DS595/CS525 FALL
Project 3 - Deep Q-learning

10/3/2019

Outline

- Introduction
    - Game Playing : Breakout
- Deep Reinforcement Learning
    - Deep Q-Learning (DQN)
    - Improvements to DQN
- Grading & Format
    - Grading Policy
    - Code Format
    - Submission
- Google Cloud Platform & Pytorch Tutorial *(12-13 hours on GPU)*

Introduction

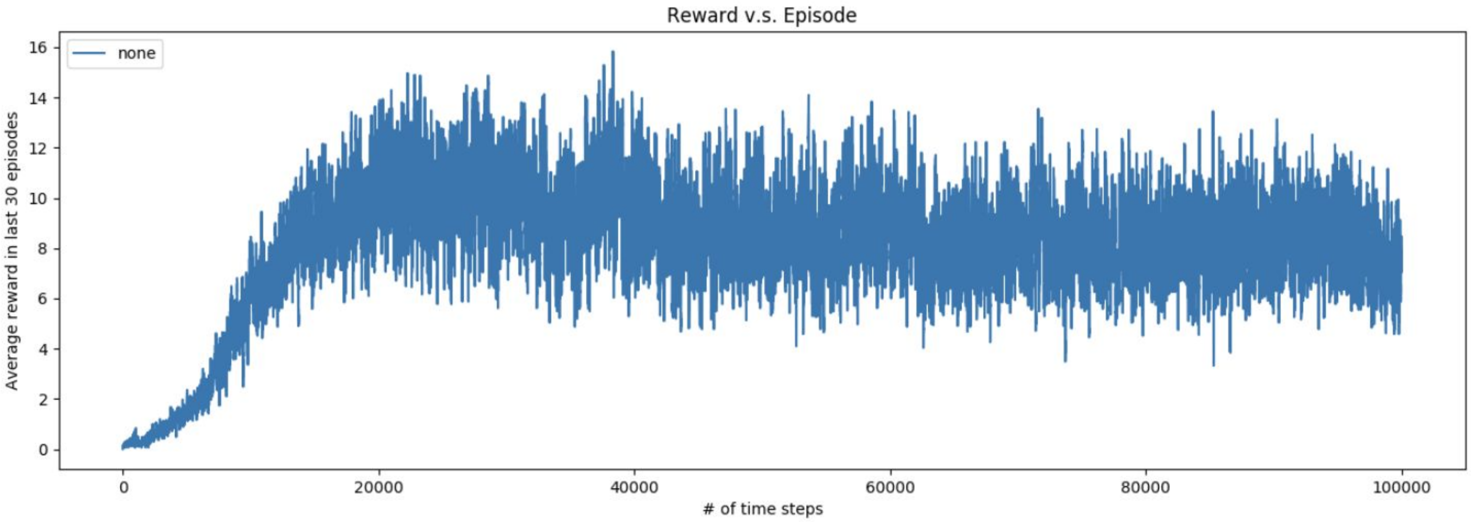
Environment

Breakout

- Get average reward >= 40 in 100 episodes
- With OpenAI’s Atari wrapper & reward clipping
  - We will unclip the reward when testing

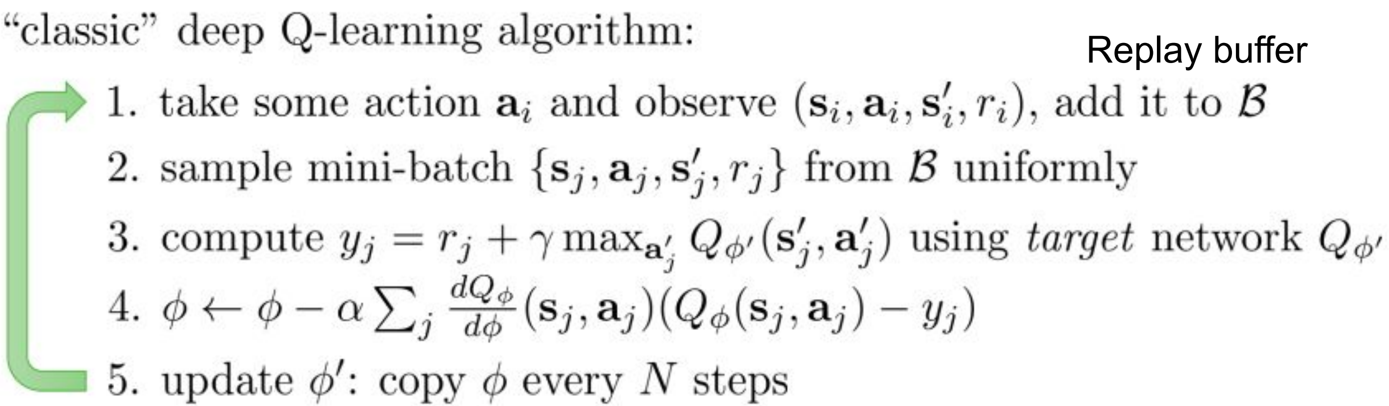
Introduction

Training Plot



- X-axis : number of training steps
- Y-axis : average clipped reward in last 30 episodes

Deep Q-Learning (DQN)



"classic" deep Q-learning algorithm:

```
1. take some action $\mathbf{a}_i$ and observe $(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)$, add it to $\mathcal{B}$
2. sample mini-batch $\{\mathbf{s}_j, \mathbf{a}_j, \mathbf{s}'_j, r_j\}$ from $\mathcal{B}$ uniformly
3. compute $y_j = r_j + \gamma \max_{\mathbf{a}'_j} Q_{\phi'}(\mathbf{s}'_j, \mathbf{a}'_j)$ using *target* network $Q_{\phi'}$
4. $\phi \leftarrow \phi - \alpha \sum_j \frac{dQ_\phi}{d\phi}(\mathbf{s}_j, \mathbf{a}_j)(Q_\phi(\mathbf{s}_j, \mathbf{a}_j) - y_j)$
5. update $\phi'$: copy $\phi$ every $N$ steps
```

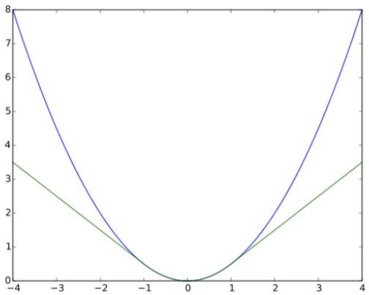
Replay buffer

Deep Q-Learning (DQN)

- The action should act ε-greedily
    - Random action with probability ε

- Linearly decline ε from 1.0 to some small value, say 0.025
    - Decline per step

- Hyperparameters (just suggestion)
    - Replay Buffer Memory Size 10000 *(deque)*
    - Start to learn 5000
    - Perform Update Target Network Step 5000
    - Learning Rate 1.5e-4, Batch Size 32
    - Adam
    - Huber Loss *(F.smooth_l1_loss)*
    - Clip gradients between (-1,1)

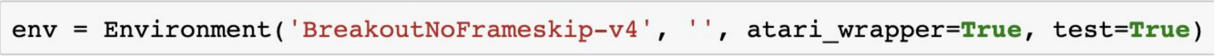


Green is the Huber loss and blue is the quadratic loss (Wikipedia)

$$L_\delta(a) = \begin{cases} \frac{1}{2}a^2 & \text{for } |a| \le \delta, \\ \delta(|a| - \frac{1}{2}\delta), & \text{otherwise.} \end{cases}$$

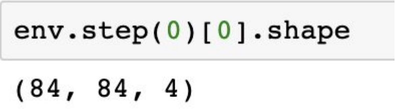
Why Reward is clipped

- Performing the same action for 4 frames
  - To use data more efficiently



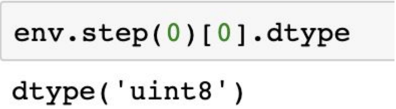
```
env = Environment('BreakoutNoFrameskip-v4', '', atari_wrapper=True, test=True)
```

- Reward may be up to 4
  - If positive, clip to 1 → reduce variance



```
env.step(0)[0].shape
```

```
(84, 84, 4)
```



```
env.step(0)[0].dtype
```

```
dtype('uint8')
```

- How to see your unclipped reward
  1. Use the *test* function
  2. Turn off the *clip_reward* option of your environment and do the clipping by yourself.

Deep Reinforcement Learning

Improvements to DQN

- Double Q-Learning
- Dueling Network
- Prioritized Replay Memory
- Noisy DQN
- Distributional DQN

https://arxiv.org/pdf/1710.02298.pdf

Deep Reinforcement Learning

Other Training Tips

- How to use Pytorch
- Official DQN Pytorch Tutorial
- DQN Tutorial on Medium
- Official DQN paper

Grading & Format

Grading Policy

- Python code (20 points)
- Trained Model (50 points)
  - Get averaging reward >= 40 in 100 episodes in **Breakout**
  - With OpenAI's Atari wrapper & reward clipping
    - We will unclip the reward when testing
- PDF Report (30 points)
  - Describe your DQN model
  - Plot the learning curve *(training steps can defined by yourself)*
    - X-axis: number of training steps
    - Y-axis: average clipped reward in last 30 episodes

Grading & Format

Code Format

- Please download the sample files from github
- Follow the instructions in README to install required packages
- **Six** functions you should implement in agent_dqn.py
  1. __init__(self, env, args)
  2. init_game_setting(self)
  3. make_action(self, state, test)
  4. train(self)
  5. push(self)
  6. repaly_buffer(self)
- **DO NOT** add any parameter in __init__(), init_game_setting() and make_action()
- You can change the seed
- You can add new functions in the agent_dqn.py

Grading & Format

Code Format

- **Two** functions you should implement in dqn_model.py
    1. __init__(self)
    2. forward(self, x)
- You can add parameters in these two functions
- You can add new functions in the dqn_model.py
- You can add your arguments in argument.py
- Please don’t revise test.py, environment.py and agent.py

Grading & Format

Deliverables

- Deadline: **Thursday 17/10/2019 23:59**
- Your submission **MUST** have following files
  - agent_dqn.py, dqn_model.py, argument.py, atari_wrapper.py
  - [saved_model_file]
  - report.pdf
  - README (optional)
  - download.sh (optional)
  - other files you need
- If your model is too large for canvas, upload it to a cloud space and write download.sh to download the model

Grading & Format

Package

- Please use Python3
- The TA will execute ‘python test.py --test_dqn’ to run your code on **ubuntu+GPU**
- The execution for the model should be done within 20 minutes, excluding model download
- Allowed packages
  a. PyTorch
  b. Numpy
  c. Scipy
  d. Pandas
  e. Python Standard Lib
- If you use other packages, please ask for permission first

Google Cloud Platform

- How to use Google Cloud Platform
- How to use Pytorch on GPU

- Naive Pytorch tutorial

Related Materials

- Course & Tutorial:
  - Berkeley Deep Reinforcement Learning, Fall 2017
  - David Silver RL course
  - Nips 2016 RL tutorial
- Blog:
  - Andrej Karpathy's blog
  - Arthur Juliani's Blog
- Text Book:
  - Reinforcement Learning: An Introduction
- Repo:
  - https://github.com/williamFalcon/DeepRLHacks