CSE 4/546: Reinforcement Learning
Spring 2022

Instructor: Alina Vereshchaka

Final Course Project

Multiple deadlines, please see below

Description

The goal of the Final Course Project is to explore advanced methods and/or applications in RL. You will be expected to prepare a proposal, checkpoint, final submission, and presentation. All projects should evaluate novel ideas that pertain to deep RL or its applications and must involve RL algorithms.

You are encouraged to use your ongoing research work as a project in this course, provided that this work relates to deep RL. Discuss the topic of your final project with course instructors by private message in Piazza, or during OH. If you are not sure about the topic, we encourage you to speak with us.

There are few directions suggested, please check the end of the description for more details.

1. Register your team (March 27)

You may work individually or in a team of up to 3 people with both undergraduate or graduate students. The evaluation will be the same for a team of any size.

Register your team at UBLearns (**UBlearns > Tools > Groups**).

2. Project Management Tool Setup (April 1)

If you are working in a team of two or three people, set up a project management (PM) tool. Choose from Trello, JIRA, Basecamp

1. Create a board with at least these columns: **To do, In Progress, Done**
2. Divide the project to at least 20 steps/milestones (E.g. Explore the topics, Create a proposal, Implement a basic version, Implement advanced algo, Submit Checkpoint, Submit Final, Prepare for the presentation, etc).
3. Add course instructors as observers or as team members to your board [devikabi@buffalo.edu & nitinvis@buffalo.edu]
4. There should be some tracked activities every week
5. Include a few screenshots of your board activities as part of your Checkpoint and Final submissions.

3. Submit the proposal (April 3)

The project proposal should be a one page single-spaced extended abstract motivating and outlining the project you plan to complete. Submit at **UBLearns > Assignments.** You proposal should have the following structure:

1. Topic
2. Objective. Explain the objective of the project and why that objective is relevant and important.
3. Related Work. Briefly review the most relevant prior work, and highlight where those works fall short of meeting the objectives described above.
4. Technical Outline. Explain your approach at a high-level, making

clear the novel technical contribution. What environment and algorithm you are planning to use.

5. List any reference you are planning to use.

4. Submit the checkpoint (April 17)

- Submit the initial results of your model (e.g. built baseline model and prepared a pipeline for training).
- You need to submit the code and a draft report with prior results.
- All project files need to have clear naming, e.g. **avereshc_nitinvis_checkpoint_project.ipynb** and **avereshc_nitinvis_checkpoint_project.pdf**
- All project files should be packed in a ZIP file named: **TEAMMATE#1_UBIT_TEAMMATE#2_UBIT_ checkpoint _project.zip** (e.g. **avereshc_nitinvis_checkpoint_project.zip**).
- Submit at **UBLearns > Assignments**
- Include Academic Integrity part

5. Submit the final results (May 8)

- Submit at **UBLearns > Assignments**
- The code of your implementations should be written in Python. You can submit multiple files, but they all need to be labeled clearly.
- All project files need to have clear naming, e.g. **avereshc_nitinvis_final_project.ipynb** and **avereshc_nitinvis_final_project.pdf**
- All project files should be packed in a ZIP file named: **TEAMMATE#1_UBIT_TEAMMATE#2_UBIT_final_project.zip** (e.g. **avereshc_nitinvis_final_project.zip**).

- Your Jupyter notebook should be saved with the results.
- A report can be submitted in combination with the presentation. Thus if you discuss all the technical implementation of your project and provide the results, a separate report will not be necessary.
- Include all the references that have been used to complete the project.
- If you are working in a team, 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	Project Part	Contribution (%)

6. Present your work (May 10)

- Present your work during the presentation day. Registration slots will be available prior to dates.
- The whole team should equally present the work.
- Your presentation should represent the work you will submit.
- Submit the final presentation by **May 9, 11:59pm**.

Presentation details

Length: 15 mins + follow-up questions

Suggested Templates: UB branded templates or UB CSE template

Suggested presentation structure:

- Project Title / Team's Name / Course / Date [1 slide]
- Project Description [1 slide]
- Background [max 2 slides]

- Implementation [max 2 pages]
- Demo (if available)
- Results (Graphs & Any Visuals) [max 4 slides]
- Key Observations / Summary [1 slide]
- For Teams: Contribution Summary by Each Team Member [1 slide]
- Thank you Page [1 slide]

Extra Points [max +20 points]

Participate in Weekly Scrum Meetings [5 points]

Weekly Scrum (Monday 12pm-12:30pm) is a regular group meeting led by Nitin Kulkarni where at least one team member gives updates about the project progress. This is not evaluated and you can also consider these meetings as an opportunity to get feedback on your current results.

Your team has to take part at least 4 times to be eligible for the bonus. We encourage all team members to join scrum meetings.

Real-world RL application [5 points]

If you are formulating and solving a real-world problem using RL methods, your work is eligible for this bonus. In your report you must highlight the references to the real-world case that you are solving as well as the real-world data you use for your project.

CSE Demo Days [10 points]

If you get interesting results, we would encourage you to share your project with the public in terms of participating in the CSE Demo Days. CSE Demo Days is a semester event, where you can highlight your project results.

Send us your prior results before May 6. **Selected teams** will have to prepare a poster and present it. No requests to take part in CSE Demo

Days will be accepted after May 6.

If you receive a winning place award for your project, **your bonus points will be +50!**

Important Information

This project can be done in a team of up to three people.

- All team members are responsible for the project files submission
- No collaboration, cheating, and plagiarism is allowed in assignments, quizzes, the midterms or final project.
- All the submissions will be checked using SafeAssign as well as other tools. SafeAssign is based on the submitted works for the past semesters as well the current submissions. We can see all the sources, so you don't need to worry if there is a high similarity with your Checkpoint submission.
- The submissions should include all the references. Kindly note that referencing the source does not mean you can copy/paste it fully and submit as your original work. Updating the hyperparameters or modifying the existing code is a subject to plagiarism. Your work has to be original. If you have any doubts, send a private post on piazza to confirm.
- Submitting a material that has been previously submitted, in whole or in any part is not allowed.
- All group members and parties involved in any suspicious cases will be officially reported using the Academic Dishonesty Report form. What does that mean?
    - In most cases, the grade for the assignment/quiz/final project/midterm will be 0 and all bonus points will be subject to removal from the final evaluation for all students involved.
    - Those found violating academic integrity more than once throughout their program will receive an immediate F in the course.

- In order to certify that you followed Academic Integrity policy while completing the assignment, please include the following statement as a comment block at the beginning of your code. In case you submit multiple files, add this statement at the beginning of each code file submitted:

"I/we certify that the code and data in this assignment were generated independently, using only the tools and resources defined in the course and that I/we did not receive any external help, coaching or contributions during the production of this work."

Please refer to the Academic Integrity Policy for more details.

Late Submission Policy

For the final project everyone will be provided with 3 late days per team, no matter whether you are working individually or with teammates. These late days can be applied only to final project-related due dates. Be aware that some of the project components have hard deadlines.

These cannot be combined with your individual late days.

You do not have to inform the instructor, as the late submission will be tracked in UBlearns.

Important Dates

March 27, Sun, 11:59pm -- Register your team (UBLearns > Group) and setup PM tool
April 3, Sun, 11:59pm -- Abstract is Due
April 17, Sun, 11:59pm -- Checkpoint is Due
May 8, Sun, 11:59pm -- Final Submission Deadline
May 10, Tue -- Presentation **[hard deadline, no late days can be applied]**

Final Project Directions

Below is a list of possible directions for your final project, choose one.

MULTI-AGENT RL

Build and solve multi-agent tasks, including but not limited to agents communications, transportation problems, multi-agent cooperation, etc. It might potentially lead to a research project.

STEPS INCLUDE:

1. Build a small multi-agent environment with at least two agents (this can be an extension of your work at Assignment 1).
2. Solve the environment using any tabular methods.

   [Steps 1 & 2 can be used for checkpoint submission]

3. Solve the environment using any deep RL methods (DQN, DDQN, AC, A2C, DDPG, TRPO, PPO, etc) and compare the results.
4. Apply the algorithm from Part 3 to solve any of the existing MARL problems. E.g. Open AI Particle Environment, Hide&Seek

References:

- A collection of papers/books/etc [github]
- Multi-Agent Particle Environment (often used for baseline) [github]
- A collection of well-defined MA Gym-like env (good reference) [github]
- Hide and Seek [blog post, github]

Suggestions:

1. Go with collaborative multi-agent systems, it is easier to scale and has more real-world reference. Possibly you can think about a mixed cooperative-competitive, e.g. Hide and Seek example.

2. While building your env, think about some real world problem that you can solve using MARL settings. It is good if the problem you formulate is a socially important one (e.g. mitigating natural disasters, transportation, rescuing people, etc), so it can be later converted to the research project.

APPLIED RL

Consider a real-world problem that can be solved using RL methods. Deploy the model and show the results.

POSSIBLE DIRECTIONS:

- RL for A/B testing [video]
- Real-time time-series prediction
- News recommendation [paper]
- Marketing -- real-time bidding with multi-agent RL [paper, paper]
- Dynamic medical treatment regimes
- Many others

Note:

For this direction you might need to build a web app or construct an environment that is based on real-world data. Deploy the RL model and show the results real-time.

RC CARS

Set up the simulator and train the cars in the simulator. These steps may require prior knowledge in robotics or autonomous vehicles. Since the simulator is based on Amazon AWS, you might need to set up a local host to train the models.

STEPS INCLUDE:

1. Install & explore the DeepRacer RC cars simulator
2. Complete the Udacity DeepRacer course
3. Check existing solutions & current rewards functions

   [Steps 1 - 3 can be used for checkpoint submission]

4. Apply other RL methods to teach the car to drive in the simulator

References:

- Udacity AWS DeepRacer Course [free]
- How to run DeepRacer locally [medium]

PARALLEL & DISTRIBUTED RL

Parallel processing is a good skill that can be applied not just to RL problems, but also to many other applications. During the course we will cover a few algorithms that introduce parallel training (e.g. A3C).

For the final project you are expected to implement an already existing algorithm and apply the parallel training to some deep RL algorithms and compare the efficiency.

EXPLORING DEEP RL ALGORITHMS [SAFE MODE]

Research and apply recent advances in RL. This may include solving any of the following environments using deep RL algorithms.

Possible environments include:

- Robotics by OpenAI [robotics, blog post]

- MuJoCo by DeepMind [DeepMind article, documentation]
- Atari by OpenAI
- PyBullet [details] - a well-supported env for robotics simulation

Steps include:

1. Set up the environment
2. Explore the existing baseline methods applied to solve it & Run it

   [Steps 1 & 2 can be used for checkpoint submission]

3. Apply a few other deep RL to improve the results. You can use your Assignments 2 & 3 implementations as a baseline code for comparison.

Some advanced directions to consider:

1. Random Network Distillation (RND) [blog post, paper]
2. Offline reinforcement learning [blog post, tutorial]

Propose you own topic

You may come to us with your topic proposal! We understand that it might end up to be pretty challenging, if you find out you are completely stacked, you are welcome to discuss your possible switching to any other directions.

Although you will follow your own direction, the general requirements for the first checkpoint include the following:

1. Build your own environment or use an existing one.
2. Explore the already existing solutions and replicate it, you will later use it as a baseline to compare your solution.
3. Propose your solution to solve the problem and compare the results

Please talk to the course instructors to ensure the project you have in mind is feasible.

GENERAL STRATEGY

- The main motivation of the final project is to explore novel ideas (either with the problem setup or the algorithm or both) or have a comparison of existing solutions. Start with a simpler problem and build on it going forward.
- If you have a challenging environment to solve, e.g. MARL, MUJOCO or RLBench, or you create you own, you can choose easy algorithms to work with such as Double DQN/ Advanced Actor-Critic etc, as there is considerable effort in designing and setting up the env
- If you choose to explore Deep RL algorithms, consider applying those algorithms on multiple env to have a better comparison and analysis, e.g. Atari Breakout, SpaceInvadors and FetchReach.
- You are encouraged to use your implementations in A1, A2 or A3 (Tabular Methods/DQN/Double DQN/Actor-Critic) as a baseline to compare against other approaches that you will use for your project.