations atleast it is possible to train the network within 900 – 1000 episodes (first RANDOM_ACT_TILL = 300 episodes no learning step is performed)

- Architecture of Actor Network
    - input size = 24
    - output size = 2
    - 2 hidden layers and one output layer
    - first hidden layer has 256 hidden units and is followed by a ReLU activation layer and then batch normalization
    - second hidden layer has 128 hidden units and is followed by a ReLU activation layer
    - output layer is followed by a tanh activation layer
- Architecture of Critic Network
    - input size = 24(states pair)*2 + 4(action pair)*2
    - output size = 1
    - 2 hidden layers and one output layer
    - first hidden layer has 256 hidden units and is followed by a ReLU activation layer and then batch normalization
    - second hidden layer has 128 hidden units and is followed by a ReLU activation layer
    - output layer is followed by a linear activation unit
- Training hyperparameters
    - Replay Buffer size : 50,000
    - Batch size : 256
    - γ : 0.99
    - τ : 0.001
    - learning rate actor : 0.0001
    - learning rate critic : 0.0001
    - weight decay : 0
    - learning performed per each environment step : 3
    - Random action till 300 episodes
    - Per episode noise decay factor : 0.0005

Plot of rewards

```
Episode 100     Average Score: 0.01     Environment Steps in Episode: 13
Episode 200     Average Score: 0.02     Environment Steps in Episode: 13
Episode 300     Average Score: 0.02     Environment Steps in Episode: 13
Episode 400     Average Score: 0.01     Environment Steps in Episode: 13
Episode 500     Average Score: 0.05     Environment Steps in Episode: 30
Episode 600     Average Score: 0.08     Environment Steps in Episode: 314
Episode 700     Average Score: 0.12     Environment Steps in Episode: 536
Episode 800     Average Score: 0.12     Environment Steps in Episode: 146
Episode 900     Average Score: 0.15     Environment Steps in Episode: 319
Episode 1000    Average Score: 0.23     Environment Steps in Episode: 88
Episode 1076    Average Score: 0.51     Environment Steps in Episode: 689
Environment solved in 976 episodes!     Average Score: 0.51
```



Ideas for future Work

- Prioritized experience replay: Prioritized experience replay selects experiences based on a priority value. This can improve learning by increasing the probability that rare and important experience vectors are sampled.
- Other algorithms: Solving the same environment with other actor critic methods like PPO, A3C or D4PG.