Project Overview

The goal of the project is to train two agents to play tennis. When we play tennis on the ground, the goal is to keep the ball in play for maximum number of times, making high scores, while hitting the ball over the net back and forth. We tried to achieve similar by training the two RL agents.

Details:

Unity Tennis environment was used for this project.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.

Algorithm in use

Multi Agent Deep Deterministic Policy Gradiet was used as a learning algorithm. This was something used in continuation of the Continous Control Project. DDPG algorithm uses stochastic behavorial policy for good exploration and deterministic target policy for estimation. This also provides the update rule for the weights of the actor network. The critic's output is the estimated Q-value of the current state in combination with the action done by the actor. The critic's network gets updated from the gradients obtained by the TD-error signal.

The set of hyperparameters which worked for me for training and achieving the desired results are:

1. BUFFER_SIZE = int(1e6) # replay buffer size
2. BATCH_SIZE = 128 # minibatch size
3. GAMMA = 0.99 # discount factor
4. TAU = 1e-3 # for soft update of target parameters
5. LR_ACTOR = 2e-4 # learning rate of the actor
6. LR_CRITIC = 2e-4 # learning rate of the critic
7. WEIGHT_DECAY = 0 # L2 weight decay

I tried different values for OUNoise related to theta and sigma, but the value which worked was 0.15 and 0.02 respectiveley

Results

The agent achieved Average Score of 0.50 on the episode 1363 and by the end of episode 1500, the average score generated by the agents were 1.65.

Ideas for future work

1. Try different network implementation with different hyperparameters
2. Try algorithms A3C, PPO
3. Batch Normalisation and Drop outs can also be tried.

In [ ]: