Overview:

This project controls two tennis racket to hit call.

Reward: +0.1

Penalty: -0.1

Aim is to get the maximum combined score, hence play as long as possible.

Approach:

I am using DDPG to solve this multi agent problem.

Parameters:

- BATCH_SIZE = 128 # Minibatch size
- GAMMA = 0.99 # Discount factor
- TAU = 1e-3 # For soft update of target parameters
- LR_ACTOR = 2e-4 # Actor learning rate
- LR_CRITIC = 2e-4 # Critic learning rate
- WEIGHT_DECAY = 0 # L2 weight decay

I use two linear layer then a batch normalization layer, and then finally a linear layer. Idea is not to make the model too complex so that it can train at a fair speed.

In this model I am using a new type of weight initializer to see if it can make it faster :

torch.nn.init.xavier_uniform(self.fc1.weight)

My Policy network is designed like this:

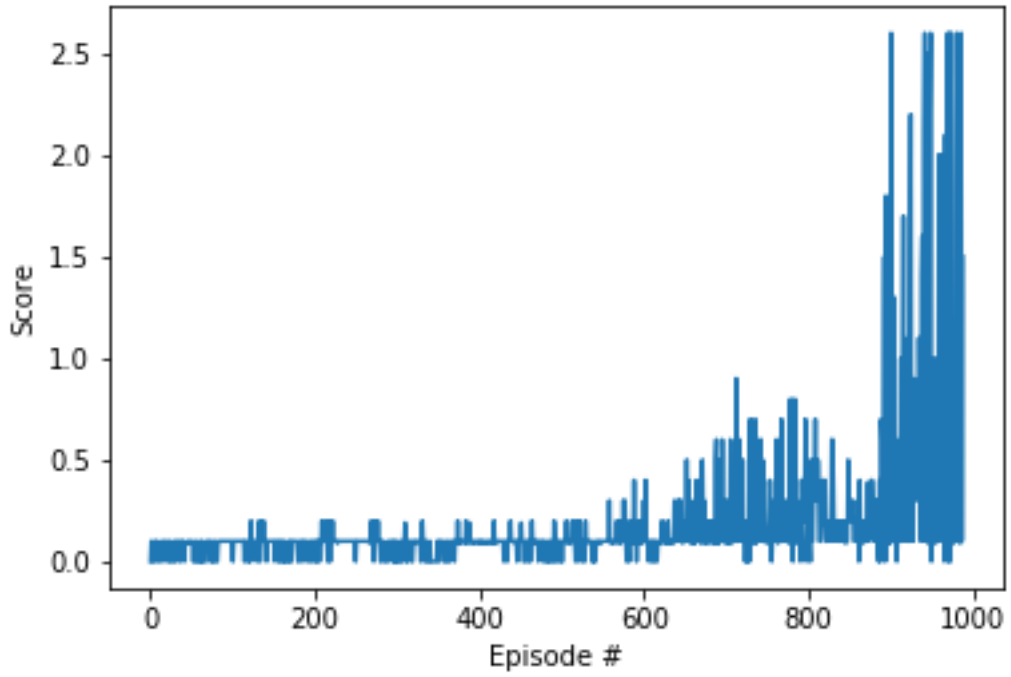
```
self.fc1 = nn.Linear(state_size, fc1_units)
self.fc2 = nn.Linear(fc1_units, fc2_units)
 self.fc3 = nn.Linear(fc2_units, action_size)
```

Hope this explanation helps.

I used A2C model with both the actor and the critic defined from same network.

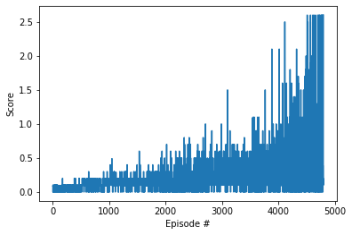
Since the paddle on the right and paddle on left have to play differently, so we have trained the two models for left and right separately. We have come to this conclusion after trying lots of things out starting from a single agent model for both the paddles.

Training Graphs



This was really hard model to train, these are the last 1000 steps but I have actually trained the model for more than 7000 episodes and took me 3 days to do it on my local machine without using GPUs.

In the end model does achieve an average score of over 0.5 for 100 consecutive episodes.



I have managed to recreate the model training graph based on one more training

Ideas to improve further

I am going to try out Priority Experience Replay (PER) but it would be helpful if you can provide any code reference again. I am also willing to try out a Policy based method such as PPO.