# COMPSCI 687 - Final Project - Fall 2025

Due **December 9**, 11:55pm Eastern Time

## 1 Overview

- For the final project, you will conduct a study of your choice.
- This project should be completed in groups of **three** students.
- The goal of this project is to give you an opportunity to investigate a topic within the RL field that we did not cover in class.
- You should submit a single .zip file containing both a .pdf write-up along with your source code.
- The source code you submit must be composed of Python (.py) files, not Jupyter Notebook (.ipynb) files.
- We will run an automated system to detect plagiarism. Any code you submit will be compared against code submitted by your classmates, as well as with many implementations available online.
- The write-up should be roughly 5 to 10 pages and should describe what you did and what results you obtained.
- In your report, clearly describe which tasks were performed by each student.
- If you are unsure whether the project you have in mind is sufficient or appropriate, you should ask the instructor and/or the TAs.

## 2 Project Options

Below we list some example projects that represent roughly the minimum amount of work required to receive full credit. These should give you a sense of what we expect.

### Option 1.

Select a problem in your field that can be modeled as an MDP. Describe the problem, explain how it fits the MDP formulation (i.e., formally define all components of the corresponding Markov Decision Process), discuss the techniques that are *currently* used to solve it, and then apply **three existing** RL methods to the problem.

For this project, the main contribution will be the *new environment*, so you are **not** allowed to use any code from existing RL domains. You may, however, use existing RL algorithm implementations, such as those in OpenAI Gym or Stable Baselines3.

You should then:

(i) Report **how** you implemented the domain (e.g., your choice of programming language and details about how you implemented each component of the MDP);

(ii) Describe any assumptions you had to make in order to model the original problem as a Markov Decision Process; and

(iii) Report how well the RL algorithms performed, including how hyperparameters were optimized. When analyzing performance, you should present learning curves similar to those in the homework assignments.

We strongly encourage you to check with the instructor and/or the TAs if the project or problem you selected is too complex to complete within the available time frame, or too simple given the expectations for final projects in this course.

### Option 2.

Select **two** RL algorithms that were not covered in class and evaluate each of them on at least **two** existing MDPs/domains. Since the point of this project is to implement these methods from scratch, you should *not* use existing code for the RL algorithms—though you may use existing code for the environments that you test.

Examples of algorithms that we have not covered in class that could be a good fit are the REINFORCE with Baseline algorithm (Section 13.4 of the RL book), the One-Step Actor-Critic algorithm for episodic tasks (Section 13.5 of the RL book), a model-based RL algorithm such as Prioritized Sweeping (Section 8.4 of the RL book), the Monte Carlo Tree Search (MCTS) algorithm (Section 8.11 of the RL book), the Episodic Semi-Gradient $n$-step SARSA algorithm (Section 10.2 of the RL book), the True Online SARSA($\lambda$) algorithm (Section 12.7 of the RL book), and the PI$^2$-CMA-ES algorithm from the paper *Path Integral Policy Improvement with Covariance Matrix Adaptation*.

If you decide to implement an algorithm that is not listed above, please check with the instructor and/or TAs to make sure it is appropriate for this course project.

In the write-up for this type of project, you should include:

(i) A brief description of the methods and what they do;

(ii) Pseudocode for the methods;

(iii) A discussion of how you tuned their hyperparameters; and

(iv) The corresponding experimental results. When analyzing the performance of each algorithm, you should present learning curves similar to those in the homework assignments.

## Extra Credit

In addition to the tasks described above, there are a few ways you can earn extra credit. Below are some suggestions.

- The ideas described on the previous page suggest that you could either formulate and implement an MDP, *or* implement two new RL algorithms. One way to earn extra credit would be to do *both* of these tasks.
- In case you chose to implement new RL algorithms, you might be planning to evaluate them in the toy domains introduced in class: the 687-GridWorld domain, the Cat-vs-Monsters domain, and the Mountain Car domain. One way to earn extra credit would be to implement other classic RL benchmark problems, such as Acrobot, Inverted Pendulum Swing-Up, or the Cartpole domain.
- You may choose to implement a *third* RL algorithm, as long as it is not a trivial modification or extension of one of the other algorithms you are implementing.
- You might also evaluate how well Value Iteration performs when given an *estimated* (or learned) model. In particular, you could first let an agent explore its environment using different exploration strategies. While doing so, the agent would collect sample experiences and use them to estimate the transition and reward functions. Then, you could use this estimated model to compute a near-optimal policy using Value Iteration. The goal of this experiment would be to quantify how close to the optimal policy you can get, depending on the type of exploration used and the total time the agent is allowed to explore—that is, depending on the quality of the learned model of the environment.