Term Project
CS4100: Introduction to Artificial Intelligence Spring 2022

1 Introduction

The project offers an opportunity to apply concepts/techniques learned in class on a substantial problem that interests students. Whereas most of the course has been focused on breadth of topics, the project provides a counterpoint by requiring students to study a problem in depth.

1.1 Topics

Your final project can be on anything that has some relation to AI. It should relate to at least one of the four modules we cover in our class (search, MDPs / RL, Bayesian inference, machine learning). It is likely that the project you choose only relates to one module – this is completely fine and expected. We suggest many topics in a later section of this document.

1.2 Types of Projects

This semester, we offer two types of projects:

- **Practical**: Find and formulate a problem, develop/implement/apply algorithms to solve it. Recommended if you want to escape the toy domains we have studied so far and actually build something substantial and potentially applicable in the real world. You would likely learn how to use new AI-related software tools and hone algorithmic / data-scientific skills. If successful, the project would be a great addition to your portfolio. Also a great chance to do the AI-related hobby project that you have always dreamed of but never got started on.
- **Theoretical**: Do an independent study of an advanced topic not covered in the course. Recommended if you have enjoyed the topics so far and just want to see more of it, or go significantly deeper into something we briefly touched on. You will learn about new problems / models / algorithms. This is recommended if you plan to take more advanced AI courses in the future (e.g., machine learning), and/or if you are interested in getting involved in AI-related research. Could also be used to deeply study / reproduce a research paper

1.3 Expectations

- Practical project: We do not expect everyone to achieve state-of-the-art results on the problem they choose – although that would be a pleasant surprise if it happened. Your team is expected to find a problem of interest, formulate it well and precisely, develop/apply an algorithm to solve the problem, analyze your results, and communicate that to the class (and us).
- Theoretical project: You should learn the proposed topic and understand it well enough to the point that you can potentially teach an hour-long lecture on it. We will not actually ask you to deliver this lecture. Instead, you will learn and digest the chosen topic, and produce a tutorial on it, such as in the

form of a report, slides, illustrative examples, or other pedagogical materials. There is more flexibility on the deliverables.

1.4 Timeline

All deadlines, except for the project presentation, are at 11:59 PM EST/EDT. Submissions should be uploaded on Canvas. For deadlines, see the syllabus on Canvas.

1.5 Teams

Teams should have 1–3 members. Our recommendation:

- Practical project: We strongly recommend forming teams of 2, which in past experience has led to much more interesting projects. Overall, we expect each team member to spend about 30-40 hours on the project (about 6-8 hours per week for five weeks). This is why teams of two are recommended – you can accomplish much more with 70 total hours compared to 35. If your team has three members, that needs to be reflected in the project scope. Projects by three-person teams that do not have sufficient scope will receive a lower grade than they would if the same project were submitted by a two-person team.
- Theoretical project: Teams of 1 or 2 are allowed. If in a team of 2, the chosen topic should be broader, and there should be a rationale as to why 2 people are needed. Three member teams are not allowed for a theoretical project.

Deviating from the above recommendations will require a written rationale in the proposal.

1.6 Grading

The project as a whole is worth 20% of your grade. That breaks down as follows: project proposal (2%), milestone report (1%) project presentation (2%), final project report (15%).

1.7 Typical practical project process

Typical project process (practical track) Overall, we expect each team member to spend about 30-40 hours on the project (about 6-8 hours per week).

1. ($\sim$ 5 hours) Decide on a project topic, do some background reading, and make an initial problem formulation. Make a rough plan for the project itself (this is the proposal).
2. ($\sim$ 10 hours) Learn the additional knowledge necessary to undertake the project. The amount of time this takes may vary depending on your chosen topic. It is very likely that you will need to learn at least one or two new concepts/algorithms for your project.
3. ($\sim$ 10 hours) Figure out the simplest way to solve your problem (i.e., a basic algorithm), and implement it / use an existing implementation. Apply the simple algorithm to your problem.
4. ($\sim$ 5 hours) There are usually two results of step 3: Either you have completely solved the problem you chose, or it doesn't work too well (at least on some cases). If the former, analyze why it did so well, and come up with some interesting extensions. If the latter, analyze what went wrong, and figure out how to address that.
5. (Remaining time: $\sim$10-20 hours?) Iterate steps 3-4, each time using the analysis in step 4 to identify some potential area to improve on, and try again (step 3).
6. Summarize the overall process: the problem, algorithm, results, and insightful analyses. Present your story to the class / in your final report.

1.8 Typical theoretical project process

This is more fluid and harder to predict. It depends on the chosen topic and its scope.

1. ($\sim$ 10-20 hours) Read the textbook and/or other tutorial material, possibly including watching recorded lectures, to gain a broad understanding of the topic and its context.
2. ($\sim$ 10-20 hours) Work on problems / code simple examples and experiment with variations, to solidify and deepen the initial understanding, and to sort out confusions and misconceptions.
3. ($\sim$ 10-20 hours) Distill your understanding into project deliverables (such as in the form of a report, slides, illustrative examples, or other pedagogical materials).

1.9 Feedback on project proposal and milestone reports

We will provide feedback to teams on their project proposals and milestone reports via Canvas. If we are happy with the submitted proposal or milestone report, then this feedback may be as simple as "good". If we feel something should be changed or expanded upon, we will note that in the feedback. If, after receiving feedback on your project, you would like to discuss it with us, we will schedule a Zoom meeting with one of the TAs.

2 Potential Topics

This section provides some general topic categories and specific suggestions, but you are welcome to choose anything else that interests you.

2.1 Topic categories

- (Practical) Extend a programming assignment (PacMan) in much greater depth. For example, you can try to apply and analyze even better heuristics / search algorithms (potentially learning parts of the heuristic / algorithm), learn better evaluation functions (learn the features and weights), or try out better reinforcement learning algorithms (potentially with learned features and weights for function approximation).
- (Practical) Find some interesting problem / dataset on Kaggle[1] or some other competition platform and tackle it. This is usually a machine learning problem. If you do well, you might be able to enter/win a competition!
- (Practical) OpenAI Gym[2] – This platform from OpenAI contains many interesting environments for training reinforcement learning (RL) algorithms. It is quite popular for RL enthusiasts and is often used to benchmark RL algorithms. DeepMind Lab is a another such platform, but the environment may be more challenging (first-person 3-D) – and potentially more interesting. There is an increasing number of similar platforms.
- (Theoretical) Learn extensively about an problem / model / algorithm that is just beyond the scope of the course. Develop / apply and convey the understanding on illustrative examples. Produce pedagogical materials about the topic. See below for some suggested specific topics.
- (Theoretical) Find a paper that tackles an interesting problem, and try to re-implement it (initially without referring to existing code, if any is available). This can be quite challenging because you may find that some crucial details (e.g., particular settings of constants or hyperparameters) may not be present in the paper!

[1] `https://www.kaggle.com/`
[2] `https://gym.openai.com/`

- (Theoretical) If you have ideas about how to substantially improve and liven pedagogical materials used in current class topics, and want to improve the learning experience of future students, you can implement them in a project. Substantial improvement means the introduction of illustrative examples, visualizations, interactive demonstrations, or other engaging material. This will require frequent discussion with Prof. Holtzen or the TAs.

Potential problems / models / algorithms (theoretical track)

- Theoretical correctness / convergence / optimality proofs of algorithms covered in class (e.g., A*, value iteration, policy iteration, Q-learning, etc.)
- Alpha-beta pruning and other search-pruning strategies (AIMA Ch. 5.3)
- Monte-Carlo tree search (AIMA 6.4)
- Local search (AIMA Ch. 4, Ch. 6.4)
- Logical reasoning (AIMA Section III)
- Classical planning (AIMA Ch. 10-11)
- Policy iteration and other MDP solution methods (AIMA Ch. 16.3)
- Partially observable Markov decision processes (POMDPs, AIMA Ch. 16.4)
- Game-theoretic sequential decision making (AIMA Ch. 16.5-16.6)
- TD(λ) (Sutton and Barto Ch. 12)
- Policy search / policy gradient (Ch. 21.5, Sutton and Barto Ch. 13)
- Kalman filters / dynamic Bayesian networks (AIMA Ch. 14.4-14.6)
- Approximate inference algorithm, e.g., Markov-chain Monte Carlo (MCMC) (AIMA Ch. 14.5)
- Learning Bayesian networks / expectation maximization (AIMA Ch. 21)
- Variations on stochastic gradient descent algorithms
- Your favorite machine learning problem / algorithm (see recent offerings of CS 6140)

As you can see, many of the above suggestions are actually covered in AIMA, but we do not have time to cover them. Flip through the textbook and see what interests you. If you choose such a problem / algorithm though, expect to read further beyond the content in the textbook – you will most likely need more depth than the textbook covers.

2.2 Some suggestions

Random specific suggestions These are just some random problems that came up in daily life that seem solvable with a little bit of AI:

- Paper / news recommendation. Many papers these days are posted on arXiv, but there are too many to browse through. Build a personalized recommendation system that can eventually take daily digests of new papers and output a ranked list of suggestions. Analogously, you can do the same for general news stories.

- Correcting bus predictions / providing uncertainty estimates. NextBus predictions report when a bus will arrive at a certain stop, but these predictions are often inaccurate. Some part is due to random noise (for which an uncertainty estimate is appreciated), and some part is simply bias (at some stops the prediction jumps by a minute). Learn to provide more accurate predictions. (There used to be a website that corrected MBTA #1 bus predictions, but we cannot find it any more.)
- Optimize elevator scheduling. This is actually one of the first successful practical applications of reinforcement learning ("Improving elevator performance using reinforcement learning", NIPS 1996).
- Optimize something that comes up in your daily life. (This is the source of the above suggestions...)

2.3 Projects that are not recommended

- Apply an existing algorithm to an existing dataset and report the results. This is not enough – you need to analyze the results, see what was good and what could be improved, and iterate.
- Projects on unrelated topics – if you are unsure, ask us!
- Projects that are too broad. 30-40 hours (or 60-80) is not a lot of time. Start small (very small), and if you succeed early, extend and iterate from there. If there is an interesting problem that is likely to take too much time (this is true for many interesting problems), identify the first step in the problem and make that your project.

2.4 Implementation

- Your project does not have to have an implementation component – for example, you can do a theoretical project. However, we expect most projects will have some implementation component, even if it is just on illustrative toy domains (e.g., if you are investigating a new algorithm). If implementation is required, you can use whatever programming language you like. This choice may be dependent on any existing code/libraries that you build upon.
- If there is an existing implementation, you can use it (with proper acknowledgment), but you do not have to. It depends on what you want to get out of the project. If you want to extend an existing method, then you will probably save time by building on existing implementations (especially if this is some non-AI related infrastructure, e.g., a game engine). If you want to have the experience of writing from scratch and deeply understanding the existing approach, you are welcome to do a re-implementation (although you should not refer to the existing "answer" in this case in your initial attempt). If you choose to re-implement, you may find that you cannot replicate existing performance, at which point you can compare against the existing implementation and see what went wrong – is it a bug on your end, or are there some "magic numbers" that make the algorithm work? Analyzing these hidden / unexpected bits can make a very interesting project.

3 Project Proposal

Each team submits one proposal. The project proposal is a short document that organizes your team's intentions and communicates that to the course staff, such that we can provide appropriate guidance. It is not a contract – we expect that projects may change course after you start working on them. There is no specific page limit, but we expect that a 1-2 pages (12pt single spacing) should suffice. Just make sure to include all the following items:

- Who is on the team? If you are working in a team of two, is there a clear division of labor? If team is smaller/larger than recommended, provide a rationale.

- Describe the problem you are trying to address, and provide a formulation of it. If it involves an algorithm (i.e., any implementation-based project), describe the input to the algorithm and the desired output.
- What is the ideal outcome of the project? What do you expect to show?
- What algorithms do you expect to use?
- What topics / libraries / platforms, if any, will you have to learn in order to undertake your project? Provide references where applicable.
- If your project has a machine learning component, it will most likely involve a dataset. Where will this come from? Is it an existing dataset, or are you making a new one? If the latter, do you have the resources to do so?
- Define milestones for your project. Milestones can include learning about certain topics / algorithms, acquiring and processing datasets, implementing an algorithm, analyzing results, etc. Again, we will not penalize you if you do not achieve your specific milestone. We will request a short progress report that addresses progress toward the milestones, and if they were not achieved, what turned out to be more challenging than expected. You may also find that your initial milestones were inappropriate, and you can choose different milestones, work toward those, and report on them. That is completely fine too – the point is to decompose your project into smaller pieces and ensure that you are making consistent progress over the next few weeks.
- Provide a week-by-week plan for your project.

3.1 Theoretical track

The proposal should be similar to the above, but certain items above may not be relevant (e.g., problem formulation, libraries/platforms, dataset). Instead, provide an initial list of learning material you will study to understand the topic, and provide a list of proposed deliverables that you would like to develop (such as in the form of a report, slides, illustrative examples, or other pedagogical materials).

4 Project Milestones

As previously mentioned, the proposal and the proposed milestones are not treated as contracts, but rather as your best guess of how things will turn out. Therefore, you will not be penalized if you fail to achieve your proposed milestones. More concretely, we expect milestone reports (one report per team) to address the following:

1. Report: What have you done so far? What is the current state of the project? To ensure best use of time, we recommend treating this question as a partial draft of the final report – ideally, if your project does not change, you can write 1-2 pages for this section in each milestone report, and then copy and paste them into the final report.
2. Reflect: Did you achieve your milestone(s)? If not, why not (e.g., some part was challenging because of [insert challenge here] and hence took longer, some other task became irrelevant)? Did you do something extra that you did not mention in your proposed milestone(s)?
3. Replan: Based on your progress so far and the knowledge you have acquired, what is the upcoming plan (week-by-week)? What is your immediate next step? If your project milestones / goals have changed from the initial proposal (e.g., you underestimated / overestimated the difficulty of the project), propose new milestones / goals.

Good project progress will roughly follow this timeline (practical track):

- By milestone 1: Have a precise problem formulation (inputs and outputs). Learned the basics of any additional topics necessary to accomplish project. Collected all the necessary external pieces (e.g., cleaned dataset doing machine learning, simulator / model if doing planning or reinforcement learning, external libraries if needed, etc.). High-level description / pseudocode of first algorithm to try, if you haven't tried anything yet; or describe which algorithms you have already tried.
- By milestone 2: Implemented and experimented with at least one algorithm / model. If some algorithm is complete, perform some empirical analysis and identify potential areas of improvement. Also, determine an "end-game" for the project (i.e., a plan for wrapping up within 1-2 weeks).
- By presentation: Current plan is for each team to present for 5 minutes. You should be able to articulate a complete project cycle: What is the high-level problem you wanted to tackle, your precise (technical) problem formulation, some method / algorithm you used (including external libraries / datasets if any), the empirical results you obtained, some analysis about why the described method worked well / not well in your experiments, some future direction to address what did not work well, possibly propose some interesting extensions, and reflection on the project as a whole (e.g., surprises, challenges, etc.). In the presentation, do not try to include everything (there is no time!) – just show the highlights.
- By final report (4/26): An extensive written version of the items described above for the presentation (e.g., describe all algorithms you tried, experimental methodology, other analyses you performed but did not show). The report should be relatively complete.

5 Presentations

The presentation should be 5 minutes long, with an additional 1-2 minutes for questions. Ideally, all members of each team will present. There are some cases where inevitably only one member is available, which is fine; but if are all present, you should decide how to split up the presentation. Suggested presentation structure

- (1 min) Describe the problem on a high level. What are you trying to do? When is the input and the output?
- (1.5 mins) Provide a technical problem formulation. This does not have to be mathematical, although it can be. For example, if you have a search problem, this section should explain what you state/action spaces are, and provide the successor function, goal test, edge cost, etc. If you have a supervised learning problem, this section should explain what features you are using, what form of output are you trying to predict, what the dataset looks like, etc. Basically, at the end of this section, we should start to see how we might obtain a solution for your problem (e.g., by applying an algorithm from class that works for the type of problem you have).
- (1 min) Describe an algorithm you chose to apply to solve your problem. If you have tried more than one, you can tell us more, or you may want to just focus on the most interesting one.
- (1 min) Explain results obtained from this algorithm, ideally compared against some baseline. Did the algorithm do well? If so, what made it work? If not, why not (and what should be the next thing to try)? Remember to take the time to explain what the experimental setup is, and explain what empirical numbers / graph axes mean. You may have been staring at the same numbers/plots for days, but we are seeing it for the first time!
- (0.5 min) Optional: Future directions / some high-level thoughts about your project so far / conclusion.

The above structure is only a suggestion. You can choose to present in a different way. Ultimately, try to tell an interesting story. If the algorithm is not too interesting, but you have interesting empirical results and insights, focus your time on the latter. Negative results (i.e., we could not get it to work) are completely acceptable too! Why did things not work – was it the algorithm, the features, the data, etc.? What surprised or frustrated you? Did certain aspects of the project take much longer than expected? Be assured that if your team does not have results yet, your team is not the only one. Your classmates (and we) will greatly appreciate whatever insights you share, including how difficult it is to get something to work. As you can see from the above, we do not expect project to be complete by the presentation. Tell an interesting story, hear about others's stories, and have a good time!

6 Final Report

6 Final Report There is no specific format that the project report should follow. You should choose a structure that best tells the story of your project. Most reports will probably have these components:

- Describe the problem on a high level, including some motivation for choosing it (just like in your presentation).
- Concrete technical problem statement, model (if applicable), inputs/outputs.
- Describe the dataset you used, if applicable.
- Methods / algorithms that you used or developed. If you are using algorithms not covered in the class, provide a description of it so we know you understand what you have been using.
- If you are basing your project on someone else's work, explain on a coarse level what they have done, and if you are doing anything different.
- Empirical results, including details on the experimental setup (be precise about what settings / data you ran experiments with). What were your hypotheses / what do you expect to see from your experiments?
- Analysis of empirical results. What worked, what did not, and why?
- Discussion / future directions (if any). What difficulties did you encounter, if any? Did anything not go according to plan? If you had more time to spend on the project, what would you have liked to do next? What advice about the project would you give to future CS 4100 students?

There is no set length / page limit for the report. It should take no more than 8 pages (including figures) to include all of the above. You can write as much as you wish, but do not be excessive – just get to the point and provide a complete description of the project, including the components described above.