Project Report Collaboration and Competition

This report highlights the Actor and Critic networks with Deep Deterministic Policy Gradients learning agents[i] along with the hyperparameters and provides some references for future work.

Actor Network

The network is a feed-forward network with 3 fully connected ReLu activation layers as follows

1. Input 24 (see state size) => Output 400
2. Input 400 => Output 300
3. Input 300 => Output 4 (action size)

with tanh activation function that maps state -> action values.

Critic Network

The network is a feed-forward network with 4 fully connected leaky ReLu activation layers as follows

1. Input 24 (see state size) => Output 256
2. Input 256 => Output 256
3. Input 256 => Output 128
4. Input 128 => Output 1

Learning Agents

The learning agents are created as a class that interacts and learns from the environment with local/target actor and critic network as initialized variants plus Adam variant as optimizer.

Hyperparameters

- Replay buffer size => 1e5
- Minibatch size => 1024
- Discount factor Gamma => 0.9
- TAU for soft update of target parameters => 1e-2
- Learning rate (Actor) => 1e-3
- Learning rate (Critic) => 1e-3
- Weight decay => 1e-4
- eps_start => 1.0
- eps_min => 0.01
- eps_decay rate => 0.9975

Training

The training of the agents via ddpg function (see Tennis.ipynb) is running the following steps in every episode:

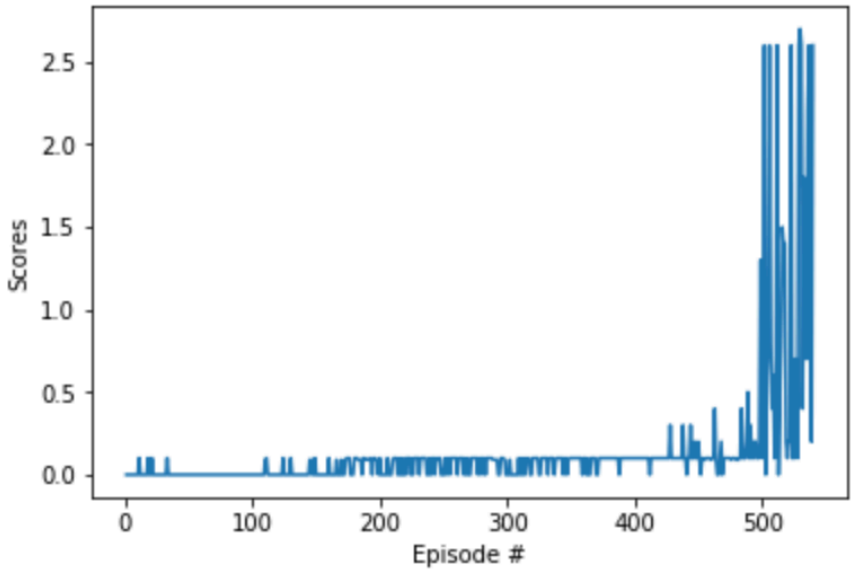
- Return actions for given state as per current policy from actor/critic local network
- Save experience in replay memory
- Learn every defined update time steps if enough samples are available in memory with random subset to update value parameters using given batch of experience tuples
    1. Get max predicted Q values (for next states) from actor/critic target network
    2. Compute Q targets for current states
    3. Get expected Q values from actor/critic local network
    4. Compute and minimize MSE loss
    5. Soft update target network

In total it is designed to run over 1,000 episodes (with 1,000 iterations per episode). The task is episodic, and in order to solve the environment, the agents must get an average score of +0.5 (over 100 consecutive episodes, after taking the maximum over both agents).

Plot of Rewards

```
Episode 100     Average Score: 0.004
Episode 200     Average Score: 0.031
Episode 300     Average Score: 0.062
Episode 400     Average Score: 0.075
Episode 500     Average Score: 0.130

Environment solved in 440 episodes!     Average Score: 0.523
```



Future Work

For training purposes, noise adding was paused since the learning process broke down quite early after only a few dozens of episodes – no matter how the different hyperparameters were tuned. In next stage the batch normalization should be added to the respective network architectures to make the learning more robust against added noise.

Additional enhancements can be achieved via implementing some benchmarking from Yan Duan, Xi Chen, Houthooft R., Schulman J., Abbeel P., Benchmarking Deep Reinforcement Learning for Continuous Control, arXiv:1509.02971.

---

[i] Implementations of the Deep Deterministic Policy Gradients adapted from https://github.com/udacity/deep-reinforcement-learning/tree/master/ddpg-pendulum and https://github.com/udacity/deep-reinforcement-learning/tree/master/ddpg-bipedal