Project 3 Playing Tennis

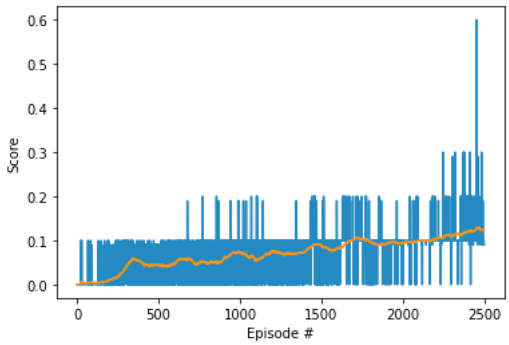
Multi-agent DDPG (4-1 in notebook)

Learning Algorithm:

- Deep Deterministic Policy Gradient (DDPG)
- Network:
  - Actor: 2 fully connected hidden layers with 256 and 128 units. Weights were uniformly and randomly initialized between -0.003 and 0.003.
  - Critic: 2 fully connected hidden layers with 256+action size and 128 units. Weights were uniformly and randomly initialized between -0.003 and 0.003.
- Training Hyperparameters:
  - Discount (γ) = 0.99
  - Soft update ratio (τ) = 0.001
  - Buffer size = 100000
  - Batch size = 128
  - Learning rate for actor network = 0.0001
  - Learning rate for critic network = 0.001
  - Number of agents: 2
  - Training start episode: 50
  - Number of training per step: 4
  - Max step in each episode: 1000

Scores:

- Never solved in 2500 training episodes.
- Scores (blue) and average of 100 episode(orange):



Multi-agent DDPG with noise decay (4-2 in notebook)

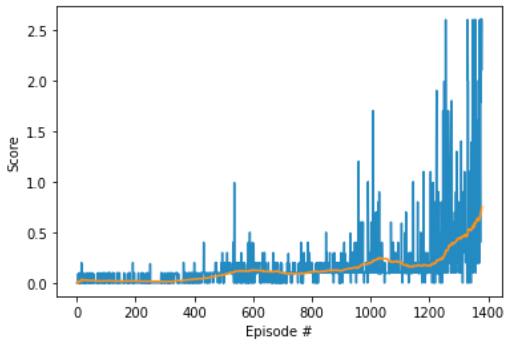
Learning Algorithm:

- Deep Deterministic Policy Gradient (DDPG)
- Network:
  - Actor: 2 fully connected hidden layers with 256 and 128 units. Weights were uniformly and randomly initialized between -0.003 and 0.003.
  - Critic: 2 fully connected hidden layers with 256+action size and 128 units. Weights were uniformly and randomly initialized between -0.003 and 0.003.

- Training Hyperparameters:
    - Discount (γ) = 0.99
    - Soft update ratio (τ) = 0.001
    - Buffer size = 100000
    - Batch size = 256
    - Learning rate for actor network = 0.0001
    - Learning rate for critic network = 0.001
    - Noise scale decay = 0.99
    - Minimum noise scale = 0.1
    - Noise distribution: Normal (Gaussian)
    - Number of agents: 2
    - Training start episode: 250
    - Number of training per step: 4
    - Max step in each episode: 1000

Scores:

- Solved in 1230 (1330-100) training episodes!
- Scores (blue) and average of 100 episode(orange):



Future work

The training of the with increased buffer size significantly increased the training time, so I didn't have enough time to try the Soccer game. In the future, I'd like to try using the same noise decay multi DDPG agent on the application.

Along the exploratory study, I came across a paper introducing ApeX which utilized distributed architecture to train DDPG agents with Centralized and Prioritized Experience Replay. It could a new direction to try solving the Tennis and Soccer game.