Deep Reinforcement Learning Nanodegree

Deep Deterministic Policy Gradient (DDPG) in Action: Multi-Agent Competition

Ayham Zaza

Contents

Introduction 2
Model Description 2
  Network Architecture 2
  Hyperparameters 3
  Results 3
Future Improvements 5
References 5

Introduction

- An Agent **G** located at **State S**, interacts with its surrounding *environment* by trying a **Action A** to transfer to a **New State S'**. A **Reward R** is associated with each action; and the goal of the agent is to navigate the environment until it reaches an *Objective* **O** while achieving a maximum possible reward. At a given state, the agent *experience* is summarized by the tuple (**S,A,R,S'**).

- Extending learning experience from a single Agent to Multiple-Agents challenging:
  - Training all Agents independently to learn individual policies by treating all other Agents as part of the environment. The environment will therefore change dynamically. This Non-Stationary constraint violates the learning assumption of single Agent learning and will make it difficult to converge.
  - Meta-Agent approach considers other interacting agents via a joint action vector. Not only the joint action space increases exponentially with the number of interactions, but also each Agent will have a different observation of the environment state if the environment is partially observable.
- In our environment setup, both Agents are fully aware of the environment. They are collaborating on achieving a group task by keeping the ball in the game for as much long time as possible. They are also competing in the sense that each Agent tries to win by maximizing their own rewards.
- The above scheme can be modeled by an Actor-Critic network. Each Actor has access to its own observations and actions. The Critic uses extra information obtained from States and Actions from all Agents.
- An introduction to the Actor-Critic Network was described in Project II.

Model Description

Network Architecture

- Actor-Critic DDPG network was implemented in PyTorch Linear module.

Actor Network:

```
self.fc1 = nn.Linear(state_size, fc1_units)
self.fc2 = nn.Linear(fc1_units, fc2_units)
#self.dropout = nn.Dropout(0.5) # not useful, the model is barely learning!
self.fc3 = nn.Linear(fc2_units, 128)
self.fc4 = nn.Linear(128, action_size)
```

- Input: 24 nodes – corresponding to state size
- Output: 2 nodes – corresponding to action size (followed by tanh activation)
- To enable learning complex non-linear functions, the 4 hidden layers (fc1, fc2,fc3,fc4) are followed by ***relu*** activation function.
- (fc1,fc2) were both chosen to be of size 256. The remaining dimensions are in the figure above.

Critic Network:

```
self.fcs1 = nn.Linear(state_size, fcs1_units)
self.fc2 = nn.Linear(fcs1_units+action_size, fc2_units)
self.fc3 = nn.Linear(fc2_units, 128)
self.fc4 = nn.Linear(128, 1)
```

- Input: 24 nodes – corresponding to state size
- Output: 1 nodes – corresponding to estimated Q_value
- To enable learning complex non-linear functions, the 4 hidden layers (fc1, fc2,fc3,fc4) are followed by ***relu*** activation function.
- (fc1,fc2) were both chosen to be of size 256. The remaining dimensions are in the figure above.

Hyperparameters

```
BUFFER_SIZE = int(1e5)  # replay buffer size
BATCH_SIZE = 256        # minibatch size
GAMMA = 0.99            # discount factor
TAU = 1e-3              # for soft update of target parameters
LR_ACTOR = 1e-4         # learning rate of the actor
LR_CRITIC = 1e-3        # learning rate of the critic

WEIGHT_DECAY = 0        # L2 weight decay
```

Results

The decision for the number of layers/hidden units, as well the choice of Actor-Critic network solved the environment by achieving the target score in 3928 episodes. figure_1

```
Episode 100     Average Score: 0.00
Episode 200     Average Score: 0.00
Episode 300     Average Score: 0.00
Episode 400     Average Score: 0.00
Episode 500     Average Score: 0.00
Episode 600     Average Score: 0.00
Episode 700     Average Score: 0.00
Episode 800     Average Score: 0.00
Episode 900     Average Score: 0.00
Episode 1000    Average Score: 0.00
Episode 1100    Average Score: 0.00
Episode 1200    Average Score: 0.00
Episode 1300    Average Score: 0.00
Episode 1400    Average Score: 0.00
Episode 1500    Average Score: 0.00
Episode 1600    Average Score: 0.00
Episode 1700    Average Score: 0.01
Episode 1800    Average Score: 0.02
Episode 1900    Average Score: 0.03
Episode 2000    Average Score: 0.05
Episode 2100    Average Score: 0.06
Episode 2200    Average Score: 0.07
Episode 2300    Average Score: 0.09
Episode 2400    Average Score: 0.09
Episode 2500    Average Score: 0.07
Episode 2600    Average Score: 0.10
Episode 2700    Average Score: 0.09
Episode 2800    Average Score: 0.10
Episode 2900    Average Score: 0.10
Episode 3000    Average Score: 0.10
Episode 3100    Average Score: 0.10
Episode 3200    Average Score: 0.10
Episode 3300    Average Score: 0.11
Episode 3400    Average Score: 0.11
Episode 3500    Average Score: 0.09
Episode 3600    Average Score: 0.17
Episode 3700    Average Score: 0.15
Episode 3800    Average Score: 0.20
Episode 3900    Average Score: 0.42
Episode 3928    Average Score: 0.50
Environment solved in 3928 episodes!    Average Score: 0.50
total time [min]:  32.65
```

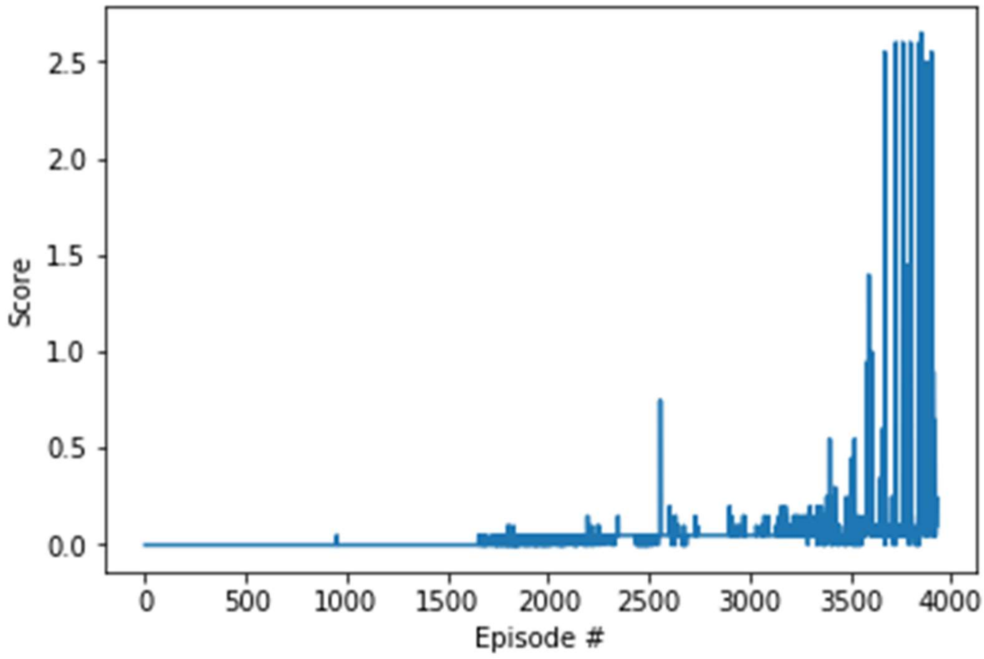

Figure 1: A plot of rewards per episode: the agent is able to receive an average reward (over 100 episodes) of at +13

Future Improvements

- Trying different network architectures
- Aiming Solving the environment with less episodes
- Solving the environment with higher average reward
- Trying other suggestions "Trust Region Policy Optimization - TRPO

References

- Udacity Deep Reinforcement learning Nanodegree Program. Lecture Notes and Videos. (https://www.udacity.com/course/deep-reinforcement-learning-nanodegree--nd893)
- https://github.com/udacity/deep-reinforcement-learning
- Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. R Lowe, et al. 2017
- Trust Region Policy Optimization. John Schulman et al. 2017
- Reinforcement Learning: An Introduction. Book by Richard S. Sutton and Andrew Barto. Second Edition. 2018