Report Udacity P3 Tennis

1 Introduction

In this report I wll describe the implemenation for the P3 Tennis project for the Udacity Nanodegree in Deep Reinforcement Learning(RL). The Algorithm itself will be explained as well as the choice of hyperparameters. The rewards of the successful agent will be documented and possible future improvement will be discussed. The successful algorithm described below is the third version I coded. The first two versions did not manage to yield averages beyond 0.05, even though the second one featured a prioritized experience replay buffer. The third version described below succeeded, though I could not make out the difference to the previous implementations.

2 The MADDPG-Algorithm

The algorithm consists of two Deep Deterministic Policy Gradient(DDPG) agents with experience replay memory(ERM). Both agents receive the full states of both agents and also take into consideration the others agents actions, when learning its own actions. They share a common replay memory but have their individual actor and critic networks.
The agents employ a DDPG algorithm with ERM. Each features reinforcement learning with a 2 neural networks. The Crtic Network approximates the Q-value of a given state and actions. Herein it considers the combined actions of both agents. The Actor Network approximates the best individual action given a state and the others agents action. In the training process the action chosen by the actors network combined with the relevant action from the other agent is the reference for the critic network in predicting the Q-value. The batches chosen come from the common ERM.
Both networks feature soft updating as target networks are updated only by a linear combination with local networks, weighted with factor tau. They consist of input, output, and 2 hidden layers. On each training step a batch of experiences is sampled from the common ERM to update the individual local networks using the policy gradients, performing the soft update mentioned above.
Furthermore, the critic network includes batch normalization layer for improved and more stable learning. The Actor network also includes a noise process which is added to the action chosen by the network.

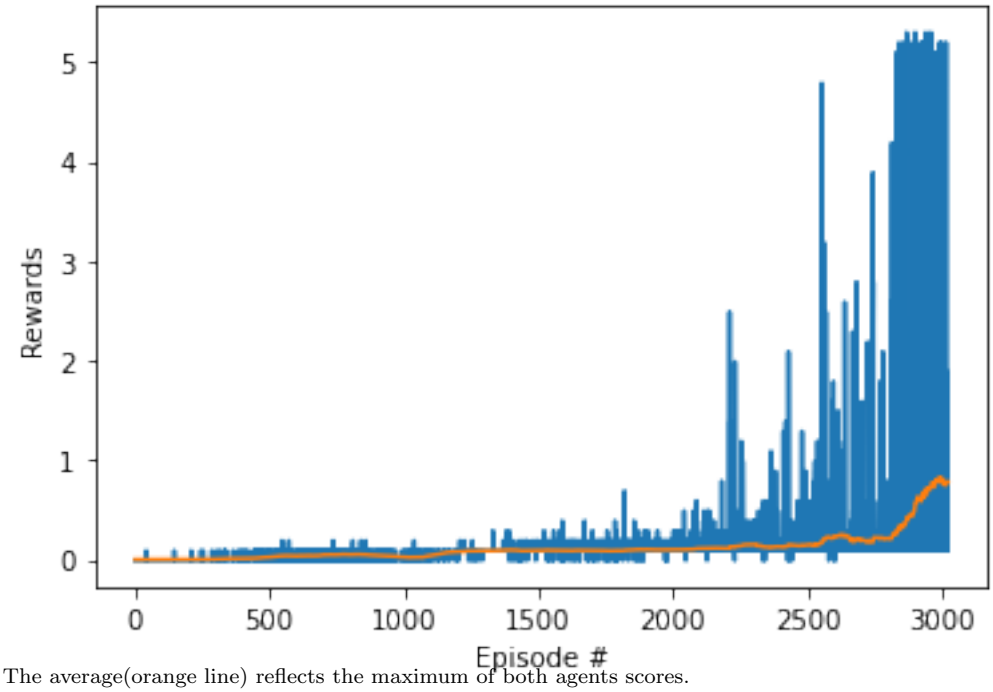
2.1 Choice Hyperparameters

I trained on various combinations of hyperparameters(HP) for extensive time periods and numerous HP combinations. Many did not get beyond 1 hit per racket.The HPs shown achieved the best performance, though the precise conditions for convergence stay still partially unclear. The learning process showed to be strongly dependent on the variance of the noise process and its decay.

Name	Value	Description
gamma	0.99	Discount Factor
tau	0.01	Dampening Factor for updating Target-Networks
LR	0.001	Learning Rate for Actor & Critic Network
batch_size	256	Batchsize of experiences drawn from replay buffer
hidden_size	256/128	Number of neurons in first/second hidden layer
buffer_size	50k	Number of elements in replay buffer
learn_every	1	The number of episodes after which to learn
laern_num	2	Number of learning passes
Ornstein-Uhlenbeck Noise Process		
sigma	1	Diffusion
theta	0.15	Mean Reversion Rate
mu	0	Mean Reversion Level
noise_decay	0.999	Rate of noise decay

3 Agents Performance

With the given hyperparameters in the main training run the agent reached the average score of 0.5 over 100 episodes in episode 2898. Primarily the local optimum at 1 hit per racket showed especially resistent to overcome. The later success on multiple hits per racket also showed to be vulnerable to degradation and instability. The following plot displays the agent scores over 3000 epsiodes:



The average(orange line) reflects the maximum of both agents scores.

The bars in blue reflect the sum of both agent scores, for better differentiation of individiual scores.

4 Discussion for future Improvements

For possible improvements I would like to further explore the precise conditions under which the agent overcomes the local optimum of 1 hit per racket and how the later successful period can achieve more stability. Furthermore I would retry on using prioritized eperience replay memory(PER) and try to discover why the older version including PER did not succeed. Concerning different algorithms, I would like to compare the used DDPG against PPO and A2C, and if one of them learns more stable or more easily overcomes the local optimum.