Lab 4: Deep Q Network

Lab Objective:

In this project, you need to design/construct and train a neural network (it's simple, very simple, take it easy!) that is able to play CartPole-v0. The training method is deep Q-learning, which is a variation of Q- learning.

We want you to learn how to combine Q-learning with neural network to do reinforcement learning and use experience replay mechanism to break the correlations and reduce the variance of the updates

Environment Setup:

- Python 3.5+
- Pytorch $\geq$ 0.2.0
- OpenAI-gym
  - pip3 install gym[all]

Game Environment – CartPole-v0:

- Introduction: A pole is attached by an un-actuated joint to a cart, which moves along a frictionless track. The system is controlled by applying a force of +1 or -1 to the cart. The pendulum starts upright, and the goal is to prevent it from falling over. A reward of +1 is provided for every time step that the pole remains upright. The episode ends when the pole is more than 15 degrees from vertical, or the cart moves more than 2.4 units from the center.
- Actions:
  - Type: Discrete (2)
  - 0(left)or 1(right)
- Observation:
  - Type: Box (4)
  - [Cart Position, Cart Velocity, Pole Angle, Pole Velocity at Tip]
- Reward
  - Every timestep +1

Implementation Details:

Network Architecture (it's simple)

- Input: Observation (4 elements, not images)
- First layer:
  - fully connected layer (relu)
  - weight initial: **He's weight init**
  - input: 4, output: 32
- Second layer:
  - fully connected layer (relu)
  - weight initial: **He's weight init**

  - input: 32, output: 2

Training Arguments

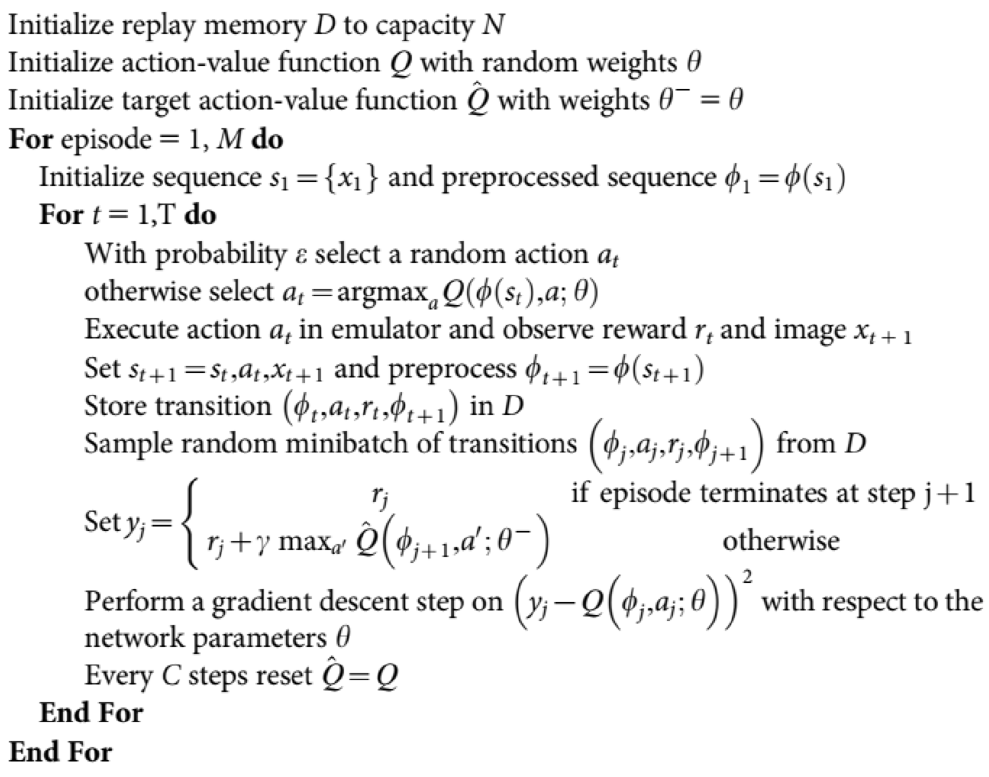
- Optimizer: Adam or RMSprop
  - Learning Rate: 0. 0005
- Epsilon: 1 → 0.1 or 1 → 0.01
- Epsilon decay: 0.995
- Batch Size: 128
- Experience buffer size (Memory capacity): 5000
- Gamma (Discount Factor): 0.95
- Training Episode: 1000
- Update target network every 50 iterations

<u>Requirements:</u>

1. Implement Deep Q Network
   - Construct the neural network
   - Target value, loss function
   - Action prediction with DQN
2. Implement Deep Q-learning learning algorithm
   - Calculate target Q values
   - Update algorithm for the network

<u>Methodology:</u>

Algorithm - Deep Q-learning with experience replay



```
Initialize replay memory $D$ to capacity $N$
Initialize action-value function $Q$ with random weights $\theta$
Initialize target action-value function $\hat{Q}$ with weights $\theta^- = \theta$
**For** episode = 1, $M$ **do**
  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$
  **For** $t$ = 1,T **do**
    With probability $\varepsilon$ select a random action $a_t$
    otherwise select $a_t = \text{argmax}_a Q(\phi(s_t), a; \theta)$
    Execute action $a_t$ in emulator and observe reward $r_t$ and image $x_{t+1}$
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $D$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $D$
    Set $y_j = \begin{cases} r_j & \text{if episode terminates at step j+1} \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^-) & \text{otherwise} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ with respect to the
    network parameters $\theta$
    Every $C$ steps reset $\hat{Q} = Q$
  **End For**
**End For**
```

Rule of Thumb:

1. The performance should greatly improve after training about 300 episodes. (reaching 100 points or more)

Scoring Criteria:

Please hand in your source code and report, and demo to TAs.

- **Report (70%)**
  - A plot shows episode rewards of 1000 training episodes (20%)
  - Describe your implement of network structure & Loss function (10%)
  - Describe the way you implement of epsilon-greedy action select method (10%)
  - Describe how you implement the training process of deep Q learning (10%)
  - Describe how the code work (the whole code, not only DQN algorithm) (10%)
  - Upload the best video during the training process (10%)
  - More you want to say
- **Performance** – average reward during testing **(30%)**
  - **Your score = average reward / 2.0**
- **Bonus (10~15%)**
  - Implement Double DQN algorithm
  - Other study or improvement for the project.

References:

[1] Mnih, Volodymyr, et al. "Playing atari with deep reinforcement learning." arXiv preprint arXiv:1312.5602 (2013).

[2] Mnih, Volodymyr, et al. "Human-level control through deep reinforcement learning." Nature 518.7540 (2015): 529-533.

[3] Van Hasselt, Hado, Arthur Guez, and David Silver. "Deep Reinforcement Learning with Double Q-Learning." AAAI. 2016.

[4] Openai gym Documentation

[5] CartPole v0 wiki

[6] Reinforcement Learning (DQN) tutorial

[7] 2017-fall-DL-training-program/Setup_tutorial

[8] DQN_Reinforcement_learning