Project 3: Collaboration and competition

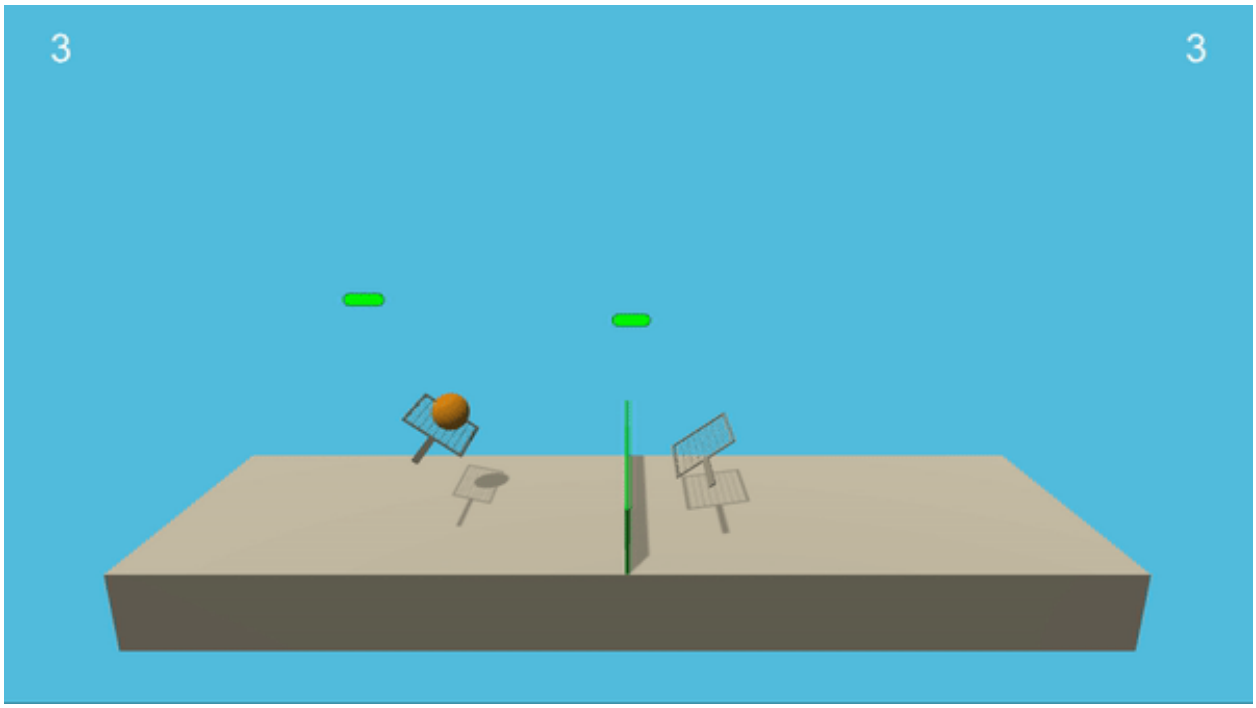

image of trained agents from udacity

Objective:

The objective of the project is to train two agents to play tennis .

The Environment

For this project, we will be using the Tennis environment from Unity ML agents. In this environment, two agents play with rackets to bounce a ball over a net. If an agent hits the ball over the net, it receives a reward of +0.1. If an agent lets a ball hit the ground or hits the ball out of bounds, it receives a reward of -0.01. Thus, the goal of each agent is to keep the ball in play.

Observation and Action Space:

```
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: ,
```

The observation space consists of 8 variables corresponding to the position and velocity of the ball and racket. The action space has two for each agent. the movement toward (or away from) the net and jumping are the two continuous actions available

Goal:

To solve the environment, your agents must get an average score of +0.5 (over 100 consecutive episodes, after taking the maximum over both agents).

Instruction to install the environment

Please install conda virtual environment with pytorch 3.6, unityagents and numpy

Select the environment that matches your operating system:

- Linux: click here
- Mac OSX: click here
- Windows (32-bit): click here
- Windows (64-bit): click here

To set up your python environment to run the code in this repository, follow the instructions below.

1. Create (and activate) a new environment with Python 3.6.
    - **Linux** or **Mac**:

2. conda create --name drlnd python=3.6

```
source activate drlnd
```

- **Windows**:

```
conda create --name drlnd python=3.6  activate drlnd
```

3. Clone the repository (if you haven't already!), and navigate to the python/ folder. Then, install several dependencies.

```
git clone https://github.com/udacity/deep-reinforcement-learning.git
cd deep-reinforcement-learning/python
pip install .
```

3. On Sucessfully installing the necessary packages and dependencies Launch the Project by navigating to the file p3_collab-compet/ where the tennis.ipynb notebook file is visible. Run the following cammand to launch the Jupyter environment to execute the notebook

```
jupyter notebook
```

open the tennis.ipynb notebook file to train the agent .

This repository

The github repository attached has solved the Tennis problem using Multi-Agent algorithm based on the DDPG (pendulum) code base with two Agents using pytorch and python.

Please download the github repository and use place all the files in a folder and open .ipynb file
https://github.com/sriten15/Project3_Multi_Agent_Learning

ipynb will install the python environment

1. Start the Environment

Run the next code cell to install a few packages. This line will take a few minutes to run!

```
: !pip -q install ./python
```

Following modules are imported and environment is setup and path stored in env.

```
from unityagents import UnityEnvironment
import numpy as np
%matplotlib inline

import torch
import numpy as np
from ddpg_agent import Agent

from collections import deque
import matplotlib.pyplot as plt
%matplotlib inline
import time, os
import pdb


env = UnityEnvironment(file_name="/data/Tennis_Linux_NoVis/Tennis")
```

Agents are trained:

```
scores = ddpg_multiple()

fig = plt.figure()
ax = fig.add_subplot(111)
plt.plot(np.arange(1, len(scores)+1), scores)
plt.ylabel('Score')
plt.xlabel('Episode #')
plt.show()
```


Training parameters are saved for both Agents (actor/critic)

```
#environment is solved is average score over last 100 episodes is above 0.5
if np.mean(scores_deque)>=0.5:
    print('\nEnvironment solved in {:d} episodes!\tAverage Score: {:.2f}'.format(i_epis
    torch.save(agents[0].actor_local.state_dict(), 'checkpoint_actor0.pth')
    torch.save(agents[0].critic_local.state_dict(), 'checkpoint_critic0.pth')
    torch.save(agents[1].actor_local.state_dict(), 'checkpoint_actor1.pth')
    torch.save(agents[1].critic_local.state_dict(), 'checkpoint_critic1.pth')
```