COMPSCI 687 - Final Project - Fall 2024

Due **December 10, 2024**, 11:55pm Eastern Time

Overview:

- For the final project, you will conduct a study of your choice. This project should be completed in groups of two students.
- Since you will have about one month to complete it, this project should take approximately 2 to 3 times the amount of time and effort as a homework assignment—although you are free to do more.
- The goal of this project is to allow you to investigate something 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 for detecting plagiarism. Any code you submit will be compared to the code submitted by your colleagues, and also with many implementations available online.
- The write-up should be roughly 5-10 pages and should describe what you did and what results you obtained. If you are unsure whether the project that you are thinking of is sufficient, you may ask the instructor and/or the TAs. In your report, you should describe clearly which tasks were performed by each of the students.

Examples: Below we list some examples of projects that you could do that are roughly the minimum amount of work required to get full credit for the project, and which can give you an idea about what we might expect.

1. Select a problem in your field that can be modeled as an MDP. Describe the problem, how it fits the MDP formulation (i.e., define formally 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*, and so you are not allowed to use any code for existing RL domains. You may, however, use existing RL algorithm implementations, such as those in OpenAI Gym/Stable Baselines 3. You should then *(i)* report *how* you implemented the domain (e.g., your choice of programming language and details about how, exactly, you implemented each component of the MDP); *(ii)* any assumptions you had to make in order to model the original problem as a Markov Decision Process; and *(iii)* how well the RL algorithms worked, along with how hyper-parameters were optimized. When analyzing the performance of each algorithm, you should present learning curves such as the ones in the homework assignments. **We strongly encourage you to check with the TAs if the project/problem you selected is not too complex to be studied/completed within this time frame, nor too simple given the expectations we have about the final projects for this course.**
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(λ) 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 a different algorithm, please check with the instructor and/or TAs to make sure that the selected algorithm is appropriate for this course project. In the write-up for a project of this sort 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 hyper-parameters; and *(iv)* the corresponding experimental results. When analyzing the performance of each algorithm, you should present learning curves such as the ones in the homework assignments.

In addition to the tasks described above, there are a few ways in which you may earn extra credit. Here are a few suggestions:

- The ideas described in the previous page suggest that you could either formulate and implement an MDP, *or* implement two RL algorithms. One way of earning extra credits would be to perform *both* of these tasks.
- In case you chose to implement 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 Inverted Pendulum Swing-Up domain. One way of earning extra credits would be to implement other classic RL benchmark problems, such as Acrobot, Mountain Car, and the Cartpole domain.
- You may also implement a third RL algorithm, as long as it is not a trivial modification/extension of one of the other algorithms you are implementing.
- You might also evaluate how well Value Iteration would perform if given an *estimated/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—i.e., depending on the quality of the learned model of the environment.