CSE 4/546: Reinforcement Learning
Spring 2023

Instructor: Alina Vereshchaka

Assignment 2 - Deep Q-Networks

Checkpoint 1: March 9, Thu, 11:59pm
Checkpoint 2: March 30, Thu, 11:59pm
Due Date: April 6, Thu, 11:59pm

1 Assignment Overview

The goal of the assignment is to work with value function approximation algorithms, to explore OpenAI Gym environments and getting experience working with a project-management tool. In the first part of the project we will explore OpenAI gym library. In the second part we will implement Deep Q-learning (DQN) following DeepMind's paper that explains how reinforcement learning algorithms can learn to play Atari from raw pixels. In the third part of the project we will implement an improvement to the DQN algorithm. The purpose of this assignment is to understand the benefits of approximation methods, the role of deep neural networks as well as some of the techniques used in practice to stabilize training and to achieve better performance. We will train our networks on the grid-world and OpenAI gym environments.

For this assignment, libraries with in-built RL methods cannot be used. Submissions with used in-built RL methods (e.g. stable-baselines, keras-RL, TF agents, etc) will not be evaluated.

Part I [20 points] - Introduction to Deep Reinforcement Learning

The goal of this part is to make you comfortable with the application of different neural network structures depending on how the reinforcement learning environment is set up.

Working with various neural network setups

Refer to the Jupyter notebook titled "assignment_2_part_1.ipynb". You would have to work with an implementation of the Wumpus World environment (environment.py) with four types of observation and three types of actions. Your task is to setup neural networks based on the structure provided to take the observation as input and provide the Q-values, action probabilities, or the action as output.

Refer to the Juypter notebook for more details.

Part I submission includes

- Jupyter Notebook (assignment_2_part_1.ipynb) – with saved outputs

Part II [50 points] - Implementing DQN & Solving Various Problems

2.1 Implementing DQN & Solving grid-world environment

Implement DQN from scratch following DeepMind's paper (Nature paper or initial version). You may use any framework (Keras/Tensorflow/Pytorch).

Apply DQN to solve the environment you or your teammate defined in Assignment 1. You can make slight improvements to the original environment, if required. Save the weights of the trained model.

2.2 Applying DQN to solve various RL problems

Test your implemented DQN algorithm in Part 2.1 on TWO environments:'CartPole-v1' and any other complex environment of your choice.
For the second environment you may use your custom made multi-agent environment or any other complex environment that you will use for your Final Project (this has to be approved by the course instructors). The environment with multiple versions is considered one environment. Provide performance results including reward dynamics (total reward per episode).

'CartPole-v1' is one of the classical RL environments that often used as a benchmark problem to check the algorithm performance. 'CartPole-v1' environment is considered to be solved if it is getting an average reward of more than 470 points over 100 consecutive episodes during evaluation.

Suggested second environment to work with: OpenAI LunarLander, OpenAI MountainCar, OpenAI Atari Breakout

In your report for Part II:

1. Discuss the benefits of:
   - Using experience replay in DQN and how its size can influence the results
   - Introducing the target network
   - Representing the Q function as $\hat{q}(s, \mathbf{w})$
2. Briefly describe 'CartPole-v1', the second complex environment, and the grid-world environment that you used (e.g. possible actions, states, agent, goal, rewards, etc). You can reuse related parts from your Assignment 1 report.
3. Show and discuss your results after applying your DQN implementation on the three environments. Plots should include epsilon decay and the total reward per episode.
4. Provide the evaluation results. Run your agent on the three environments for at least 5 episodes, where the agent chooses only greedy actions from the learnt policy. Plot should include the total reward per episode.
5. Provide your interpretation of the results. E.g. how the DQN algorithm behaves on different envs.
6. Include all the references that have been used to complete this part

Part II submission includes

- Report (as a pdf file)
- Jupyter Notebook (.ipynb) with all saved outputs. Do not combine with the Juypter notebook from Part I.
- Saved weights of the trained models as pickle files or .h5 for each model and each environment.

Part III [30 points] - Improving DQN & Solving Various Problems

3.1 Improving vanilla version of DQN

There have been many improvements developed for the vanilla algorithm. In this part we will implement one of improved algorithms that is based on DQN. Modify your DQN code from Part 2.1 to one of the improved versions, apply it to the grid-world environment from Part 2.1 and compare the results.

Possible algorithms to implement include:

- Double DQN
- Dueling DQN
- Prioritized Experience Replay (PER)

3.2 Applying improved version of DQN algorithm to solve TWO environments

Apply your improved DQN algorithm implemented in Part 3.1 to solve 'CartPole-v1' and the second complex environment from Part 2.2. Compare the results with the performance of DQN vs Improved version of DQN. Provide reward dynamics (total reward per episode).

In your report for Part III:

1. Discuss the algorithm you implemented.
2. What is the main improvement over the vanilla DQN?
3. Show and discuss your results after applying your the two algorithms implementation on the environment. Plots should include epsilon decay and the total reward per episode.
4. Provide the evaluation results. Run your environment for at least 5 episodes, where the agent chooses only greedy actions from the learnt policy. Plot should include the total reward per episode.
5. Compare the performance of both algorithms (DQN & Improved version of DQN) on the same environments (e.g. show one graph with two reward dynamics) and provide your interpretation of the results. Overall three rewards dynamics plots with results from two algorithms applied on:
   - Grid-world environment
   - 'CartPole-v1'
   - Gymnasium envs or Multi-agent environment or any other complex environment
6. Provide your interpretation of the results. E.g., how the same algorithm behaves on different environments, or how various algorithms behave on the same environment.
7. Include all the references that have been used to complete this part

Part III submission includes

- Report (as a pdf file) – combined with your report from Part II
- Jupyter Notebook (.ipynb) with all saved outputs. It can be combined with the Jupyter notebook from Part II. Do not combine with the Jupyter notebook from Part I.
- Saved weights of the trained models as pickle files or .h5 for each model and each environment.

Extra Points [max +10 points]

- **Solving Image-based Enviroment [7 points]**
  Use one of the environments with image representation of the state that requires to utilize CCN (Convolution Neural Network) for the state preprocessing (e.g. OpenAI Breakout).
- **Project Management [3 points]**
  Project management includes the application of specialized skills, knowledge, processes and procedures to successfully take an assignment from start to finish. These tools can be used for collaboration, status reporting, time management and many other cases. There is an extensive range of tools available with free tiers available for small team usage. You can choose any of the following: Trello, Basecamp.

  Requirements:

  - Create a board with at least these columns: To do, In Progress, Done
  - Divide the project to at least 20 steps/milestones (E.g. Implement DQN, Recheck grid-world env, Generate rewards dynamic graph, Report for Part 1: Q1/Q2/Q2, Submit Checkpoint / Final).
  - Add a link to the board in your report. Make sure your board is not private so that we can monitor it.
  - The board should be created at least one week before the checkpoint submission and every week there should be some activities
  - Include a few screenshots of your board activities as part of your Checkpoint and Final submissions.

4 Deliverables

Submit your work using UBLearns group in both cases if you work individually or in a team of two. There are two parts in your submission:

4.1 Report

The report should be delivered as a separate pdf file, and it is recommended for you to use the NIPS template to structure your report. You may include comments in the Jupyter Notebook, however you will need to duplicate the results in the separate pdf file.

Report should follow the following naming convention:
UBIT $TEAMMATE1\_TEAMMATE2\_assignment2\_report.pdf$
(e.g. $avereshc\_nitinvis\_assignment2\_report.pdf$

4.2 Code

Python is the only code accepted for this project. You can submit the code in Jupyter Notebook with the saved results. You can submit multiple files, but they all need to have a clear name. After executing command Jupyter Notebook, it should generate all the results and plots you used in your report and should be able to be printed out in a clear manner.

Name your Jupyter Notebooks following the pattern:
$TEAMMATE1\_TEAMMATE2\_assignment2\_part1.ipynb$
and $TEAMMATE1\_TEAMMATE2\_assignment2\_part2.ipynb$
(e.g. $avereshc\_nitinvis\_assignment2\_part1.ipynb$)

4.3 Saved weights

Submit the trained parameters as a pickle file or .h5 for all the models and environments, so that your obtained results can be fully replicated.

Name your pickle files or .h5 using the following pattern:
$TEAMMATE1\_TEAMMATE2\_assignment2\_part2\_dqn\_gridworld.pickle$

$TEAMMATE1\_TEAMMATE2\_assignment2\_part2\_dqn\_cartpole.pickle$
$TEAMMATE1\_TEAMMATE2\_assignment2\_part3\_ddqn\_cartpole.pickle$
(e.g. $avereshc\_nitinvis\_assignment2\_part3\_ddqn\_cartpole.pickle$)

5 References

- NIPS Styles (docx, tex)
- Overleaf (LaTex based online document generator) - a free tool for creating professional reports
- Gymnasium environments
- Lecture slides
- Human-level control through deep reinforcement learning
- Prioritized Experience Replay
- Deep Reinforcement Learning with Double Q-learning

6 ASSIGNMENT STEPS

1. Register your team (Due date: March 5)

- You may work individually or in a team of up to 2 people. The evaluation will be the same for a team of any size.
- Register your team at UBlearns > Tools > Groups.

2. Submit checkpoint I (Due date: March 9)

- Complete Part I
- Add your ipynb to zip file with UBIT $TEAMMATE1\_TEAMMATE2\_assignment2\_checkpoint\_1.zip$ (e.g. $avereshc\_nitinvis\_assignment2\_checkpoint\_1.zip$)
- Submit at UBLearns > Assignments
- Checkpoint will be evaluated after the final submission

3. Submit checkpoint II (Due date: March 30)

- Complete Part II
- Add your pdf and ipynb to zip file with UBIT $TEAMMATE1\_TEAMMATE2\_assignment2\_checkpoint\_2.zip$ (e.g. $avereshc\_nitinvis\_assignment2\_checkpoint\_2.zip$)
- Submit at UBLearns > Assignments
- Checkpoint will be evaluated after the final submission

4. Submit final results (Due date: April 6)

- Fully complete all parts of the assignment
- Add your pdf and ipynb for Part I, Part II, and Part III to a zip file $TEAMMATE1\_TEAMMATE2\_assignment2\_final.zip$ (e.g. $avereshc\_nitinvis\_assignment2\_final.zip$)
- Submit at UBLearns > Assignments

- The code of your implementations should be written in Python. You can submit multiple files, but they all need to be labeled clearly.
- Your Jupyter notebook should be saved with the results
- Include all the references that have been used to complete the assignment
- If you are working in a team of two, we expect equal contribution for the assignment. Each team member is expected to make a code-related contribution. Provide a contribution summary by each team member in a form of a table below. If the contribution is highly skewed, then the scores of the team members may be scaled w.r.t the contribution.

Team Member	Assignment Part	Contribution (%)

7 Academic Integrity

This assignment can be completed individually or in a team of two students. The standing policy of the Department is that all students involved in any academic integrity violation (e.g. plagiarism in any way, shape, or form) will receive an F grade for the course. The catalog describes plagiarism as "Copying or receiving material from any source and submitting that material as one's own, without acknowledging and citing the particular debts to the source, or in any other manner representing the work of another as one's own.". Updating the hyperparameters or modifying the existing code is not part of the assignment's requirements and will result in a zero. Please refer to the UB Academic Integrity Policy.

8 Important Information

This assignment can be completed in groups of two or individually. Teams can not be the same for A2 & A3 assignments.

9 Late Days Policy

You can use up to 5 late days throughout the course toward any assignments' checkpoint or final submission. You do not have to inform the instructor, as the late submission will be tracked in UBlearns. If you work in teams the late days used will be subtracted from both partners. E.g. you have 4 late days and your partner has 3 days left. If you submit one day after the due date, you will have 3 days and your partner will have 2 days left.

10 Important Dates

March 5, Thu 11:59pm - Register your team at UBlearns > Tools > Teams

March 9, Thu 11:59pm - Checkpoint I is Due

March 30, Thu 11:59pm - Checkpoint II is Due

April 6, Thu, 11:59pm - Assignment 2 is Due