DS551/CS551/CS525 2024 Fall Project 3 - Deep Q-learning

10/6/2024

Outline

- Introduction
  - Game Playing : Breakout
- Deep Reinforcement Learning
  - Deep Q-Learning (DQN)
  - Improvements to DQN
- Grading & Format
  - Grading Policy
  - Code Format
  - Submission
- WPI Turing or Google Cloud Platform & Pytorch Tutorial

Introduction

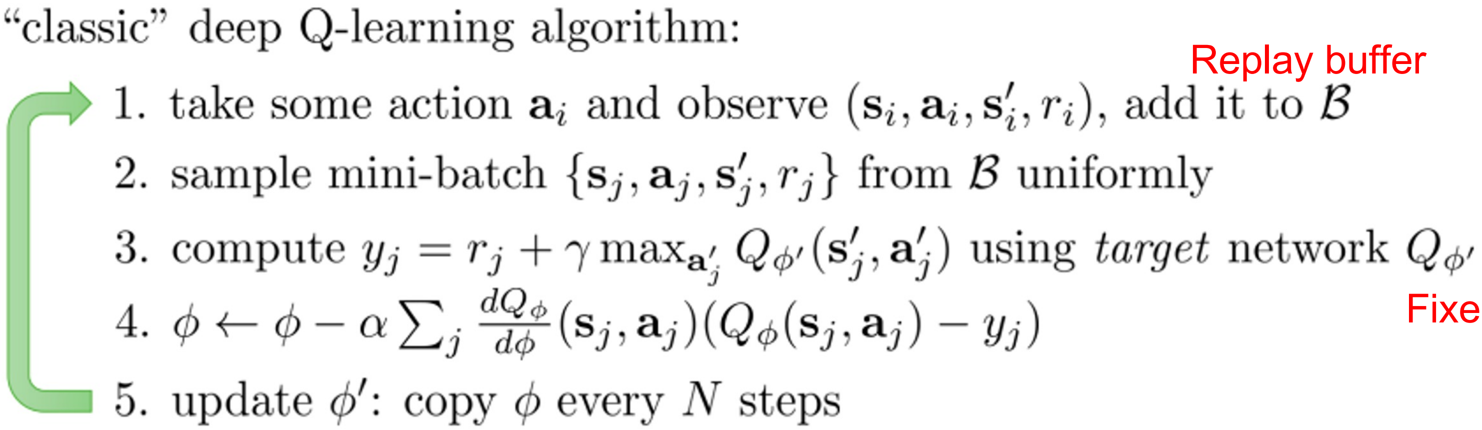
Environment

Breakout

- Get average reward >= 40 in 100 episodes (5 lives per episode)
- In testing, we consider each episode with its all 5 lives
- With OpenAI's Atari wrapper (modified by us a little bit)

Deep Reinforcement Learning

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 ϕ': copy ϕ every N steps

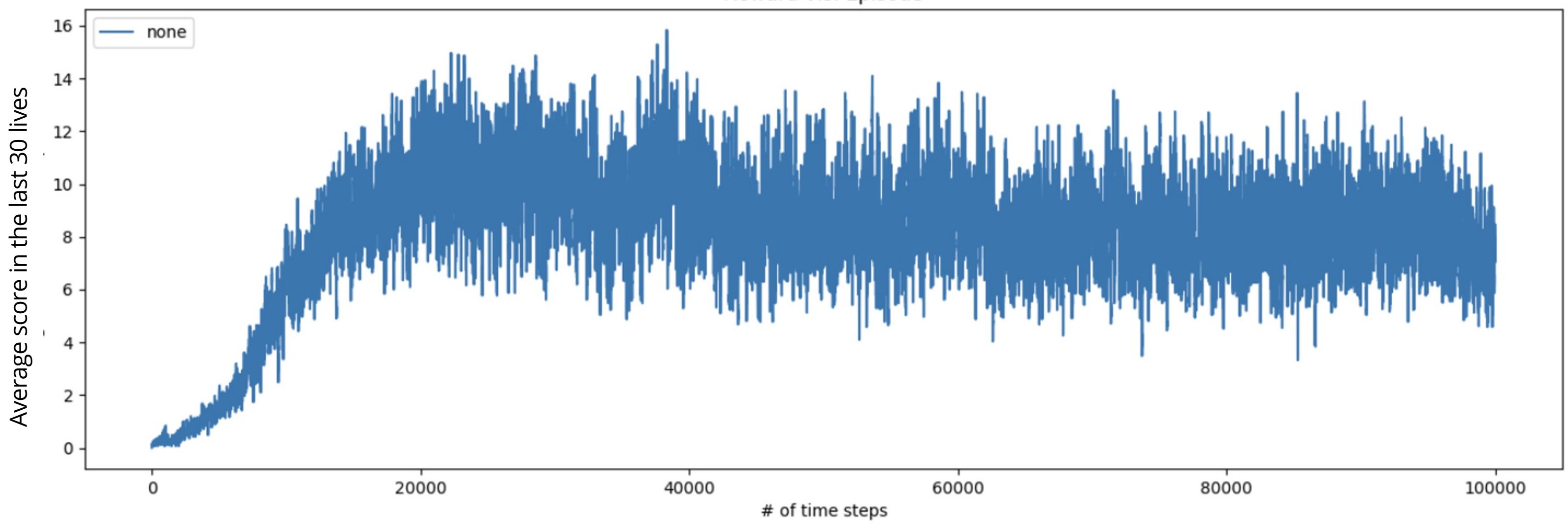
Replay buffer

Fixed targe-Q

Introduction

Training Plot

Training curve



- X-axis : number of training steps
- Y-axis : average reward in every 30 **lives** (not 30 complete episodes).

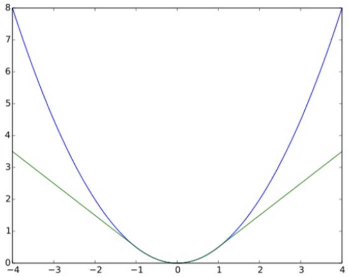
Deep Reinforcement Learning

Deep Q-Learning (DQN)

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

- Linearly decay ε from 1.0 to some small value, say 0.025
  - Decay per step:(epsilon - epsilon_min) /number of epsilon step

- Hyperparameters (just suggestion)
  - Replay Buffer Memory Size 10,000 *(deque)*
  - Start to train DQN with buffer size 5000
  - Update Target Network every 5000 steps
  - Learning Rate 1.5e-4, Batch Size 32
  - Adam optimizer
  - Huber Loss *(F.smooth_l1_loss)*
  - Clip gradients between (-1,1)



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

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

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
- See more tips on project website
- https://github.com/lllyyyt123/WPI-DS551-Fall23/blob/main/Project3/README.md

Grading & Format

Grading Policy

- Python code (20 points)
- Trained Model (50 points)
  - Get averaging reward >= 40 in 100 episodes (each of 5 lives) in **Breakout**
  - With OpenAI’s Atari wrapper
- PDF Report (30 points)
  - Describe your DQN model
  - Screenshot of the average score in 100 episodes
  - Plot the training curve *(training steps can defined by yourself)*
    - X-axis: number of training steps
    - Y-axis: average reward in last 30 lives

Grading & Format

Code Format

- Please download all the .py files from project github page
- Follow the instructions in README to install 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 (if needed)

- Please do not change test.py, main.py, environment.py, atari_wrapper.py and agent.py

Grading & Format

Deliverables

- Deadline: **Tuesday Oct 29, 2024 (23:59)**
- Your submission **MUST** have following files
    - agent_dqn.py, dqn_model.py, argument.py
    - [saved_model_file] (.pth file)
    - report.pdf
    - README (with details of what files you have modified.)
    - other files you need
- If your model is too large for canvas, upload it to a cloud space (like dropbox, google drive) and provide the link to download the model

Grading & Format

Package

- Please use Python3
- The TA will execute ‘python main.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
    f. etc.

Setup

- Recommended programming IDE (integrated development environment): VS code (See install VS code)
- Install Miniconda
- Install Python 3, by default, it's Python 3.11.4.
- For more details, please refer to project 3 website

https://github.com/UrbanIntelligence/WPI-DS551-Fall24/tree/main/Project3

Environment Preparation

- GPU resources:

  1. How to use WPI Turing GPUs with your WPI account

  2. Google Cloud https://cloud.google.com/gpu

backup