CSE 4/546: Reinforcement Learning
Fall 2021

Instructor: Alina Vereshchaka

Assignment 1 - Defining & Solving RL Environments

Checkpoint: September 19, Sun, 11:59pm

Due Date: October 3, Sun, 11:59pm

1 Assignment Overview

The goal of the assignment is to acquire experience in defining and solving reinforcement learning environments, following OpenAI Gym standards. The assignment consists of two parts. The first focuses on defining deterministic and stochastic environments that are based on Markov decision process. In the second part we will apply two tabular methods to solve environments that were previously defined.

Part 1 [Total: 50 points] - Defining RL environments

1.1 Deterministic environment [25 points]

Define a deterministic environment, where $P(s', r|s, a) = \{0, 1\}$. Run a random agent for at least 10 timesteps to show that the environment logic is defined correctly.

Environment requirements:

- Min number of states: 12
- Min number of actions: 4
- Min number of rewards: 4

Environment definition should follow OpenAI Gym structure, which includes the following basic methods:

```
def __init__:
    # Initializes the class
    # Define action and observation space

def step:
    # Executes one timestep within the environment
    # Input to the function is an action

def reset:
    # Resets the state of the environment to an initial state

def render:
    # Visualizes the environment
    # Any form like vector representation or visualizing using matplotlib will be sufficient
```

1.2 Stochastic environment [25 points]

Define a stochastic environment, where $\sum_{s',r} P(s', r|s, a) = 1$. A modified version of the environment defined in Part 1.1 should be used. Run a random agent for at least 10 timesteps to show that the environment logic is defined correctly.

In your report for Part 1:

1. Describe the deterministic and stochastic environments, which were defined (set of actions/states/rewards, main objective, etc).
2. Provide visualizations of your environments.
3. How did you define the stochastic environment?
4. What is the difference between the deterministic and stochastic environments?
5. **Safety in AI**: Write a brief review explaining how you ensure the safety of your environments.

Part 2 [Total: 50 points] - Applying tabular methods

Apply **two tabular methods** to solve both the deterministic and stochastic environments that were defined in Part 1. You need to implement **Q-learning and any other tabular algorithm of your choice** (e.g. SARSA, Double Q-learning, Monte Carlo or n-step bootstrapping).

In your report for Part 2:

1. Show and discuss the results after:
   - Applying Q-learning to solve the deterministic environment defined in Part 1. Plots should include epsilon decay and total reward per episode.
   - Applying Q-learning to solve the stochastic environment defined in Part 1. Plots should include epsilon decay and total reward per episode.
   - Applying any other algorithm of your choice to solve the deterministic environment defined in Part 1. Plots should include total reward per episode.
   - Applying any other algorithm of your choice to solve the stochastic environment defined in Part 1. Plots should include total reward per episode.
   - Provide the evaluation results. Run your environment for at least 10 episodes, where the agent chooses only greedy actions from the learnt policy. Plot should include the total reward per episode.
2. Compare the performance of both algorithms on the same deterministic environment (e.g. show one graph with two reward dynamics) and give your interpretation of the results.
3. Compare how both algorithms perform in the same stochastic environment (e.g. show one graph with two reward dynamics) and give your interpretation of the results.
4. Briefly explain the tabular methods, including Q-learning, that were used to solve the problems. Provide their update functions and key features.

Recommended graphs that can help to debug the code

- Cumulative reward per episode. Ideally we want this to increase linearly over time (it can be non-linear for stochastic environments).
- Total reward per episode during training.

- Total reward per episode during evaluation. Ideally the graph is around the max reward that the agent can get within the episode.
- Percentage of episodes in which the agent achieves the goal. E.g. for 1000 training iterations, provide the percentage of goal achievements for every 100 iterations. You can use any size-window applicable to your setup.
- Average number of timesteps per episode.
- Average number of times our agent receives a penalty by going into the 'bad' state, if applicable.

Extra Points [max +10 points]

- **Early Bird Bonus [3 points]**
  Submit your final version of the assignment before **September 26, 11:59pm**.
- **Hyperparameter Tuning Bonus [5 points]**
  Provide the analysis after tuning at least **two hyperparameters** listed below:
  - Discount factor (γ)
  - Epsilon decay rate
  - Epsilon min/max values
  - Number of episodes
  - Max timesteps

  Try at least **3 different values** for each of the parameters that you choose. Provide the reward graphs and your explanation for each of the results. In total you should have at least 6 graphs and your explanations. Make your suggestion on the most efficient hyperparameters values for your problem setup.
- **Git Expert [2 points]**
  Along with your submission at UBlearns, upload your project on GitHub in a **private repository** and add **@ub-rl** as collaborators. Git is one the essential tools to know, you can check some foundations of it here.

2 Deliverables

There should two parts in your submission:

2.1 Report

The report should be delivered as a separate pdf file, and it is recommended for you to use the NIPS template as a report structure. You may include comments in the Jupyter Notebook, however you will need to duplicate the results in the separate pdf file. For the final submission, combine the reports for both Part 1 and Part 2 into one file.

2.2 Code

Python is the only code accepted for this project. You can submit the code in Jupyter Notebook or Python script. Ensure that your code follows a clear structure and contains comments for the main functions and some specific attributes related to your solution. You can submit multiple files, but they all need to have a clear name. After executing command python main.py in the first level directory or Jupyter Notebook, it should generate all the results and plots you used in your report and print them out in a clear manner. Additionally you can submit the trained parameters (.$h5$), so that the grader can fully replicate your results. For the final submission you can combine the code from both parts into one.

3 References

- NIPS Styles (docx, tex)
- Overleaf (LaTex based online document generator) - a free tool for creating professional reports
- GYM environments
- Lecture slides
- Richard S. Sutton and Andrew G. Barto, "Reinforcement learning: An introduction" (pdf)

4 Checkpoint Submission [Due date: Sep 19]

Complete Part 1 and submit the code and draft report. To submit your work, add your pdf, ipynb/python script to zip file $YOUR\_UBIT\_assignment1\_checkpoint.zip$ (e.g. $avereshc\_assignment1\_checkpoint.zip$ and upload it to UBlearns (Assignments section). Checkpoint will be evaluated after the final submission.

5 Final Submission [Due date: Oct 3]

Add your combined pdf and ipynb/python script for Part 1 and Part 2 to a zip file $YOUR\_UBIT\_assignment1.zip$ (e.g. $avereshc\_assignment1.zip$) and upload it to UBlearns (Assignments section). After the assignment is graded, you may be asked to demonstrate it to the instructor if your results or reasoning in your report are not clear enough.

6 Important Information

This assignment must be completed individually. 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.

7 Late Days Policy

You can use up to 5 late days throughout the course toward any assignments' checkpoint or final submission. You don't have to inform the instructor, as the late submission will be tracked in UBlearns.

8 Important Dates

September 19, Sun 11:59pm - Checkpoint is Due

September 26, Sun, 11:59pm - Early Bird submission (optional)

October 3, Sun, 11:59pm - Assignment 1 is Due