DRLND - Collaboration and Competition – Project 3

Introduction

The objective of this project is to train artificial agent in a multi agent environment settings, to be able to collaborate to maximize the collective reward as well as compete to maximize the agents rewards. This project specifically deals with Tennis environment provided where two agents compete and collaborate.

Solution

In order to solve this multiagent scenario, I would choose the Multi-Agent Deep Deterministic Policy Gradient (MADDPG). MADDPG is an extension of DDPG algorithm where each agent would still run as a DDPG agent. Each of the DDPG agent will have its Actor and Critic networks. However, the replay buffer will be shared between them making available the entire set of actions and states to both agents here.

This is as per the implementation mentioned in MADDPG paper, where the execution is decentralized and training is centralized. Here each agent receives its own local observation at execution time. The ability to act in mixed cooperative-competitive environments is critical for intelligent agents; while competitive training provides a natural curriculum for learning, agents must also exhibit cooperative behavior at execution time. This is exactly what happens here.

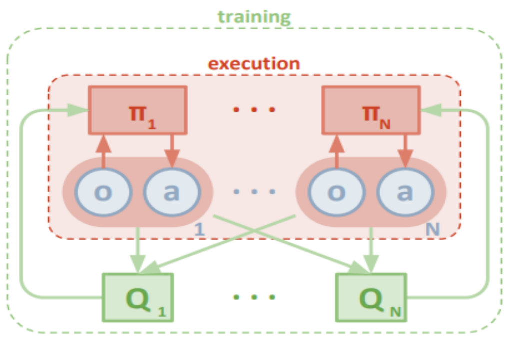


Figure – Multiagent decentralized actor, centralized critic

As stated before, this setting is extension of actor-critic policy gradient methods where the critic is augmented with extra information about the policies of other agents, while the actor only has access to local information. In execution time, only the local actors are used, acting in a decentralized manner and equally applicable in cooperative and competitive settings.

One of the main problem in extending Q-learning to multi agent scenario is non stationarity, as each agent would independently update policies as learning progresses. In MADDPG as we know the actions taken by all agents, the environment is stationary even as the policies change. This is possible as Experience replay buffer contains the tuples having experience of all agents and is shared across agents

The solution MADDPG agent would have two separate DDPG agents. Each actor would take a state as input which has 24 dimensions. Each critic would have concatenation of states(2*24 = 48) and actions (2+2) for both agents

Network Model

Actor Network: This network has an input layer which takes the states which are then passed through batch normalization layer to next layer. Two hidden layers with size 400 and 300 were tried for this experiment. The output layer was a fully connected layer with output size set as action size representing action probability. Leaky Relu activation layer was used after first and second layer. Tanh activation was used on the last layer.

Critic network: Network used had an input layer which takes the states which are then passed through batch normalization layer to next layer. Two hidden layers with size 400 and 300 were used here as well. Actions were added in the first hidden layer. This proved effective. The output layer was a fully connected layer with output size of 1. Leaky Relu activation function was used after first and second layer.

In DDPG, there are two copies of network weights or in other words, a regular network for the actor and one for critic, a target network for the actor and one for critic. As an update is issued to the MADDPG agent, it would sample records from replay buffer. Then it would further figure out the actual underlying ddpg agent to be updated.

Hyper parameters

After a working code was setup, the hyper parameters were tuned to arrive at these values

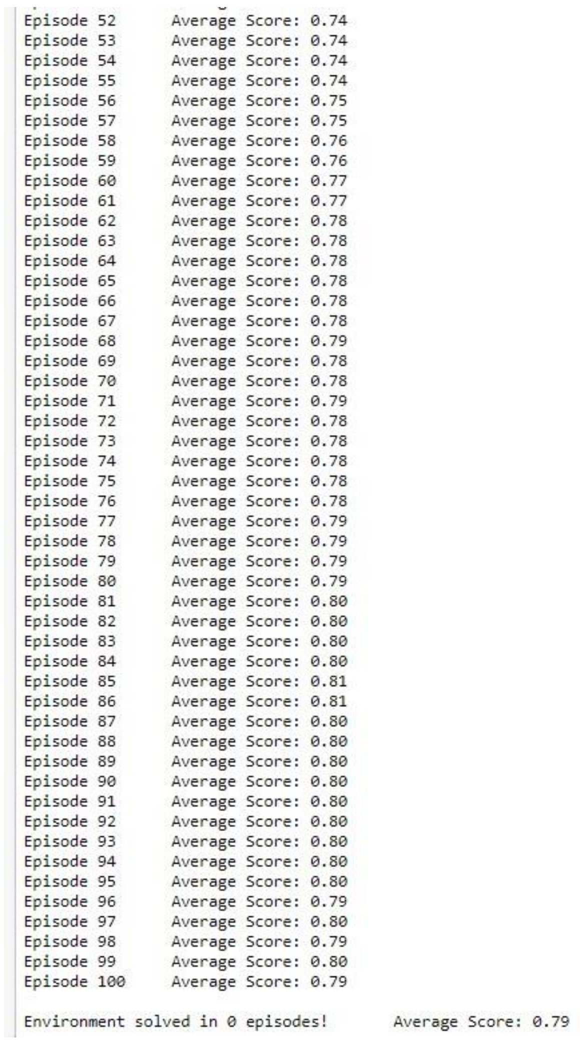
discount_factor = 0.95
tau = 1e-3
batch_size = 500
replay_buffer_size= 50000
critic input = States across agents + actions = 24+24+2+2=52
lr_actor=1.0e-4
lr_critic=1.0e-3
max number of episodes = 1000,
episode_length = 500
noise_reduction = 0.9999
goal_score = 0.5
weight_decay = 1.e-5

Huber loss was used and it seemed to be better as its less sensitive to outliers in data.

Results

With the above configuration and hyperparameter, the environment could be solved in 100 episodes. i.e that average across the first set of 100 episode returned 0.5 rewards. The average obtain was 0.79 over first 100 episodes.

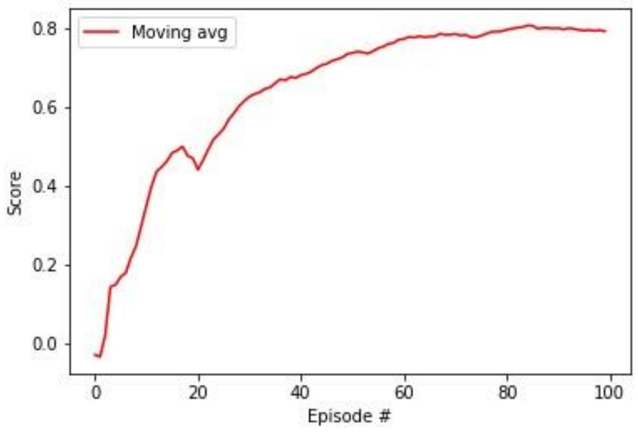
The checkpoint were captured every 10 episodes and when the goal was achieved.



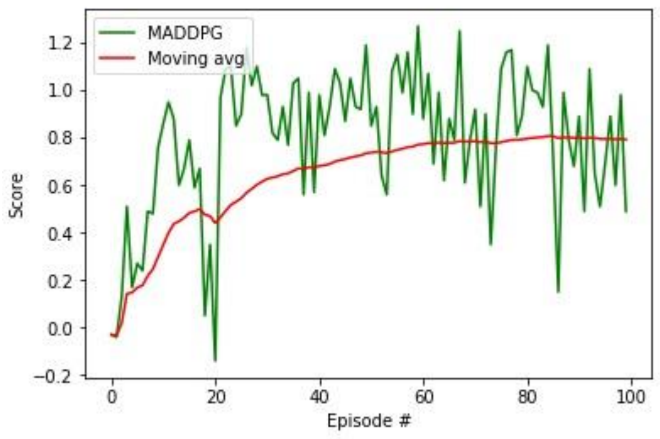
```
Episode 52      Average Score: 0.74
Episode 53      Average Score: 0.74
Episode 54      Average Score: 0.74
Episode 55      Average Score: 0.74
Episode 56      Average Score: 0.75
Episode 57      Average Score: 0.75
Episode 58      Average Score: 0.76
Episode 59      Average Score: 0.76
Episode 60      Average Score: 0.77
Episode 61      Average Score: 0.77
Episode 62      Average Score: 0.78
Episode 63      Average Score: 0.78
Episode 64      Average Score: 0.78
Episode 65      Average Score: 0.78
Episode 66      Average Score: 0.78
Episode 67      Average Score: 0.78
Episode 68      Average Score: 0.79
Episode 69      Average Score: 0.78
Episode 70      Average Score: 0.78
Episode 71      Average Score: 0.79
Episode 72      Average Score: 0.78
Episode 73      Average Score: 0.78
Episode 74      Average Score: 0.78
Episode 75      Average Score: 0.78
Episode 76      Average Score: 0.78
Episode 77      Average Score: 0.79
Episode 78      Average Score: 0.79
Episode 79      Average Score: 0.79
Episode 80      Average Score: 0.79
Episode 81      Average Score: 0.80
Episode 82      Average Score: 0.80
Episode 83      Average Score: 0.80
Episode 84      Average Score: 0.80
Episode 85      Average Score: 0.81
Episode 86      Average Score: 0.81
Episode 87      Average Score: 0.80
Episode 88      Average Score: 0.80
Episode 89      Average Score: 0.80
Episode 90      Average Score: 0.80
Episode 91      Average Score: 0.80
Episode 92      Average Score: 0.80
Episode 93      Average Score: 0.80
Episode 94      Average Score: 0.80
Episode 95      Average Score: 0.80
Episode 96      Average Score: 0.79
Episode 97      Average Score: 0.80
Episode 98      Average Score: 0.79
Episode 99      Average Score: 0.80
Episode 100     Average Score: 0.79

Environment solved in 0 episodes!       Average Score: 0.79
```

Moving average



Episode score with Moving average vs episodes



Future Enhancements

The MADDPG in itself was challenging to arrive a good hyper parameters. Further improvements to this could include

- Trying out a delayed training where a fixed number of initial episode would be set just for random exploration and fill the replay buffer with experience.
- Further hyperparameter tuning. The episode timesteps seem to have an impact on the training duration. With a better hardware and powerful compute could try experiment to achieve better results.
- Trying out environment with more than 2 agents could yield more challenge in finetuning the hyperparameter for MADDPG.

References

https://papers.nips.cc/paper/7217-multi-agent-actor-critic-for-mixed-cooperative-competitive-environments.pdf