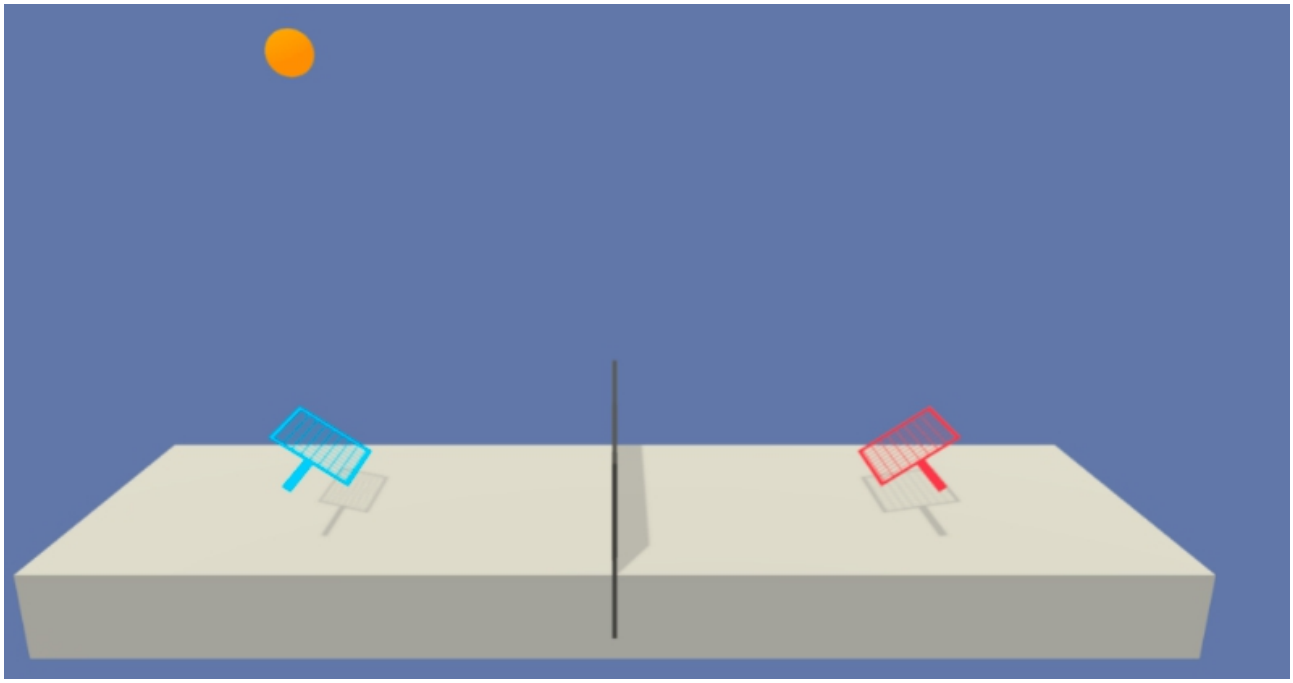
Deep Reinforcement Learning Nano Degree
Project 3 Collaboration and Competition
Report by Anas Al-Nuaimi
31. August 2021

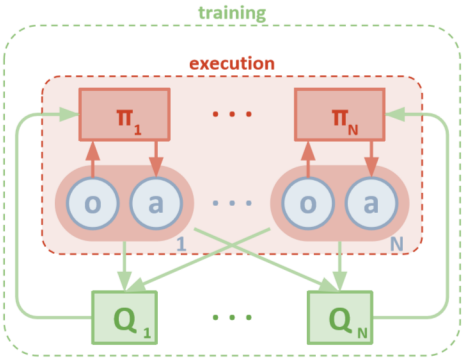
Challenge:
The challenge involves train competing (also collaborating) RL agents to play a tennis game.

A screenshot from the game is shown below. Each racket, controlled by a separate agent, can be moved in two directions (left-right, up-down). The goal of each agent should be to bounce back a ball to the opponent's area of the table. In this case a reward of 0.1 is returned. Otherwise the reward is 0. The overall of the game is to collect the highest reward which implicitly means the competition is also a collaboration.



Implemented Solution Algorithm:
For solving the tennis competition challenge we use the Multi Agent Deep Discrete Policy Gradient (DDPG) originally published by OpenAI in their landmark paper https://papers.nips.cc/paper/2017/file/68a9750337a418a86fe06c1991a1d64c-Paper.pdf.

As the name implies, MADDPG is a multi-agent extension of DDPG, which has been explained in continuous_navigation_project/README.md of this repository. In particular in introduces a per agent actor as well as per agent critic. However, in order to achieve effective learning and counter act the negative effect of environment non-stationarity it couples the agents in the critic. As such, it allows each agent to have a partial observation of the environment while requiring the critic to have access to all of the agents' observations. This is furthermore depicted in the figure below that is found in the paper.



The result is that during training every agent's actor needs to forward its observations and action output to a all critics allowing it to more effectively estimate the Q function. However, at execution time each actor has only access to its partial observation of the state. This special structure allows us to categorize MADDPG among the class of multi-agent algorithms performing *central planning and decentralized execution.*

This special construction that does not require actors to have access to the observations of other agents furthermore means that MADDPG can be used not just competition but also collaboration as well as mixed games.

Since actors have to have access to all observations and predicted actions the replay buffer is different from the one used in the ../continuous_navigation_project. In particular each entry has the following elements:

(a[1],..,a[N],o[1],..,o[N],r[1],..r[N],b[1],..b[N])

a[i] is the action generated by actor i,
o[i] is agent i's observation of the state,
r[i] is agent i's reward,
b[i] is agent i's next state

This way, each time a sample is drawn the environment is stationary for that moment in that all states and actions that happened at that time are captured.

Finally, it can be mentioned that it is possible to have one central critic instead of one critic per agent.

The maddpg agent is implemented in maddpg.py. We implement a one critic per agent approach requiring us to have 8 networks

- a local and target actor per agent
- a local and target critic per agent

Otherwise, the reader will notice that the remainder of the code is quite similar to that in continuous_navigation_project/ddpg.py

In order to facilitate exploration we add a regular Gaussian random noise which we found to be more

Parameters

num_trainings_per_update	Number of forward backward passes each time an update is done	2+int(episode/500) i.e. 3 for episode 600

time_steps_before_training	Number of time steps skipped before an update of the model is done	5
batch_size	Batch size	512
lr_actor	Learning rate actor	1.00E-003
lr_critic	Learning rate critic	1.00E-003
clip_grad_norm	Clip gradient norm	TRUE
weight_decay	Weight decay	1.00E-004
start_noise_varinace	The noise variance at episode 1	1.00E+000

Analysis
It was hard to set the different parameters properly. Perhaps one of the most central parameters is the number of maximal time steps. Hence, we collected a statistic showing the maximal time step reached per episode. Observing this value not also helped set the value properly but also check if training is progressing as with increasing episode count we should have more and more time steps passing. This value is seen in the output of the program.

The noise process is also another crucial factor. Analyzing the training process – by observing the highest time step reached per episode, and watching the game visually as training commenced - we noticed that initially it takes a lot of tries to get some balls across. As such we determined it is better to use a Gaussian random process for exploration as opposed to the Ornstein-Uhlenbeck process which fosters consistency. This way we get a more erratic behavior initially which is good for collecting successful experiences.

The visual inspection allowed us also to realize that the ball at the beginning of the game was always falling from the same place. Hence, we tried setting the weights and biases of the actors to zero to ensure the ball always initially falls on the racket. While this worked in collecting a first set of experiences, it didn't help in the long run as as soon training started the network diverged away. So we resorted back to uniform normalization.

Another value we looked at was the norm of the action vector produced by the actors. We quickly noticed that the norm was converging to 1.41 showing that we were suffering from the exploding gradient problem. Hence, we added gradient clipping and weight decay.

Ideas for the Future
Due to time considerations we could not thoroughly, or only partially tested the following ideas and hence suggest them as future work:

- One critic only: While it is clear why we need one actor per agent (each agent sees the environment from a different perspective) it is conceivable to have one critic only.

- Ensemble of critics: since we already have multiple critics and in principle they should show the same behaviour it may be beneficial to consider them as an ensemble of learners to compute a more reliable Q value for action predictions. This would necitate investigating how to best compute the ensemble estimate of the Q value (simple average, weighted average, etc.) as well as add another hyper param for blending the critics with one another over time to ensure they don't diverge.
- Comparison to DDPG: I have seen that some DRLND course participants have solved the challenge by using the regular DDPG and having a separate ddpg agent instantiated per agent. As such, the only commong thing is the replay buffer in which both agents feed in their values. It is worthwhile comparing the effectiveness of MADDPG compared to multiple DDPG agents.
- Switching to Ornstein -Uhlenbeck) process: Having a Gaussian process initially seemed to help in collecting good experiences. However, over time as the agent(s) become better trained, it may be more beneficial to switch to an Ornstein-Uhlenbeck random process to foster consistency over time (correlated noise samples)

Output from running main.py

Below is the output by running main.py in the git commit
abe2a2ad3b2864baa18212cd920e2681039b10f6
The produced figure is also found below.
The trained agent models are found under the files
checkpoint_*-challenge_solved-git_abe2a2ad3b2864baa18212cd920e2681039b10f6.pth
where * represents a wild card

```
----------
Environment successfully started
INFO:unityagents:
'Academy' started successfully!
Unity Academy name: Academy
        Number of Brains: 1
        Number of External Brains : 1
        Lesson number : 0
        Reset Parameters :

Unity brain name: TennisBrain
        Number of Visual Observations (per agent): 0
        Vector Observation space type: continuous
        Vector Observation space size (per agent): 8
        Number of stacked Vector Observation: 3
        Vector Action space type: continuous
        Vector Action space size (per agent): 2
        Vector Action descriptions: ,
Number of agents: 2
Size of each action: 2
There are 2 agents. Each observes a state with length: 24
The state for the first agent looks like: [ 0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.         -6.65278625 -1.5
 -0.          0.          6.83172083  6.         -0.          0.        ]
/home/q409893/work/tools/miniconda3/envs/udacity/lib/python3.6/site-
packages/torch/nn/functional.py:995: UserWarning: nn.functional.tanh is deprecated. Use torch.tanh
```

```
instead.
  warnings.warn("nn.functional.tanh is deprecated. Use torch.tanh instead.")
Episode 20	Average max agent Score: 0.02. Average duration 0.0s. Avg timestep reached 19.35
Episode 27	Max (over agents) episode score: 0.00. Duration: 0.0s/home/q409893/work/repos/udacity/collaboration_and_competition_project/ReplayBuffer.py:46: FutureWarning: The pandas.np module is deprecated and will be removed from pandas in a future version. Import numpy directly instead
  self.device) for i in range(self.num_collaborating_agents)]
/home/q409893/work/repos/udacity/collaboration_and_competition_project/ReplayBuffer.py:49: FutureWarning: The pandas.np module is deprecated and will be removed from pandas in a future version. Import numpy directly instead
  self.device) for i in range(self.num_collaborating_agents)]
/home/q409893/work/repos/udacity/collaboration_and_competition_project/ReplayBuffer.py:52: FutureWarning: The pandas.np module is deprecated and will be removed from pandas in a future version. Import numpy directly instead
  self.device) for i in range(self.num_collaborating_agents)]
/home/q409893/work/repos/udacity/collaboration_and_competition_project/ReplayBuffer.py:55: FutureWarning: The pandas.np module is deprecated and will be removed from pandas in a future version. Import numpy directly instead
  self.device) for i in range(self.num_collaborating_agents)]
/home/q409893/work/repos/udacity/collaboration_and_competition_project/ReplayBuffer.py:58: FutureWarning: The pandas.np module is deprecated and will be removed from pandas in a future version. Import numpy directly instead
  self.device) for i in range(self.num_collaborating_agents)]
Episode 40	Average max agent Score: 0.00. Average duration 0.1s. Avg timestep reached 14.35
Episode 60	Average max agent Score: 0.00. Average duration 0.1s. Avg timestep reached 13.2
Episode 80	Average max agent Score: 0.00. Average duration 0.1s. Avg timestep reached 13.2
Noise variance: 0.6057704364907279
Episode 100	Average max agent Score: 0.00. Average duration 0.2s. Avg timestep reached 13.2
Episode 120	Average max agent Score: 0.00. Average duration 0.1s. Avg timestep reached 13.2
Episode 140	Average max agent Score: 0.00. Average duration 0.1s. Avg timestep reached 13.2
Episode 160	Average max agent Score: 0.00. Average duration 0.1s. Avg timestep reached 13.2
Episode 180	Average max agent Score: 0.00. Average duration 0.1s. Avg timestep reached 13.2
Noise variance: 0.36695782172616703
Episode 200	Average max agent Score: 0.00. Average duration 0.2s. Avg timestep reached 13.2
Episode 220	Average max agent Score: 0.01. Average duration 0.1s. Avg timestep reached 14.75
Episode 240	Average max agent Score: 0.00. Average duration 0.1s. Avg timestep reached 13.2
Episode 260	Average max agent Score: 0.00. Average duration 0.1s. Avg timestep reached 13.2
Episode 280	Average max agent Score: 0.00. Average duration 0.1s. Avg timestep reached 13.2
Noise variance: 0.22229219984074694
Episode 300	Average max agent Score: 0.00. Average duration 0.2s. Avg timestep reached 13.2
Episode 320	Average max agent Score: 0.01. Average duration 0.1s. Avg timestep reached 14.7
Episode 340	Average max agent Score: 0.00. Average duration 0.1s. Avg timestep reached 13.45
Episode 360	Average max agent Score: 0.00. Average duration 0.1s. Avg timestep reached 13.2
Episode 380	Average max agent Score: 0.00. Average duration 0.1s. Avg timestep reached 13.2
Noise variance: 0.1346580429260134
Episode 400	Average max agent Score: 0.00. Average duration 0.2s. Avg timestep reached 13.2
Episode 420	Average max agent Score: 0.00. Average duration 0.1s. Avg timestep reached 13.2
Episode 440	Average max agent Score: 0.00. Average duration 0.1s. Avg timestep reached 13.2
Episode 460	Average max agent Score: 0.00. Average duration 0.1s. Avg timestep reached 13.4
Episode 480	Average max agent Score: 0.01. Average duration 0.1s. Avg timestep reached 15.2
Noise variance: 0.08157186144027832
```

Episode 500 Average max agent Score: 0.01. Average duration 0.2s. Avg timestep reached 15.45
Episode 520 Average max agent Score: 0.00. Average duration 0.2s. Avg timestep reached 14.0
Episode 540 Average max agent Score: 0.00. Average duration 0.2s. Avg timestep reached 13.2
Episode 560 Average max agent Score: 0.00. Average duration 0.2s. Avg timestep reached 13.15
Episode 580 Average max agent Score: 0.02. Average duration 0.3s. Avg timestep reached 17.4
Noise variance: 0.04941382211003858
Episode 600 Average max agent Score: 0.00. Average duration 0.3s. Avg timestep reached 14.05
Episode 620 Average max agent Score: 0.00. Average duration 0.2s. Avg timestep reached 13.15
Episode 640 Average max agent Score: 0.00. Average duration 0.2s. Avg timestep reached 13.2
Episode 660 Average max agent Score: 0.00. Average duration 0.2s. Avg timestep reached 13.2
Episode 680 Average max agent Score: 0.00. Average duration 0.2s. Avg timestep reached 13.2
Noise variance: 0.04
Episode 700 Average max agent Score: 0.00. Average duration 0.3s. Avg timestep reached 13.2
Episode 720 Average max agent Score: 0.00. Average duration 0.2s. Avg timestep reached 13.15
Episode 740 Average max agent Score: 0.05. Average duration 0.3s. Avg timestep reached 20.6
Episode 760 Average max agent Score: 0.04. Average duration 0.3s. Avg timestep reached 19.65
Episode 780 Average max agent Score: 0.01. Average duration 0.3s. Avg timestep reached 15.35
Noise variance: 0.04
Episode 800 Average max agent Score: 0.01. Average duration 0.3s. Avg timestep reached 15.05
Episode 820 Average max agent Score: 0.03. Average duration 0.3s. Avg timestep reached 20.4
Episode 840 Average max agent Score: 0.08. Average duration 0.4s. Avg timestep reached 29.6
Episode 860 Average max agent Score: 0.05. Average duration 0.4s. Avg timestep reached 28.6
Episode 880 Average max agent Score: 0.05. Average duration 0.4s. Avg timestep reached 28.25
Noise variance: 0.04
Episode 900 Average max agent Score: 0.05. Average duration 0.5s. Avg timestep reached 28.25
Episode 920 Average max agent Score: 0.09. Average duration 0.5s. Avg timestep reached 37.5
Episode 940 Average max agent Score: 0.04. Average duration 0.3s. Avg timestep reached 22.7
Episode 960 Average max agent Score: 0.03. Average duration 0.3s. Avg timestep reached 18.25
Episode 980 Average max agent Score: 0.03. Average duration 0.3s. Avg timestep reached 18.7
Noise variance: 0.04
Episode 1000 Average max agent Score: 0.03. Average duration 0.5s. Avg timestep reached 26.15
Episode 1020 Average max agent Score: 0.03. Average duration 0.4s. Avg timestep reached 18.15
Episode 1040 Average max agent Score: 0.04. Average duration 0.4s. Avg timestep reached 20.75
Episode 1060 Average max agent Score: 0.07. Average duration 0.5s. Avg timestep reached 25.7
Episode 1080 Average max agent Score: 0.08. Average duration 0.6s. Avg timestep reached 27.3
Noise variance: 0.04
Episode 1100 Average max agent Score: 0.09. Average duration 0.7s. Avg timestep reached 30.5
Episode 1120 Average max agent Score: 0.09. Average duration 0.6s. Avg timestep reached 29.15
Episode 1140 Average max agent Score: 0.07. Average duration 0.6s. Avg timestep reached 30.6
Episode 1160 Average max agent Score: 0.07. Average duration 0.6s. Avg timestep reached 32.05
Episode 1180 Average max agent Score: 0.09. Average duration 0.7s. Avg timestep reached 36.05
Noise variance: 0.04
Episode 1200 Average max agent Score: 0.13. Average duration 1.1s. Avg timestep reached 46.6
Episode 1220 Average max agent Score: 0.09. Average duration 0.7s. Avg timestep reached 35.4
Episode 1240 Average max agent Score: 0.12. Average duration 0.9s. Avg timestep reached 45.4
Episode 1260 Average max agent Score: 0.11. Average duration 1.0s. Avg timestep reached 48.45
Episode 1280 Average max agent Score: 0.14. Average duration 1.0s. Avg timestep reached 51.4
Noise variance: 0.04
Episode 1300 Average max agent Score: 0.09. Average duration 0.8s. Avg timestep reached 35.3
Episode 1320 Average max agent Score: 0.10. Average duration 0.8s. Avg timestep reached 37.75
Episode 1340 Average max agent Score: 0.15. Average duration 1.2s. Avg timestep reached 60.6
Episode 1360 Average max agent Score: 0.60. Average duration 4.6s. Avg timestep reached 80.5

Episode 1380 Average max agent Score: 0.21. Average duration 1.8s. Avg timestep reached 90.15
Noise variance: 0.04
Episode 1400 Average max agent Score: 0.33. Average duration 2.8s. Avg timestep reached 131.35
Episode 1420 Average max agent Score: 0.33. Average duration 2.7s. Avg timestep reached 133.95
Episode 1440 Average max agent Score: 0.45. Average duration 3.6s. Avg timestep reached 155.1
Episode 1460 Average max agent Score: 0.51. Average duration 3.8s. Avg timestep reached 147.7
Episode 1480 Average max agent Score: 0.67. Average duration 5.0s. Avg timestep reached 155.4
Episode 1490 Max (over agents) episode score: 0.40. Duration: 3.0s
Environment solved in 1390 episodes! Average mean score over last 100 episodes: 0.50

DONE ----------
Total time consumed: 17.0m

critic Critic(
(bn0): BatchNorm1d(48, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
(fcs1): Linear(in_features=48, out_features=200, bias=True)
(fc2): Linear(in_features=204, out_features=100, bias=True)
(fc3): Linear(in_features=100, out_features=1, bias=True)
)
actor Actor(
(bn0): BatchNorm1d(24, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
(fc1): Linear(in_features=24, out_features=64, bias=True)
(fc2): Linear(in_features=64, out_features=64, bias=True)
(fc3): Linear(in_features=64, out_features=2, bias=True)
)
critic_optim Adam (, Parameter Group 0, amsgrad: False, betas: (0.9, 0.999), eps: 1e-08, lr: 0.001, weight_decay: 0.0001,)
actor_optim Adam (, Parameter Group 0, amsgrad: False, betas: (0.9, 0.999), eps: 1e-08, lr: 0.001, weight_decay: 0.0001,)
clip_grad_norm True
batch_size 512
max_t 500
time_steps_before_training 5
num_trainings_per_update 2
num_episodes_to_increase_num_trainings 500
start_noise_variance 1.0
noise_decay 0.995
add_samples_only_if_high_reward False
episodes_to_make_target_equal_to_local 200