

CS 182: Artificial Intelligence

Final Projects

1 Important Deadlines

The final project will consist of the following four aspects:

1. Project Proposal
2. Status Update
3. Final Paper
4. Presentation, either:
 - Oral presentation in class
 - Poster presentation at computer science fair

No late days for any course project deadline. If more students prefer oral presentations than can be accommodated during the last lecture period, talk slots will be filled by lottery or we will consider adding a second presentation day on 12/5 during the normal lecture time.

These deliverables will be due at the following deadlines:

| Aspect | Deadline |
|----------------------|--------------------------|
| Project Proposal | 11/2, 5:00 PM |
| Status Update | 11/22, 5:00 PM |
| Oral Presentations | 11/30, During Lecture |
| Posters to Printer | Before 12/4 !! |
| Poster Presentations | 12/6, 9:00 AM - 12:30 PM |
| Final Project Report | 12/8, 5:00 PM |

2 Goals and Scope

The CS 182 final project provides an opportunity for you to apply and extend the foundational concepts you have learned in the course. It is intended to encourage you to integrate ideas from different course components and allow you to delve deeper into areas of interest. Projects may be implementation, testing, and analysis of algorithms mentioned in lecture or described in the

text but not covered in assignments or design and implementation of an intelligent system that combines algorithms and representations from several course components (e.g., a more complex game-playing program or an intelligent advising system). Pick something that interests you. Students are expected to design and carry out final projects working in teams of 2-3 students. Note that we expect all students to demonstrate a roughly equal amount of work, so teams of 3 should be sure to tackle appropriately sized problems. You can use the Piazza forum to find partners. **Special Note:** If you want to undertake a theoretical project or work alone, please talk to Scott or one of the teaching fellows before beginning your proposal.

A list of possible course projects has been given below. Their content and scope are meant to be suggestive, not definitive. The teaching staff would be delighted to talk with you about possibilities. After the proposal, each final group will be assigned a course mentor who will be the best point of contact throughout the project.

Final projects that fall outside the specifics but within the spirit of these suggestions are encouraged, as are projects that draw on other computer science courses you have taken. It is, however, essential that the project make clear connections with topics covered in the course. If the idea you have for a project varies significantly from the suggestions, or you are unsure of its suitability, please contact the teaching staff enough in advance of the project proposal due date that our discussions with you can help shape your proposal. Some places beyond the text and course readings to look for inspiration: Proceedings of the major AI conferences (AAAI, IJCAI, ECAI, ICAPS, AAMAS, HCOMP, UAI) and journals (Artificial Intelligence, JAIR: J. of AI Research, JAAMAS: J. of Autonomous Agents and Multi-Agent Systems). Most CS papers are available free online either through the ArXiv or through search on resources like Google Scholar.

If your project involves building a system that includes designing a GUI or other significant software components, make sure that you focus most of your effort on the AI aspects of the project. We will evaluate your project on the concepts it investigates and the results and on how well it demonstrates your comprehension of the concepts, techniques and issues we have covered in the class. The project grade will incorporate evaluation of the proposal, presentation (oral or poster) and final paper quality. As there is no final exam for CS 182, your final project is the major integrative element of coursework.

3 Deliverables

3.1 Proposal

To ensure that you choose an appropriate project, you are required to turn in a 1–2 page project proposal by the date indicated above. The proposal should begin with a clear, unambiguous statement of your topic, and include all of the following:

1. a brief discussion of the problem and algorithms you intend to investigate and the system you intend to build in doing so,
2. identification of specific related course topics (e.g. heuristic search, MDPs, CSPs, etc.).
3. examples of expected behavior of the system or the types of problems the algorithms you investigate are intended to handle,
4. the issues you expect to focus on,

5. and a list of papers or other resources you intend to use inform your project effort. This list will form the core of your project report reference list. If your project includes anything unusual (such as having significant systems demands), please state this as well.

This document will be worth 5% of your grade for the final project. Use it to demonstrate to us that you have completed some background work on your chosen topic. The proposal should resemble a CS 182 problem set in the sense that if one were to give your proposal to another student, s/he should be able to complete the project as well. Proposals should identify all members of the team and indicate how you intend to divide work on the project. We will review your proposal and return it to you with comments and suggested modifications.

3.2 Update

To ensure that you are on track with the project and to identify any issues on time, you are required to submit a short (1-page) report with a status update. The report should describe the problem you are working on, the progress you've made so far, and any problems you came across that you would like to get help with. With the status update each group should also identify whether they would prefer to give an oral presentation or present at the poster session.

3.3 Presentation

The oral presentations will be done in class and the poster session will be at the CS 182 poster session on the date indicated above. These presentations are a chance to explain your problem and approach, showcase what you've accomplished, and get advice on surmounting any hurdles you've encountered. Students are expected to attend both presentation sessions, as they provide an opportunity for you to learn from each other.

- Oral presentations will be allocated 8-10 minutes, and should be focused on key issues. You are encouraged to bring less than 4 slides, because often just one diagram or chart can explain the essence of your idea and save lots of presentation time. You need not prepare fancy graphics; just come prepared to explain your topic and share what you have discovered.
- Posters will be printed through a SEAS allocated fund for student poster printing. A good rule of thumb is to design a poster to act similarly to a presentation as a tool for walking readers through your work. Along with the poster each member of your group should be able to explain the project to a general audience at the fair as well as the TFs and other students in the class.

3.4 Report

You must submit a written report on your project and the complete, well-documented source code for it. The report should be 5-10 pages in length. It should describe the algorithms you implemented and the data on which they were tested and include an analysis of results or system performance (depending on the scope of the project). We recommend [GitHub](#) as a method of collaborating and submitting source code. Reports must be formatted using LaTeX. We strongly recommend using the provided template on Canvas to structure your report. We recommend [GitHub](#), [ShareLatex](#), and/or [Overleaf](#) for collaboration on the report.

The report must contain all of the following content:

- A description of the purpose, goals, and scope of your system or empirical investigation. You should include references to papers you read on which your project and any algorithms you used are based. Include a discussion of whether you adapted a published algorithm or devised a new one, the range of problems and issues you addressed, and the relation of these problems and issues to the techniques and ideas covered in the course.
- A clear specification of the algorithm(s) you used and a description of the main data structures in the implementation. Include a discussion of any details of the algorithm that were not in the published paper(s) that formed the basis of your implementation. A reader should be able to reconstruct and verify your work from reading your paper.
- Analysis, evaluation, and critique of the algorithm and your implementation. Include a description of the testing data you used and a discussion of examples that illustrate major features of your system. Testing is a critical part of system construction, and the scope of your testing will be an important component in our evaluation. Discuss what you learned from the implementation.
- For algorithm-comparison projects: a section reporting empirical comparison results preferably presented graphically.

In addition, the report should include two appendices:

- Appendix 1 – A clear description of how to use your system and how to generate the output you discussed in the write-up. *The teaching staff must be able to run your system.*
- Appendix 2 – A list of each project participant and that participant's contributions to the project. If the division of work varies significantly from the project proposal, provide a brief explanation.

Your code should be clearly documented. Submit your code along with the project document to Canvas (or link directly to a public repository in your report). This link should appear in Appendix 2. If you have any questions about these specifications, please ask the teaching staff.

4 Suggestive Project Ideas

Below is a list of ideas that is intended to inspire your own project ideas, not define them!

4.1 TF ideas from 2017

1. (Annie) Implementing a genetic algorithm for the knapsack problem that would help HR of a company recruit for the best candidates considering their budget (capacity) for hiring and the skillsets (value) they are looking for in a candidate (item). Useful links: [1](#) and [2](#).
2. (Annie) Can you formulate a CSP-like optimization problem that deals with soft and hard constraints; an example of this would be restaurant recommendations for people with hard constraints like dietary restrictions and soft constraints like personal preference for food. Useful links: [1](#) and [2](#).

3. (Brian) As discussed in class RRT doesn't work well with robots that have complex dynamics. Could you train a NN to learn a good distance function? You could pass it in a bunch of valid trajectory optimization solutions (we can help you generate those) and then try to learn a good metric to use for a given system. [RRTs, NN, RL, Trajectory Optimization, Robotic Planning]
4. (Brian) Taking CS classes to satisfy the concentration requirements can often be tricky. Can you pose this problem as a CSP and find all solutions. Can you allow users to specify classes they definitely want to take and solve solutions given those new constraints. Can you allow users to specify values for how much they like certain classes and compute a valid course plan while trying to optimize the user's utility? [CSPs]
5. (Brian) There is a lot of randomness out there in the world. Can you incorporate robustness to perturbations in RRT, in RRT*? Can we compute an optimal, and robust trajectory? Aka can we find a safe path even if we get hit with a big gust of wind? Can we model this using the probabilistic techniques from this calls? [RRTs, Robust Algorithms, MDP, Bayes Net]
6. (Brian) Building off the previous 2 ideas can we design an algorithm to solve a CSP that is robust to classes suddenly not being offered? Therefore, can we return a safer course selection to make sure you don't get screwed when classes are suddenly not offered. [CSPs, Robust Algorithms, MDP, Bayes Net]
7. (Brian) Spam filters are so common now that most of you probably haven't had to deal with tons of spam emails ever. That said, they used to be a huge problem. Could you use learning to identify spam and get rid of it? [NN, RL]
8. (Brian) Fantasy Sports are growing in popularity every year but it is really hard to predict how players will perform from year to year. Can you use learning and probabilistic inference to make a model that predicts player performance year over year? [NN, RL, Bayes Nets, MDP]
9. (Brian) What's the probability of extreme weather events? Build a bayes net and use it to infer the relative probability of extreme weather events with and without global temperature rises [Bayes Net]
10. (Emily) Use CSPs to model and solve an optimal schedule for a Harvard student's daily life!
 - The basic version involves using CSPs to manage the time constraints, can optimize for different variables such as distance traveled between meetings (to create the most efficient schedule, so you're not walking in loops and wasting time walking back and forth to the quad)
 - May also combine the CSP solver with some sort of RL algorithm (maybe can use rewards == student's grades / psets / quality of time / reported stress levels) to figure out an ideal course schedule / extracurricular commitments.
 - Maybe you can even use this tool to figure out what are your true passions in life, and the activities that bring you the greatest fulfillment/reward.
11. (Emily) Use deep learning to predict diagnoses for different medical data including: Mammography for cancer, Tumor samples: benign?, Brain scans: neurological disorders

12. (Emily) Use search algorithms to create a planner for long trips (such as a trip across Europe or in Asia) in order to minimize travel cost and unnecessary flight travel/walking distances
13. (He) Model trading problem as a reinforcement learning problem. The final goal is to learn a policy (strategy). Use yahoo finance data. [reinforcement learning, trading, MDP]
14. (He) Simulated Annealing for portfolio selection problem. Portfolio selection problem consists on choice of best titles to constitute a portfolio which maximizes revenue and minimizes risk. It is well known that such a combinatorial problem is intractable with exact methods for large dimension problems. We can approach from Simulated Annealing method. [Simulated Annealing, portfolio optimization]
15. (He) Regime detection problem via Hidden Markov Model. In the financial market, the regime changes very frequently over time, including the trendy market and volatile market. The goal of this project is to use HMM to identify the underlying regime in the market. [Hidden Markov Model]
16. (J.T.) The program Libratus has performed excellently in No Limit Holdem against both human and AI opponents ([link to paper](#)). Can you build a game playing agent leveraging the decision making under uncertainty and learning portions of the course to play other games of imperfect information?
17. (J.T.) Lexical entailment is a semantic relationship between words where, in a sense, the meaning of one word is contained in another. For instance, 'puppy' entails 'dog' but 'dog' does not entail 'puppy'. Devise a model that, when trained, can predict instances of lexical entailment from reading the context of a sentence.
18. (James) Develop a reinforcement learning algorithm for a game with teams (such as capture the flag). Learn about collaboration and competition in reinforcement learning and experiment with features to achieve the best performance.
19. (James) Building off of Brian's CS only idea, create a system for figuring out a plan of courses at Harvard to satisfy a concentration. Allow for various options to optimize for (i.e. highest Q scores, lowest workload?). Investigate using CSP's and search algorithms to implement a fast algorithm to find the optimal (or approximately best) Curriculum Vitae.
20. (James) Design and implement a mapping algorithm for a robot to map a new location with obstacles and various other object. It will build up a map by searching the area, and then use the map to solve various tasks. You can use a particle filter to figure out where you are in the map.
21. (Max) Four part harmony music is composed under a set of commonly held constraints. Use your knowledge of CSPs to model the constraints of four part harmony and create some music to share with the class! Experiment with your own transition models to influence the variation within the constraint space. [CSP]
22. (Max) Starcraft – so hot right now. Use some of the APIs out there to build a RL agent for Starcraft, Starcraft 2, or Dota 2 (all are being worked on right now by Facebook, Google, and OpenAI respectively). Try to create agents to do smaller tasks within the game (superhuman agents to play the full game are still an unsolved problem). [RL]

23. (Max) Solve your favorite board game! Chess, checkers, Settlers of Catan, Ascension, Ticket to Ride, Risk, Blokus. Make sure that the game is sufficiently complex to warrant an interesting exploration. Possible implementations could use classical search, CSPs, MDPs, RL, really almost anything that's been taught in class (depending on the problem). [CSP, MDP, RL, classical search]
24. (Nisha) [OpenAI gym](#). Choose an environment. Train some agents to solve the problem in your environment. Compare performances of your trained agents. [RL, CSP, MDP, classical search]. Here are some examples of environments ([source](#)):
- (a) Classic control and toy text: complete small-scale tasks, mostly from the RL literature.
 - (b) Algorithmic: perform computations such as adding multi-digit numbers and reversing sequences. One might object that these tasks are easy for a computer. The challenge is to learn these algorithms purely from examples. These tasks have the nice property that it's easy to vary the difficulty by varying the sequence length.
 - (c) Atari: play classic Atari games (with integrated the Arcade Learning Environment)
 - (d) Board games: play Go on 9x9 and 19x19 boards. Two-player games are fundamentally different than other settings, because there is an adversary playing against you.
 - (e) 2D and 3D robots: control a robot in simulation. These tasks use the MuJoCo physics engine, which was designed for fast and accurate robot simulation. (Both Brian and Scott have experience with the [Drake](#) environment as well)
 - (f) NOTE: Keep in mind that generality is more important than scores. Try not to overfit or handcraft solutions to the particular tasks you choose.
25. (Nisha) Choose an interesting dataset. Build a classifier, NN, or RL algorithm that uses material from this course to predict outcomes with some reasonable accuracy / precision. [NN, classification, RL]. Here are some examples of interesting data sets that can be found on [Kaggle](#).
- (a) Predict the loan status [fully paid, late, etc.] of a Lending Club loan given the borrower's information
 - (b) Predict whether a transaction is fraudulent given anonymized credit card transaction
 - (c) Use NYSE data fundamental metrics to predict things like one day ahead prediction, the quality of momentum strategies, or security clustering strategies
 - (d) Use data on a breast cancer tumor to predict whether it is benign or malignant
 - (e) Build a network to predict who will die in the next season of Game of Thrones
26. (Sebastian) Train a language model to imitate how a public figure of your choice speaks (or tweets). Initially, most generated texts will be boring because they are the sentences with highest probability. The project is to generate interesting text that is still grammatical by modifying the probability distribution and applying a particle filter. [keywords: NLP, language generation, temperature, particle filter]
27. (Sebastian) An extension to beam search is to include a parameter that measures difference between a beam and other beams in order to receive diverse solutions, when there is no clear

“best” solution. Can you use this approach to generate diverse sentences that talk about the same topic? [keywords: NLP, sequence2sequence, beam search]

28. (Sebastian) Can you use reinforcement learning to identify sections and phrases that are most relevant to predictions? For example, the phrase “The drink is fruity and tasteful” is relevant for the rating of “taste” in a review. [keywords: NLP, RL, REINFORCE, classification]
29. (Suproteem) Develop an article recommendation system that takes into account the content of the articles and the user’s browsing history. Learn about relevant tools like TF-IDF to better classify articles and find similar content to consume. [NLP, classification]
30. (Suproteem) Develop an RL agent for Super Smash Bros Melee (or your video game of choice!). Develop a representation of state, action, and reward, and use an emulator to train your agent to different styles of play. [RL, MDP]
31. (Suproteem) Machine learning applied to healthcare has been a field gaining in prominence. Find a dataset like DDSM (Digital Database for Screening Mammography) with low signal-to-noise and determine what sorts of classification methods from class can be used to detect disease. Bonus points if you can offer some interpretation of your model’s findings. [deep learning, CNNs, classification]

4.2 TF ideas from 2016

1. Baseball season is almost over, so it’s time to create a schedule for next season! Model the season as a CSP incorporating the conference/division requirements for teams. Try to be as realistic as possible: teams typically play each other for a 3-4 game series in a row, certain holidays do or don’t have games. Bonus: try to optimize rest/traveling for players (having a variable n and m for n games in m days and a soft constraint to minimize this, maybe also keep track of distance traveled for a team and have a soft constraint to minimize that as well).
2. Uber Satisfaction Problem - given a set of drivers and passengers spread out in various locations/desired destinations, optimize the routes for the drivers. Try to implement a pooling mechanism to save the passengers money!
3. An extension of MDP’s is the case where the current state is not completely observable: partially observable markov decision processes. There are variations of value iteration for POMDP’s and it might be cool to try to modify the given pacman framework to run in a more “realistic” environment and make comparisons to the full information case.
4. It’s election season, which means that political rhetoric is as heated as ever. Explore relationships between different congresspeople using the models from this course. Consider using clustering algorithms to find congresspeople with similar voting patterns. Or Hidden Markov Models to analyze their speeches or bills.
5. Use the OpenAI Gym, a python-based platform for reinforcement learning tasks, to build an RL algorithm to solve a popular Atari game (<https://gym.openai.com/envs#atari>). Want to reimplement AlphaGo but don’t have a million dollars worth of GPU compute lying around? Try your hand at OpenAI’s smaller 9x9 Go board with reinforcement learning:

<https://gym.openai.com/envs/Go9x9-v0>. If you are feeling ambitious, considering using deep reinforcement learning with a neural network library like Torch or TensorFlow.

6. Who doesn't want to make money? Take data from equity markets and use reinforcement learning to construct an agent that can predict with some accuracy whether or not you should buy or sell a stock.
7. Create an agent that is a competent "general game player." Check out: <http://www.gvgai.net>. Other game ideas might include Ultimate Tic Tac Toe or games from OpenAI. Make sure if you pick a game that it is sufficiently challenging to constitute a final project.
8. Pick your favorite board game or card game and model it as a formalized game. Then, build a few agents to play the game using different game-playing algorithms. Modeling a board game or card game can introduce cool extensions to the minimax game-playing we saw in class, such as imperfect or incomplete information, multiplayer games, and combinatorial games. Additional research can be done on heuristic-based game-playing algorithms like Monte Carlo Tree Search. You can also compare reinforcement learning techniques to game-playing algorithms like minimax.
9. Track one or more than one objects in a video. You may build a Hidden Markov Model to do it. There are many off-the-shelf datasets for tracking, e.g., the motchallenge dataset. They usually provide you noisy observations of objects in each video frame, and your job is to find the motion trajectory of those objects as accurate as possible across frames.
10. Write an AI player for Hive - representing the state will involve some thought, but once you've done that there is lots of strategy for your algorithms to discover and a manageable branching factor
11. Map the connectome - turn imaged slices of a brain into a 3-D model by optimally matching the cross-sections of neurons. Note: this is a very challenging open research problem, so you will likely have to scale it back to something achievable.
12. Write a program that learns how to write/speak in the style of a certain person (i.e. Shakespeare, Yoda, Wikipedia) from a corpus of text by building a statistical language model.
13. Write a program that recommends news articles to read. Some potential directions this could go: article content tagging, machine learning recommendations, topic clustering, etc.
14. Write a CSP solving program that can decompose sentences into their component parts of speech.
15. Develop software that intelligently identifies trolls in online forums (reddit, Yahoo Answers, etc.)
16. Deep-Pacman: Think back to the first time you played Pacman. You looked at your TV screen started smashing buttons and, in a very short amount of time, figured out what the goal of the game was, what are the relevant state features, and how your actions changed the game state. In this project, you will build an RL agent that only has access to only pacman board pixels (no hand-crafted MDP state or features) and uses Deep Q-Learning come up with good policy.

17. Realtime bike routes: Often following Google maps for bike directions in Boston takes you down undesirable (bumpy, narrow, dangerous, etc.) roadways. Design a bike directions program for the Boston area that quickly returns the best bike routes between two points by taking into account information that can be mined from the web (e.g., bike accident frequency, road construction, dedicated bike paths, etc.). Users should be able to adjust their preferences to tradeoff speed with comfort/safety.
18. Learning to ride a bike: Download or implement a simulator for a bicycle model, define an MDP representation, and develop a RL algorithm that learns to balance and steer the bicycle.