Deep Reinforcement Learning Nanodegree – Project 2 “Continuous Control”

Tim Wu

May 12th, 2020

**Introduction**

In this project, I built the reinforcement learning (RL) DDPG agent to solve Unity's Reacher and Crawler environment. In the following sessions of the report, I summarized how to fine tune and improve the learning of DDPG agent and how I selected best agent.

Reacher

* I am able to solve the Reacher environment in 273 episodes with average score 30.11.

**Reacher**

**Code Location**

* The Agent Class implemented in the (ddpg\_agent.py)
* The Deep Q-Network model implemented in the file (model.py)
* The model training and etc. implemented in the (Continuous\_Control.ipynb)
* The model weights are saved in the (Reacher\_ckpt\_path) folder

**Deep Deterministic Policy Gradient (DDPG) Implementation**

The solution is based on DDPG architecture – Actor Critic Network

The Network Architecture and Parameters:

[MODEL INFO] Actor initialized with parameters:

state\_size=33

action\_size=4

seed=123

fc1\_units=128

fc2\_units=128

[MODEL INFO] CRITIC initialized with parameters:

state\_size=33

action\_size=4

seed=123

fcs1\_units=128

fc2\_units=128

The Agent Hyperparameters:

[AGENT INFO] DDPG constructor initialized parameters:

# Set parameters for training

seed = 123 # random seed number

n\_episodes\_max = 1000 # number of training episodes

max\_t = 1000 # number of timesteps per episode

actor\_fc1\_units = 128 # actor network hidden layer #1 number of unit

actor\_fc2\_units = 128 # actor network hidden layer #2 number of unit

critic\_fcs1\_units = 128 # critic network hidden layer #1 number of unit

critic\_fc2\_units = 128 # critic network hidden layer #2 number of unit

BUFFER\_SIZE = int(1e6) # replay buffer size

BATCH\_SIZE = 128 # minibatch size

GAMMA = 0.99 # discount factor

TAU = 1e-3 # for soft update of target parameters

LR\_ACTOR = 2e-4 # learning rate of the actor

LR\_CRITIC = 2e-4 # learning rate of the critic

WEIGHT\_DECAY = 0.00 # L2 weight decay

OU\_MU = 0.0 # OUNoise mu

OU\_THETA = 0.15 # OUNoise theta

OU\_SIGMA = 0.18 # OUNoise sigma

UPDATE\_EVERY\_T\_STEPS = 20 # timesteps between updates

NUM\_OF\_UPDATES = 10 # num of update passes when updating

actor\_ckpt\_path = 'Reacher\_ckpt\_path/checkpoint\_actor.pth'

critic\_ckpt\_path = 'Reacher\_ckpt\_path/checkpoint\_critic.pth'

**Results**

The best performing agent is the rainbow network (Dueling + Double-DQN + Experience Replay + Clip Error). This agent has all the improvement and it can solve the environment in 248 episodes. The file with the saved model weights of the best agent saved in the checkpoint folder and named rainbow\_dqn\_v1\_checkpoint.pth.

**The Best Agent Navigation Result:**

Episode 100 (24min) Moving Average Score (over time window): 3.24

Episode 200 (50min) Moving Average Score (over time window): 16.59

Environment solved in 273 episodes! Average Score: 30.11

A screenshot of a map

Description automatically generated

**Future Improvements**

1. Extensive hyperparameter optimization, fine tune the experience replay feeding buffer size and update frequency
2. Add prioritized experience replay

**Reference**

1. <https://ai.atamai.biz/> - Reinforcement Learning with Pytorch course slides
2. <https://www.freecodecamp.org/news/improvements-in-deep-q-learning-dueling-double-dqn-prioritized-experience-replay-and-fixed-58b130cc5682/>
3. <https://chan-y-park.github.io/blog/rl_atari_part_2.html#.XqjgYNNKhTZ>
4. Human-level control through deep reinforcement learning [Nature. 2015]