Nanodegree: Deep Reinforcement Learning

Project: Collaboration and Competition (By: Kranthi Kumar Talluri)

Section 1: Introduction

The project demonstrates the efficiency of Multi-Agent Actor-Critic Method, particularly the implementation of Multi-Agent Deep Deterministic Policy Gradient (MADDPG), in learning optimal policy for Reinforcement Learning tasks. The task utilizes a Tennis environment with 2 agents responsible for controlling rackets. The agent's objective is to collectively work together to sustain the ball in the play for an extended time period. The agents optimize the cumulative rewards by associating states with actions through a policy. Optimal policy is achieved by using Deep Neural Network (DNN) as non-linear function approximator. Throughout the project, various files such as Jupyter Notebooks and code files are used to set up dependencies, define agent characteristics and store trained models.

A tennis game is implemented here based on DNN to demonstrate its model free learning. The agents in tennis environment must collaboratively maintain the ball in play to fetch a reward of +0.1. The observation space is 8 dimensions.

- The goal is to train an agent to achieve an average score of +0.5 over 100 consecutive episodes, after taking max of both agents, by maintaining the ball in play for as many time steps as possible.
- The state space consists of 8 features, including the position and velocity of ball and racket.
- The two continuous action spaces are involved, one for each agent, for the movement towards or away from the net and jumping.
- MADDPG, an off-policy learning algorithm, is used as the core learning algorithm. It learns the action-value function (q) through Multi-Agent Actor-Critic network and uses a DNN as a function approximator.
- Code cell is provided with a random action agent, to learn how agent is controlled by Python API to receive feedback from environment.

Section 2: Implementation

Subsection 1: Algorithm

The Deep Deterministic Policy Gradient (DDPG) algorithm is a RL technique which is a combination of both Policy gradients and Q-learning. It has two models namely the actor and the critic, hence this algorithm is also known as an actor-critic technique. The actor gives suggestions to perform action on the bases of current state whereas Critic helps to validate about how good is the taken action.

- Multi-Agent Deep Deterministic Policy Gradient (MADDPG) an extension of DDPG algorithm is most widely used in multi-agent scenarios.
- In MADDPG, the agents are bound to consider their own actions along with the actions of other agents, because the action of one agent could have an impact on the action of another agent as they share the same environment.

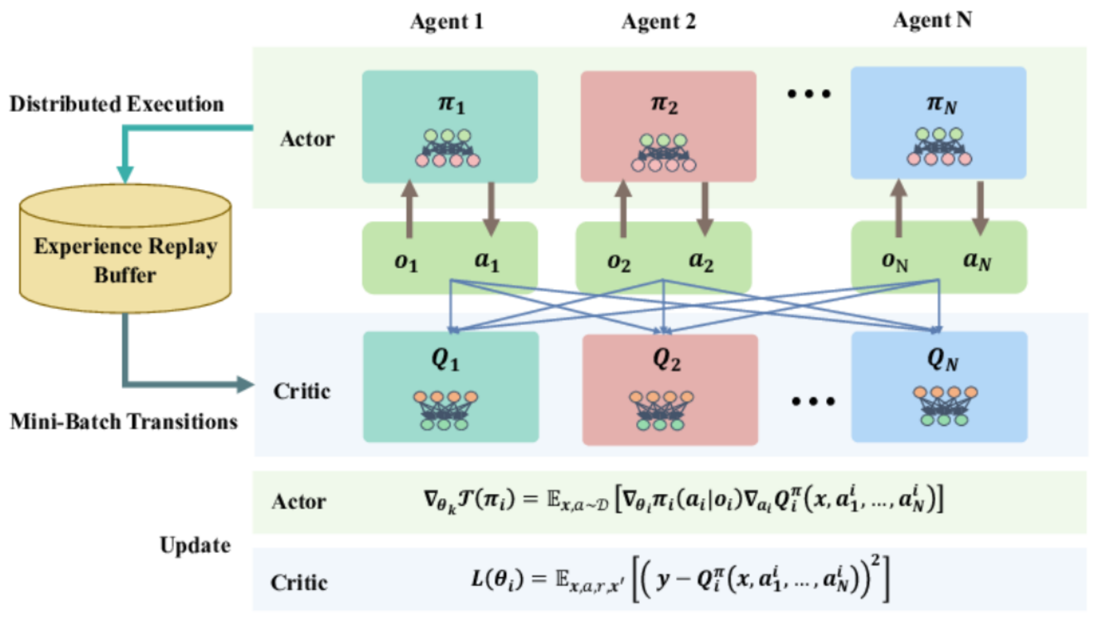

Agent 1
Agent 2
Agent N
Distributed Execution
Actor
π_1
π_2
π_N
Experience Replay Buffer
o_1 a_1
o_2 a_2
o_N a_N
Critic
Q_1
Q_2
Q_N
Mini-Batch Transitions
Update
Actor $\nabla_{\theta_k}\mathcal{J}(\pi_i) = \mathbb{E}_{x,a\sim\mathcal{D}}\left[\nabla_{\theta_i}\pi_i(a_i|o_i)\nabla_{a_i}Q_i^{\pi}(x, a_1^i, \ldots, a_N^i)\right]$
Critic $L(\theta_i) = \mathbb{E}_{x,a,r,x'}\left[\left(y - Q_i^{\pi}(x, a_1^i, \ldots, a_N^i)\right)^2\right]$

Fig: Workflow architecture for MADDPG algorithm[source]

- MADDPG are mostly used in model scenarios where cooperation or competition is needed by the agent to complete the task. In cooperative scenarios agents work together to achieve a common goal, whereas the agents try to outperform each other in competitive scenarios.
- For policy approximation of each agent during training, MADDPG utilizes deep neural networks. The policies guide the agents in taking actions in various situations, which leads the agent to maximize the reward.
- In MADDPG, the Centralized or decentralized are the two types of learning processes. Here Centralized learning means the learning of an agent's policy depends on the actions of other agent whereas policies are independent in decentralized learning.

Subsection 2: Architecture of Neural Network

The actor network consists of 3 hidden layers, with 100, 75, and 75 nodes respectively. It takes game states as input and gives best action as output. The inputs are initially normalized with one batch normalization layer and then comes the hidden layer followed by an activation function named Rectified Linear Unit (ReLU) and for handling the continuous action space Tanh activation function is used at output layer.

Critic network consists of 2 hidden layers with 100 and 100 nodes respectively. Here also the inputs are initially normalized with only one batch normalization layer and then comes the hidden layer followed by ReLU as an activation function. The state and action from actor network are passed as input and Q-value score to evaluate how good is the action obtained at output.

Subsection 3: Hyperparameters

Hyperparameters tuning plays a vital role in model performance. Among various important hyperparameters, the Learning rate (LR) plays a crucial role in achieving better performance for both Actor and Critic network. In can be seen from below figure that, Initially the performance of agent with MADDPG algorithm is poor. Changing the learning rate parameter helped the agent to learn and show some better progress.

With LR 1e-4, I was able to achieve decent exponential progress in learning with an average score of **0.5011** in 1364 episodes as shown in the below table:

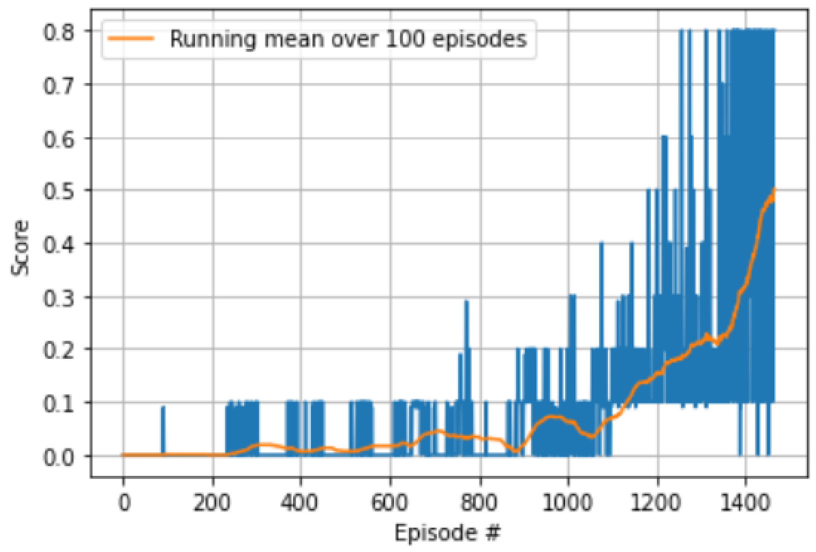
Number of Episodes	**Average score**
100	0.0009
400	0.0069
800	0.0335
1200	0.1542
1400	0.3188
1464	0.5011

Other hyperparameters are shown in the below table.

Hyperparameters used	**Assigned values**
Learning rate Actor	1e-4
Learning rate Critic	1e-4
Relay buffer size	1e5
Batch size	128
Gamma (discount factor)	0.99
TAU	2e-3
Number of episodes	2000

Section 3: Results

Below graph is plotted between the number of episodes on X-axis and score on Y-axis. After tweaking some parameters, the below results are obtained. The maximum number of episodes for solving is 1364 episodes.



Conclusion and Ideas for Future Improvement

Multi-Agent RL is a flexible approach and when combined with policy-based methods, the continuous space problems can be solved efficiently as it approximates the real work with efficient agents collaboratively solving the complex problems. The current problem solved here consists of two agents with racket and the tennis environment. As a future work MADDPG can be used in optimizing the real time traffic flow scenarios where autonomous vehicles can be treated as multiple agents trying to interact with each other and the environment to collaboratively solve problems like traffic congestion, accident prevention while lane changing, merging etc.