Trevor McCants
Project 3 – Tennis
Udacity – Deep Reinforcement Learning

Tennis

Algorithm Description

The Deep Deterministic Policy Gradient (DDPG) method was used to solve this environment. This method uses a neural network to estimate the action value function, Q, and another neural network to estimate the policy. These are referred to as the critic and actor networks respectively.

The specific implementation I used has an epsilon and a sigma value to control exploration. Each time an action is chosen, a small noise value is added to it. This noise comes from the Ornstein-Uhlenbeck process.

A replay buffer was also used to allow the agent the train on uncorrelated experiences each epoch.

Each agent shares the same actor and critic models. Both observations are added each time step to the same replay buffer and the networks are updated.

A local and target network were used to stabilize training with a soft update TAU each time step. These networks are initialized with a hard update step so that the local and target networks start with the same random weights. The Temporal Difference (TD) method was used to calculate Q along with the target critic network. This approach increases bias but decreases noise allowing the network to train more reliably.

Hyperparameters

The following hyperparameters were used in this solution:

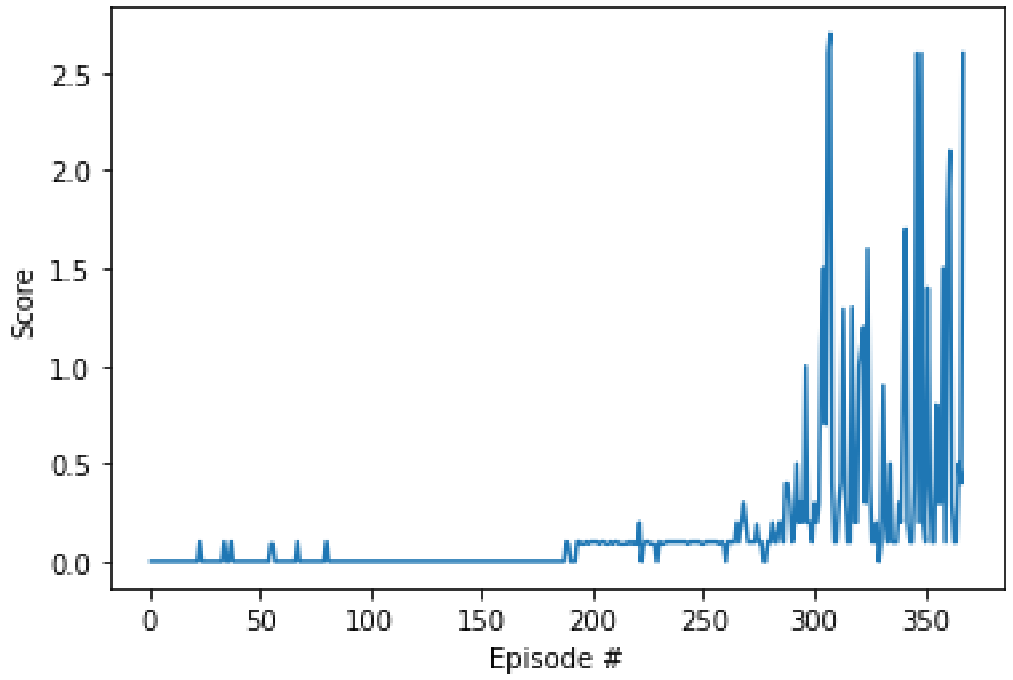
- BUFFER_SIZE = int(1e6) – Size of the replay buffer
- BATCH_SIZE = 128 – Number of experiences used in each training epoch
- GAMMA = 0.999 – Discount factor on rewards
- TAU = 1e-2 – Soft update interpolation parameter
- LR_ACTOR = 1e-4 – Learning rate for actor network
- LR_CRITIC = 1e-3 – Learning rate for critic network
- WEIGHT_DECAY = 0 – L2 weight decay on critic network
- eps_start = 1.0 – Starting value of epsilon
- eps_decay = 0.995 – Decay rate of epsilon (eps(t+1) = eps(t)*eps_decay)

The actor network has 3 layers (1 input, 1 output, and 1 hidden). The input layer is the state size, the output layer is the action size, and the hidden layer has 64 nodes. Each layer uses a tanh activation function.

The critic network has 4 layers (1 input, 1 output, and 2 hidden). The input layer is the state size + action size, the output layer has 1 node (the predicted value), and the hidden layers all have 64 nodes. Each layer uses a tanh activation function except the output layer which has no activation function.

Results

The agent took a total of 367 episodes to train. The score per episode can be seen in the plot below:



The final average reward over 100 episodes was 0.51.

Future Work

This problem may be able to be solved more effectively using:

- <u>Prioritized Experience Replay</u> – Assign each experience an importance level based on its TD error then sample from the replay buffer non-uniformly where more important experiences are used more often.
- <u>N step bootstrapping</u> – Instead of using pure TD estimation or Monte Carlo estimation for the Q function, use a mixture of the two where n true rewards are used to compute the value. This increases noise but reduces bias.
- <u>Modified Reward Function</u> – The conditions in which the agents get a reward is somewhat hard to randomly stumble upon. Providing a small reward for hitting the ball (even if it doesn't go over the net) might make it easier for the network to learn that skill.