Collaboration and Competition

Introduction

This report provides a brief introduction to project name 'Collaboration and Competition' offered as part of Udacity Nano degree on deep reinforcement learning. The project covers the concepts of multi-agent reinforcement learning in which multiple agents has to collaborate and compete to solve a task.

For the project, we are presented with a Tennis environment, consisting of two agents. The goal of the agents is to keep the ball in play (air). Both agents receive a **reward** of *+1* for hitting the ball over the net, the reward of *-1* is given in case ball hit the ground or agent hit the out of the bound ball. Each agent has its own 8 variable observation/**state space** consisting of position of ball, its velocity among others. Both agents can take 2 **actions of continuous nature**, corresponding to movement towards or away from net and jumping. Since the task is episodic, the environment is considered solved if agents get an average score of +0.5 over 100 consecutive episodes.

Deep deterministic policy gradient (DDPG) that has shown to work well for the continuous action spaces with single agent. Therefore, I employed multi-agent variant of DDPG to solve the environment. In the following, a brief overview of Multi-Agent DDPG (MADDPG) algorithm, the agents training performance and few pointers for future work is provided.

Multi-Agent Deep Deterministic Policy Gradient (MADDPG)

Traditional reinforcement approaches such as Q learning and policy gradient methods are poorly suited for multi-agent environment due to high variance in training and non-stationary environment. Lowe et al. [1] proposed a general purpose multi-agent learning algorithm that learn policies using local observation and support collaborative and competitive interactions among multiple agents. The core idea of MADDPG is in adopting the *centralized training with decentralized execution*. Centralized training mean, even though each agent has access to its own local observations, all agents are guided by a centralized critic module. This reduces the non-stationarity of the environment. The decentralized execution emphasizes the fact that the centralized critic is removed during testing, which mean each individual agent has learnt his own local policy to collaborate and solve the environment.

MADDPG also utilize the actor-critic, fixed target and shared replay buffer framework. For each maintain three components:

- An actor network that uses local observations to decide deterministic actions
- A target network, similar to actor network, for training stability
- A critic network that advice actor to reinforce certain actions since it has access to observations of all agents

Following is the MADDPG algorithm:

Algorithm 1: Multi-Agent Deep Deterministic Policy Gradient for N agents

for episode $= 1$ to M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial state $\mathbf{x}$
  for $t = 1$ to max-episode-length **do**
    for each agent i, select action $a_i = \boldsymbol{\mu}_{\theta_i}(o_i) + \mathcal{N}_t$ w.r.t. the current policy and exploration
    Execute actions $a = (a_1, \ldots, a_N)$ and observe reward r and new state $\mathbf{x}'$
    Store $(\mathbf{x}, a, r, \mathbf{x}')$ in replay buffer $\mathcal{D}$
    $\mathbf{x} \leftarrow \mathbf{x}'$
    for agent $i = 1$ to N **do**
      Sample a random minibatch of S samples $(\mathbf{x}^j, a^j, r^j, \mathbf{x}'^j)$ from $\mathcal{D}$
      Set $y^j = r_i^j + \gamma Q_i^{\boldsymbol{\mu}'}(\mathbf{x}'^j, a'_1, \ldots, a'_N)|_{a'_k = \boldsymbol{\mu}'_k(o_k^j)}$
      Update critic by minimizing the loss $\mathcal{L}(\theta_i) = \frac{1}{S}\sum_j \left(y^j - Q_i^{\boldsymbol{\mu}}(\mathbf{x}^j, a_1^j, \ldots, a_N^j)\right)^2$
      Update actor using the sampled policy gradient:

$$\nabla_{\theta_i} J \approx \frac{1}{S}\sum_j \nabla_{\theta_i}\boldsymbol{\mu}_i(o_i^j)\nabla_{a_i} Q_i^{\boldsymbol{\mu}}(\mathbf{x}^j, a_1^j, \ldots, a_i, \ldots, a_N^j)\big|_{a_i = \boldsymbol{\mu}_i(o_i^j)}$$

    end for
    Update target network parameters for each agent i:

$$\theta_i' \leftarrow \tau\theta_i + (1-\tau)\theta_i'$$

  end for
end for

Figure 1: MADDPG Training Algorithm [1]

Agent in Action:

For the Tennis environment in multi-agent setting, the agent are trained using MADDPG algorithm. I tuned several hyper-parameters namely batch size, weight decay, fully connected units, and sigma value in OUNoise. The agent was able to solve the environment in 1400 episodes. The environment is considered solved, when the agent is able to get average of 0.5+ score for 100 consecutive episodes. Figure 2 provide the training progress plot of agents' max score.

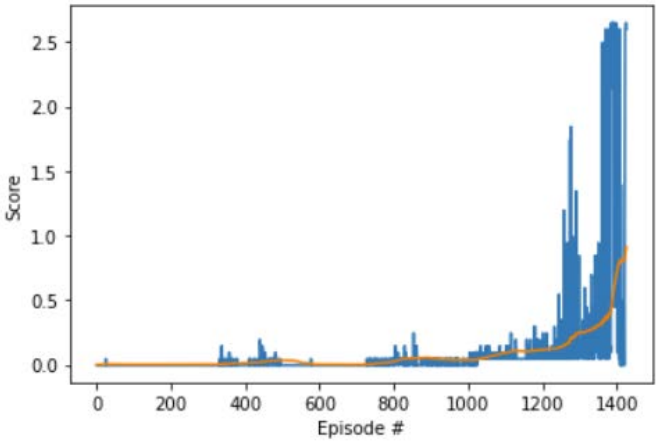

Figure 2: Average rewards using DDPG during Agent's Training

Future work

For the future work, I plan to apply other policy-based methods to compare the performance of with MADDPG.

References

[1]. Lowe, R., Wu, Y. I., Tamar, A., Harb, J., Abbeel, O. P., & Mordatch, I. (2017). Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in neural information processing systems* (pp. 6379-6390).