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)

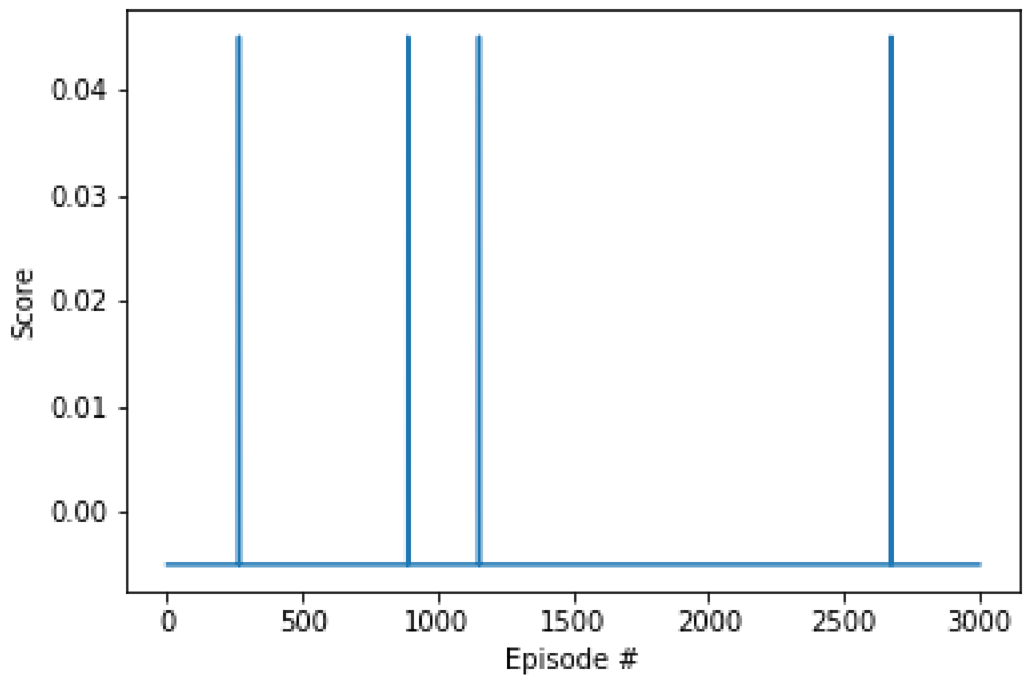
My Policy network is designed like this:

def __init__(self, state_size, action_size, seed, fc1_units=300, fc2_units=200):

- super(Actor, self).__init__()
- self.seed = torch.manual_seed(seed)
- self.fc1 = nn.Linear(state_size, fc1_units)
- self.bn1 = nn.BatchNorm1d(fc1_units)
- self.fc2 = nn.Linear(fc1_units, fc2_units)
- self.bn2 = nn.BatchNorm1d(fc2_units) ## added 2nd batch norm Saurabh
- self.fc3 = nn.Linear(fc2_units, action_size)
- self.reset_parameters()

Hope this explanation helps.

Training Graphs



Ideas to improve further

I am yet to find the ideas to try, please suggest some for me to try out. Since its not a final submission.