9/30/2019

Udacity Deep Reinforcement Learning Nanodegree
Project 1: Navigation

Hiroshi Ashikaga, MD, PhD
ha8000@gmail.com

1. Computational resource

Hardware	MacBook Air (11-inch, Early 2014)
CPU	1.7 GHz Intel Core i7
Memory	8 GB 1600 MHz DDR3
Graphics	Intel HD Graphics 5000 1536 MB

2. Learning Algorithm

a. Deep Q-Learning (DQL) algorithm (reference: https://classroom.udacity.com/nanodegrees/nd893/parts/6b0c03a7-6667-4fcf-a9ed-dd41a2f76485/modules/4eeb16ab-5ac5-47bf-974d-12784e9730d7/lessons/a6829f14-5ef0-4b4a-83ed-234029c5cc60/concepts/637ff801-c1e1-4eb8-90cb-c9bcda92ca77)

- Initialize replay memory D with capacity N
- Initialize action-value function $\hat{q}$ with random weights w
- Initialize target action-value weights $w^- \leftarrow w$
- For the episode $e \leftarrow 1$ to M:
  - Initial input frame x_1
  - Prepare initial state: $S \leftarrow \phi(<x_1>)$
  - For time step $t \leftarrow 1$ to T:

    ----- SAMPLE -----
    - Choose action A from status S using policy $\pi \leftarrow \epsilon$-Greedy($\hat{q}(S, A, w)$)
    - Take action A, observe reward R, and next input frame x_{1+1}
    - Prepare next state: $S' \leftarrow \phi(<x_{1-2}, x_{1-1}, x_1, x_{1+1}>)$
    - Store experience tuple (S, A, R, S') in replay memory D
    - $S \leftarrow S'$

    ----- LEARN -----
    - Obtain random minibatch of tuples (s_j, a_j, r_j, s_{j+1}) from D
    - Set target $y_j = r_j + \gamma \max_a \hat{q}(s_{j+1}, a, w^-)$
    - Update: $\Delta w = \alpha(y_j - \hat{q}(s_j, aj, w))\nabla_w \hat{q}(s_j, aj, w)$
    - Every C steps, reset: $w^- \leftarrow w$

b. Chosen hyperparameters

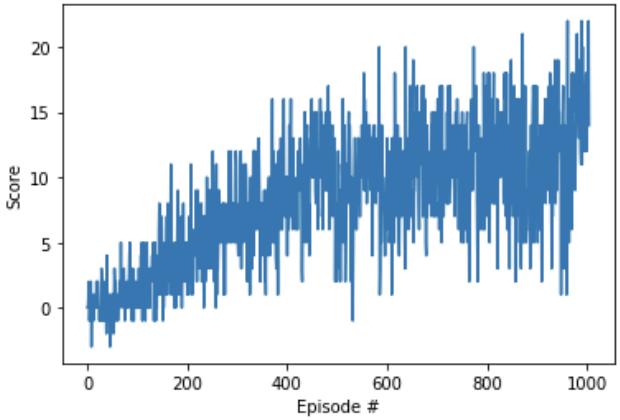
```
BUFFER_SIZE = int(1e5)  # replay buffer size (default 1e5)
BATCH_SIZE = 64         # minibatch size (default 64)
GAMMA = 0.9965          # discount factor (default 0.99)
TAU = 1e-3              # for soft update of target parameters (default 1e-3)
LR = 5e-4               # learning rate (default 5e-4)
UPDATE_EVERY = 4        # how often to update the network (default 4)
```

c. Neural network architecture

```
Input nodes 37 (number of state space dimension)
                        ↓
Fully connected layer nodes 1024 (Relu activation)
                        ↓
Fully connected layer nodes 1024 (Relu activation)
                        ↓
Output nodes 4 (number of action space dimension)
```

3. Plot of Rewards

```
Episode 100     Average Score: 0.73
Episode 200     Average Score: 2.91
Episode 300     Average Score: 5.82
Episode 400     Average Score: 7.73
Episode 500     Average Score: 9.890
Episode 600     Average Score: 9.67
Episode 700     Average Score: 10.60
Episode 800     Average Score: 10.83
Episode 900     Average Score: 11.52
Episode 1000    Average Score: 12.67
Episode 1005    Average Score: 13.00
Environment solved in 905 episodes!     Average Score: 13.00
```



4. Ideas for Future Work

- Need to understand how best to optimize the neural network architecture according to the complexity of the task.
- Need to try the optional challenge: Learning from Pixels to understand why learning directly from pixels may improve performance better.