Project 3: Multi-Agent RL with Unity Tennis Environment

Algorithm description

The DDPG algorithm was used to solve this environment, specifically the version of the environment designed to work with multiple agents. DDPG is an actor-critic method that entails an actor to produce optimal actions according to a policy, and a critic to predict the state-action value of following the policy. It derives from Deep Q-Network (DQN) and Deterministic Policy Gradient (DPG), enabling the jump to continuous action spaces [1]. Most of this code was provided in the Udacity nanodegree repository, and changes were made to the replay buffer and the noise model to accommodate multiple agents.

The actor consists of two fully connected layers, with 256 neurons each, an input size of the number of state features (33 in this case), and an output of the action size (4 in this case). The first layer uses a Relu activation function, and the second layer uses a tanh activation function, to bound the action outputs between -1 and 1, according to the environment specifications.

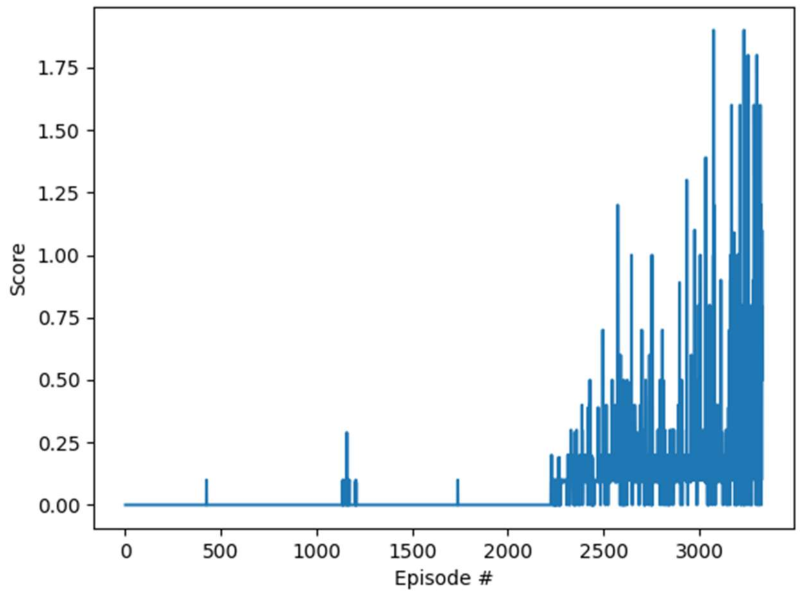
The critic consists of four fully connected layers, with the first two having 256 neurons, and the other two having 128 neurons each. The first layer uses a leaky Relu activation function, and the output is concatenated with the action to be passed to the second and third layers, which also use a leaky Relu activation function. The last layer has no activation function, as it simply maps the result of the above layers to a single value that represents the state-action value.

Below is the list of hyperparameters used for the training:

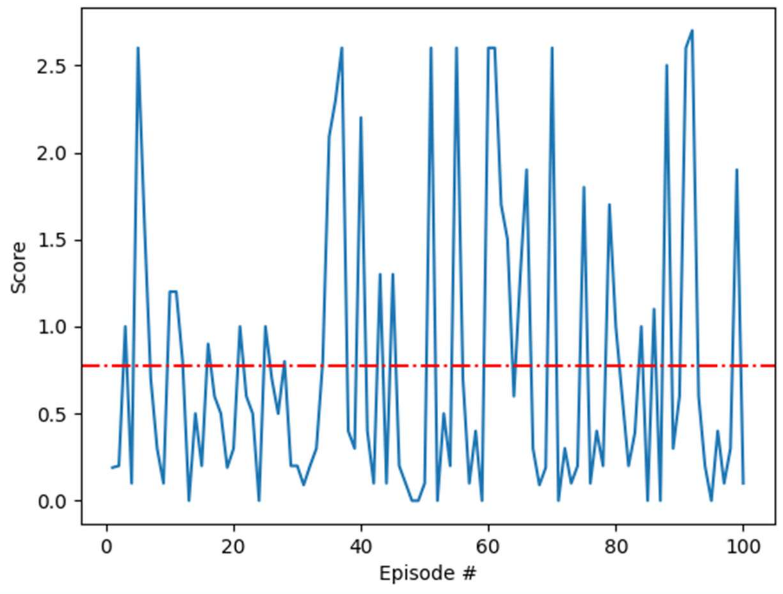
```
BUFFER_SIZE = int(1e5) # replay buffer size

BATCH_SIZE = 128       # minibatch size

GAMMA = 0.99           # discount factor

TAU = 1e-3             # for soft update of target parameters

LR_ACTOR = 1e-4        # learning rate of the actor

LR_CRITIC = 3e-4       # learning rate of the critic

WEIGHT_DECAY = 0.0     # L2 weight decay
```

Training and Testing Results

The environment was solved by the DDPG agent in 3328 episodes, obtaining an average score over the previous 100 episodes of +0.5. The agent does not see any substantial reward for many episodes, before it learned to keep the ball from hitting the ground. Once it began to learn, the competing agents quickly learned from one another's experiences.



The trained agent was run for an additional 100 episodes to prove its capability to generalize to new states. These results are included here.



Opportunities for Future Work

The training here was slow to gain momentum. I expect this is because the agent must first learn to hit the ball over the net before it can learn to hit the ball over the net in such a way that the opponent cannot return the ball. This compound learning task is complex, so longer training could be expected. However, it may be possible to reduce the time horizon to score improvement. Perhaps a more complex reward function, accounting for direction and initial velocity of the hit as an early indicator of the success of each hit would help make the learning more intuitive.

References

[1] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, Daan Wierstra, “Continuous control with deep reinforcement learning”, https://arxiv.org/abs/1509.02971