CSE 4/546: Reinforcement Learning
Spring 2025

Instructor: Alina Vereshchaka

Assignment 2 - Deep Q-Networks

Checkpoint 1: Mar 6, Thu, 11:59pm
Checkpoint 2: Mar 27, Thu, 11:59pm
Due Date: Apr 3, Thu, 11:59pm

1 Assignment Overview

The goal of the assignment is to work with value function approximation algorithms, to explore Gymnasium environments and getting experience working with a project-management tool. In the first part of the assignment focuses on understanding the required Neural Network setups for environments with different observation and action spaces. In the second part of the project we will explore Gymnasium library and implement Deep Q-learning (DQN) following DeepMind's paper that explains how RL 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 Gymnasium environments. It is recommended that you use PyTorch to create and train your Neural Networks.

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. Exception is set only for Bonus - RL Libraries. Check the Bonus Part for more details.

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 RL environment is set up.

Working with various neural network setups

Refer to the Jupyter notebook titled "assignment_2_part_1.ipynb" on UBlearns. 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 Jupyter notebook on UBlearns for more details.

Part I submission includes

- Jupyter Notebook (assignment_2_part_1.ipynb) – with saved outputs
- There is no report assosiated with this part. You need to submit only .ipynb

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 during evaluation.

Suggested second environment to work with: LunarLander-v2, MountainCar-v0, CliffWalking-v0, FrozenLake-v1, Atari Pong, or Atari Breakout.

Note: You can use custom reward functions to train the agent. E.g., for MountainCar, the original reward function is very sparse which makes it difficult for the agent to converge to a good policy (Refer to the RL Handbook for the custom reward function for MountainCar.) To evaluate the trained agent's performance, you must use the original reward function to identify if the trained agent has met the environment's solve criteria.

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. **Grid World environments only:** Run your environment for 1 episode where the agent chooses only greedy actions from the learned policy. Render each step of the episode and verify that the agent has completed all the required steps to solve the environment. Save this render and include it in your report as clearly ordered screenshots or as a clearly named video file in your submission.
6. Provide your interpretation of the results. E.g. how the DQN algorithm behaves on different envs.
7. 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, .h5 or .pth for each model and each environment.
- **Grid World environments only:** Saved video file of 1 evaluation episode. (No need to include the video if you are including screenshots in your report instead.)

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 2.2 to one of the improved versions, apply it to the grid-world environment from Part 2.2 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).

Note:You can use custom reward functions to train the agent. E.g., for MountainCar, the original reward function is very sparse which makes it difficult for the agent to converge to a good policy (Refer to the RL Handbook for the custom reward function for MountainCar.) To evaluate the trained agent's performance, you must use the original reward function to identify if the trained agent has met the environment's solve criteria.

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. **Grid World environments only:** Run your environment for 1 episode where the agent chooses only greedy actions from the learned policy. Render each step of the episode and verify that the agent has completed all the required steps to solve the environment. Save this render and include it in your report as clearly ordered screenshots or as a clearly named video file in your submission.
6. 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

7. 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.
8. 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, .h5 or .pth for each model and each environment.
- **Grid World environments only:** Saved video file of 1 evaluation episode. (No need to include the video if you are including screenshots in your report instead.)

Bonus Part [max +10 points]

Solving Image-based Environment [7 points]

Apply your implementations to solve a more complex environment.

STEPS:

1. Select any Atari environment with image representation of the state that requires to utilize CNN (Convolution Neural Network) for the state preprocessing (e.g. Atari Breakout). This env should be different from the one use used for this assignment.
2. Apply both your implementations of DQN (Part II) and an improved version of DQN (Part III) to solve this environment. Use the same metrics and analysis.
3. In the report, create a new section at the end "Bonus: Atari". Include all the results, compare the results and provide your analysis.

SUBMISSION:

Submit your results as a Jupyter Notebook file named: $a2\_bonus\_atari\_TEAMMATE1\_TEAMMATE2.ipynb$
Include your analysis directly in the report.

RL Libraries [3 points]

STEPS:

1. Use the existing implementation of DQN in Stable-baselines3 or Ray RLLib to solve three environments that you used in Part 2 (e.g, your Grid World environment, CartPole-v1, and the second complex environment).
2. Compare the results with your scratch DQN implementation (Part 2) in terms of training time, episodes required, and cumulative reward per episode. Use the same metrics and analysis
3. In the report, create a new section at the end "Bonus: RL Libraries". Include all the results, compare the results and provide your analysis.

Note: You may use the RL in-built model only to complete this bonus task; using RL in-built models for other parts will not be evaluated.

SUBMISSION:

Submit your results as a Jupyter Notebook file named: *a2_bonus_rllib_TEAMMATE1_TEAMMATE2.ipynb*
Include your analysis directly in the report.

4 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

5 Assignment Steps

1(a). Register your team (Due date: Feb 25)

- 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 > Groups. You have to enroll in a team on UBLearns even if you are working individually. Enroll in an empty team where you'll be the only member.

1(b). For a team of 2 students (Due date: Mar 6)

- Create a private GitHub repository for the project and add our course GitHub account as a collaborator: @ub-rl
- Each team member should regularly push their progress to the repository. For example, you can sync the repository daily to reflect any updates or improvements made.
- In your report include a link to the repository along with the contribution table. Additionally, add screenshot(s) of your commit history.

1(c). Submission Format

- **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. Name your report as:
a2_report_TEAMMATE1_TEAMMATE2.pdf
(e.g. *a2_report_avereshc_nitinvis.pdf*)
- **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:
a2_part_1_TEAMMATE1_TEAMMATE2.ipynb
and *a2_part_2_TEAMMATE1_TEAMMATE2.ipynb*
(e.g. *a2_part_1_avereshc_nitinvis_.ipynb*)

- **Model Parameters:** Saved weights of the model(s) as a pickle file or .h5, so that the grader can fully replicate your results. Name your .pickle, .h5 or .pth files using the following pattern:
a2_part_2_dqn_gridworld_TEAMMATE1_TEAMMATE2.pickle
a2_part_2_dqn_cartpole_TEAMMATE1_TEAMMATE2.pickle
(e.g. *a2_part_3_ddqn_cartpole_avereshc_nitinvis.pickle*)

2. Submit Checkpoint I (Due date: Mar 6)

- Complete Part I
- Submit your .ipynb with saved outputs, named as
a2_part_1_TEAMMATE1_TEAMMATE2.ipynb
e.g. *a2_part_1_avereshc_nitinvis.ipynb*
- Submit at UBLearns > Assignments > Assignment 2

3. Submit Checkpoint II (Due date: Mar 27)

- Complete Part II
- For the checkpoint, it is okay if your report is not fully completed. You can finalize it for the final submission.
- 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, .h5 or .pth for each model and each environment.
- Add your code as .ipynb with saved outputs and saved weights to the zip folder. Submit your .zip folder named as:
assignment_2_checkpoint_2_TEAMMATE1_TEAMMATE2.zip
- Submit at UBLearns > Assignments
- Suggested file structure:

  assignment_2_checkpoint_2_TEAMMATE1_TEAMMATE2.zip
  - *a2_part_2_TEAMMATE1_TEAMMATE2.ipynb*
  - *a2_part_2_dqn_gridworld_TEAMMATE1_TEAMMATE2.h5*
  - *a2_part_2_dqn_cartpole_TEAMMATE1_TEAMMATE2.h5*
  - *a2_report_TEAMMATE1_TEAMMATE2.pdf*

4. Submit final results (Due date: Apr 3)

- Fully complete all parts of the assignment
- Add all your assignment files in a zip folder including ipynb files for Part I, Part II, Part III Bonus part (optional) and report.
- Submit at UBLearns > Assignments

- Suggested file structure (bonus part is optional):

  assignment_2_final_TEAMMATE1_TEAMMATE2.zip

  - *a2_part_1_TEAMMATE1_TEAMMATE2.ipynb*
  - *a2_part_2_TEAMMATE1_TEAMMATE2.ipynb*
  - *a2_part_2_dqn_gridworld_TEAMMATE1_TEAMMATE2.h5*
  - *a2_part_2_dqn_cartpole_TEAMMATE1_TEAMMATE2.h5*
  - *a2_part_3_TEAMMATE1_TEAMMATE2.ipynb*
  - *a2_part_3_ddqn_ENV1_TEAMMATE1_TEAMMATE2.h5*
  - *a2_part_3_ddqn_ENV2_TEAMMATE1_TEAMMATE2.h5*
  - *a2_report_TEAMMATE1_TEAMMATE2.pdf*
  - *a2_bonus_atari_TEAMMATE1_TEAMMATE2.ipynb*
  - *a2_bonus_rllib_TEAMMATE1_TEAMMATE2.ipynb*

- 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 (%)**

6 Grading Details

Checkpoint I Submission

- Graded on a 0/1 scale.
- A grade of "1" is awarded for completing more than 80% of the checkpoint I related part and "0" is assigned for all other cases.
- Receiving a "0" on a checkpoint submission results in a 10-point deduction from the final assignment grade.

Checkpoint II Submission

- Graded on a 0/1 scale.
- A grade of "1" is awarded for completing more than 80% of the checkpoint II related part and "0" is assigned for all other cases.
- Receiving a "0" on a checkpoint submission results in a 25-point deduction from the final assignment grade.
- Missing Checkpoint I and Checkpoint II will result in a fixed 35 points reduction from the final submission.

Final Submission

- Graded on a scale of X out of 100 points, with potential bonus points (if applicable).
- All parts of the assignment are evaluated during final evaluation.
- Ensure your final submission includes the final version of all assignment parts.

Important Notes

1. Only files submitted on UBlearns are considered for evaluation.
2. Files from local devices, GitHub, Google Colab, Google Docs, or other locations are not accepted.
3. Regularly submit your work-in-progress on UBlearns and always verify submitted files by downloading and opening them.

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

Feb 25, Thu 11:59pm - Register your team at UBLearns > Groups

Mar 6, Thu 11:59pm - Checkpoint I is Due

Mar 27, Thu 11:59pm - Checkpoint II is Due

Apr 3, Thu, 11:59pm - Assignment 2 is Due