Reinforcement Learning 2025 (Reichman University)

Semester B - Midterm Assignment

A starter Colab notebook can be found here.

Objective

The goal of this assignment is to design, implement, and analyze **tabular reinforcement learning agents** in the `MiniGrid-Dynamic-Obstacles-6x6-v0` environment. While training a successful agent is important, the **primary objective** is to **communicate your findings** through a well-structured, insightful report that reflects your **understanding of the challenges**, **design decisions**, **limitations**, and **learning outcomes**.

Background

Gym and MiniGrid

Gym is a standard API for reinforcement learning environments. It allows agents to interact with environments via `step`, `reset`, `render`, and `sample` functions. MiniGrid is a library built on top of Gym that provides lightweight 2D grid-based environments, ideal for benchmarking RL agents.

MiniGrid environments feature discrete action spaces and partial observability. Observations are usually provided as a 7x7 egocentric view.

Dynamic Obstacles Environment

In the `MiniGrid-Dynamic-Obstacles-6x6-v0` environment, the agent is placed in a grid world that includes **one or more moving obstacles**. The agent must reach a fixed goal position while **avoiding collisions** with these moving objects. These obstacles follow **pre-defined movement patterns**, making the environment **temporally dynamic**. Detailed information is available in the notebook.

You are encouraged to explore **smaller variants** of this environment for debugging and experimentation, available here. Testing your agent in simpler settings can help identify bugs, refine preprocessing strategies, and evaluate early-stage learning behavior before scaling up to the full 6x6 environment. Available environments include:

- `MiniGrid-Dynamic-Obstacles-5x5-v0`
- `MiniGrid-Dynamic-Obstacles-Random-5x5-v0`
- `MiniGrid-Dynamic-Obstacles-6x6-v0`

Assignment Task

Your task is to implement and train **at least two tabular reinforcement learning algorithms** (e.g., Q-learning, SARSA) to solve this environment. In addition to implementation, your focus should be on analyzing and comparing your agents through experimentation and reporting.

You are encouraged to:

- Apply **preprocessing techniques** to simplify or reduce the observation space and analyze their impact (e.g., reduced state space size) or to capture a more complex dynamics.
- Use **reward shaping** techniques, if needed, and justify them.
- Evaluate your decisions using quantitative data.

Project Requirements

1. Theoretical Analysis

- Estimate and describe the theoretical size of your Q-table.
- Discuss the expected success rate and average episode length under a random policy.
- Provide a short theoretical comparison between the algorithms you selected.

2. Implementation

- Implement **at least two** tabular RL agents covered in the course.
- If you applied **preprocessing** or **reward shaping**, analyze their effect (e.g., Q-table size, learning speed).

3. Training Process

- Clearly document your hyperparameters (learning rate, discount factor, epsilon decay strategy, etc.).
- Include graphs showing the training performance.
- Discuss your tuning process and what insights you gained.

4. Performance and Evaluation

- Track key metrics: total reward per episode, number of steps, success rate, etc....
- Provide visualizations (e.g., moving averages, exploration curves, etc).
- Develop a reproducible **evaluation procedure**: run each trained policy for 100 episodes and report performance metrics (graphs + summary statistics).

5. Insights and Conclusions

- Write a detailed section in your report summarizing what you learned.
- Support your insights with data (e.g., comparisons between hyperparameter configs, exploration strategies, or algorithms).
- Include both successful and failed experiments, and attempt to explain the observed behaviors.

Report

You are required to submit a well-organized and concise report that presents your complete analysis. The **main body of the report must not exceed 4 pages of text**. This limitation is intentional: it is designed to encourage **focused, high-quality communication**, and to discourage the inclusion of excessive or redundant explanations that do not contribute meaningful insights.

Despite the text limit, the total report length (including figures) may be longer. You are encouraged to include as many **supporting graphs and visuals** as needed, but they must be placed **after** the main body of text.

Your report must include the following sections:

- **Introduction**
- **Environment Analysis**
- **Implementation Details**: Describe the agents, strategies used, preprocessing steps, and reward shaping decisions.
- **Hyperparameter Tuning**: Outline the tuning process and justify your final choices.
- **Results**: Present and summarize performance metrics and training curves.
- **Insights and Conclusions**: This is the most important section. Reflect on what you learned, compare different approaches, explain successes and failures, and back your analysis with quantitative data and references to figures.

> All figures must appear at the end of the report. Each figure must have a clear title, a descriptive caption, and a number. Every figure should be referenced explicitly in the main text.
>
> **Figures or any other material (text, image, code etc...) which is not explicitly referenced will not be considered during grading**.

Evaluation Notebook

In addition to your training notebook, submit a separate **evaluation notebook** that:

- Loads your trained policies (you may link to them externally or include them in your ZIP submission).
- Runs the evaluation procedure (100 episodes per policy).
- Displays metrics and outputs for each agent.
- Includes clear instructions for running the code.

Submission Guidelines

Submit the following via Moodle:

1. **Training Notebook**: A Jupyter/Colab notebook with full training process.
2. **Report**: A PDF named `report_ID1_ID2.pdf` (max 4 pages of text, unlimited figures).
3. **details.txt**: Names, IDs, and notebook links.
4. **explainer.txt**: Short guide for running your notebooks.
5. **Evaluation Notebook**: A runnable notebook with your evaluation pipeline.

Important Notes

- **Teams**: Work in pairs only.
- **Original Work**: You may use helper libraries already imported in the starter notebook, but **do not use prebuilt RL algorithms** from libraries like `stable-baselines3`.
- +5 **bonus points** for groups that also solve the `8x8` environment.

Good luck, and enjoy experimenting with dynamic environments!