P3. Collaboration and Competition

Learning algorithm

Even though we are on a multi-agent environment, since it is symetric, we can use Deep Deterministic Policy Gradient (DDPG) to create a shared critic and actor for both agents and solve the environment. DDPG is an extension of DQN to continuous action spaces on an actor-critic manner. The main difference is that instead of just one network that outputs the value for every action, we have two: the actor, a maximizer that outputs the action with the biggest value, and the critic, that takes this action and a state and outputs the value.

To update the value network, the critic, we use an target actor and critic (fixed parameters) to create an Q used as target to train the learning critic.

Then, we use our updated critic to calculate a value with which we will do gradient ascent on the actor.

Also, an Ornstein-Unlenbeck process is implemented as noise generator to favor exploration.

Hyperparameters

```
BUFFER_SIZE = int(2e5)  # replay buffer size
BATCH_SIZE = 256        # minibatch size
GAMMA = 0.9             # discount factor
TAU = 1e-3              # for soft update of target parameters
LR_ACTOR  = 1e-3        # learning rate of the actor
LR_CRITIC = 1e-3        # learning rate of the critic
WEIGHT_DECAY = 0        # L2 weight decay
OU_MU = 0.2
OU_SIGMA = 0.2
OU_THETA = 0.15
```

noise_scaling_factor = 0.995

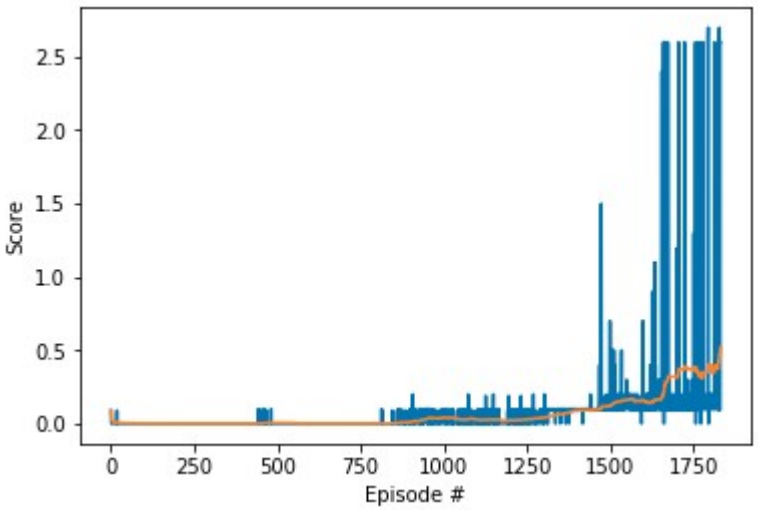
They hyperparameters have been chosen based on the previous project, with the principal difference of higher learning rates and more Ornstein-Unlenbeck noise, and the addition of a noise scaling factor to decrease the noise with each episode.

Model architectures

Both are neural networks with 2 hidden with 64 units, with batch normalization on the first hidden layer and ReLU as activation function. For the critic, the actions come in in the second hidden layer.

Plot of rewards

The environment is considered solved, when the average (over 100 episodes) of thescores is a least +0.5.




```
Episode 100     Average Score: 0.0018   Time:0.36 min.
Episode 200     Average Score: 0.0000   Time:0.75 min.
Episode 300     Average Score: 0.0000   Time:1.14 min.
Episode 400     Average Score: 0.0000   Time:1.54 min.
Episode 500     Average Score: 0.0059   Time:1.97 min.
Episode 600     Average Score: 0.0000   Time:2.36 min.
Episode 700     Average Score: 0.0000   Time:2.75 min.
Episode 800     Average Score: 0.0000   Time:3.15 min.
Episode 900     Average Score: 0.0113   Time:3.60 min.
Episode 1000    Average Score: 0.0405   Time:4.19 min.
Episode 1100    Average Score: 0.0275   Time:4.74 min.
Episode 1200    Average Score: 0.0241   Time:5.32 min.
Episode 1300    Average Score: 0.0393   Time:5.94 min.
Episode 1400    Average Score: 0.0893   Time:6.80 min.
Episode 1500    Average Score: 0.1242   Time:8.04 min.
Episode 1600    Average Score: 0.1559   Time:9.62 min.
Episode 1700    Average Score: 0.3143   Time:12.98 min.
Episode 1800    Average Score: 0.4045   Time:17.49 min.
Episode 1832    Average Score: 0.5198   Time:18.28 sec.
Environment solved in 1832 episodes!    Average Score: 0.52
CPU times: user 18min 27s, sys: 29.3 s, total: 18min 56s
Wall time: 20min 13s
```

Ideas for future work

1. Prioritized Experience Replay: at the beginning, actions are random and there are not many useful samples to start learning from. Giving priority to the samples with good trajectories should speed convergence.
2. Multi-Agent DDPG(MADDPG): similar to DDPG, but with an actor-critic for each agent and training the critic the all the observations and action of all agents.
3. N-step bootstrap: the rewards are very sparse, make us have few samples that our critic can learn a value from. If when increase the number of step, we increase the possibility of having a reward, making the training more stable.