Artificial Intelligence & Society: Project

STEM II Summer at Brown

Summer 2017

1 Overview

For your course project, you and a partner will design, implement (in python), and conduct experiments with an AI algorithm in a task of your choosing. Your project will follow the guidelines listed in the next section for one of the following four paths:

1. **Reinforcement Learning**: Write and experiment with a simple RL agent.
2. **Machine Learning**: Implement a basic machine learning algorithm on a classification-esque task involving images, text, or sound.
3. **Markov Chain**: Write a program that learns to generate novel objects, such as poems, quotes, recipes, or character names.
4. **Search & Heuristics**: Create an "old-school" AI to solve a simple game with search & heuristic methods.

The goal of the project is for you to get a taste of the scientific process involved in the study of Artificial Intelligence. You will have to determine how to design, implement, and evaluate the algorithm you choose, with specifics varying between the four different project types. In particular, support code will be offered to help get started.

Upon completion, you will be responsible for submitting your code and a short writeup including results, a summary of your process including experimental design, and a discussion of any conclusions you draw from the results. More details for the specifics of the hand-in are included in the final section.

.....

2 Project Paths

You can choose from four different paths, with the major direction and details left up to you and your partner.

2.1 Reinforcement Learning

The Reinforcement Learning path involves programming an agent to learn to play a variety of games. The core of the work will be split between implementing the agent and testing it in games compared to simple baselines. You have three steps to complete:

> **Checkpoint One (C1)**: Pick an aspect of RL to experiment with and a domain to use. Get this checked off by Dave and Carl.
>
> **Checkpoint Two (C2)**: Create an Agent class in the `simple_rl` library. Overwrite the `act(state, reward)` method.
>
> **Checkpoint Three (C3)**: Experiment with your new agent on the domain of choice. Compare it to baselines and do some diagnosis: How does it perform? Why? What insights can be gained from the results? What are its weakness? Document your answer to some of these questions, then *redesign* some fundamental aspect of the agent and rerun the experiments. How did your answers to the previous questions change?

2.2 Machine Learning

For the Machine Learning path, you will be tasked with implementing a classifier on a domain of relevance. For instance, you will write a program that learns to distinguish between Spam emails and regular emails. To complete this branch, you will need to:

> **Checkpoint One (C1)**: Pick a classification task and domain. Some options include spam detection, object detection in images, sentiment analysis in text, lie detection in sound bites, and genre detection in music.
>
> **Checkpoint Two (C2)**: Gather an appropriate dataset for the task. For more complex problems, you will need to rely on existing datasets. Check in with Dave or Carl about possible datasets to use.
>
> **Checkpoint Three (C3)**: Pick a classification approach. In class, we looked at the k-Nearest Neighbor approach, but there are many others. Depending on the complexity of the approach, you will either implement the algorithm yourself or use an existing implementation. If you use an existing implementation you will need to perform an additional step for *Checkpoint Four*.
>
> **Checkpoint Four (C4)**: Split the data set into training and testing. Train the classifier on the training data and evaluate the result on the test. Visualize and diagnose the results. If you used an off the shelf method, pick a parameter associated with the classifier. Experiment with the effect of changing this parameter and visualize the results. What parameter setting maximizes performance on the test data? What about on the training data?

For example, in the Spam detection case, you might choose to use a bag of words approach with a Naive Bayes classifier, which typically achieves up to 80% accuracy.

2.3 Markov Chain

A *markov chain* is a mathematical model that captures the behavior of processes over time. For this project, you will be *build* a Markov chain to generate music, text, poems, book titles, or another artistic medium of your choice. To complete this project, you will need to:

Checkpoint One (C1): Select a medium, such as poems, songs, sketches, fortune cookie phrases, song lyrics, or some other entity. Check with Dave or Carl first when you have one in mind.

Checkpoint Two (C2): Collect a dataset of these entities.

Checkpoint Three (C3): Implement an algorithm using a unigram model that can *generate* novel examples of the entities you choose. For example, I wrote a Markov chain to generate character names from Harry Potter books, which spit out the following names:

Drals Oiori *Setsgorde Wwnfljgetd* *Bocjun Tooak* *Alir Bioamec*

Clearly the names I've generated aren't perfect. You'll start with a simple approach and refine it over time until the output produced is as close to the original medium as possible.

Checkpoint Four (C4): Make one additional change to the model to improve performance. For instance, you might implement a smoothing technique, or add additional structure to the generated objects. As a default extension, you can extend the unigram model to a mixture of unigrams, bigrams, and trigrams.

2.4 Search & Heuristics

The final project path is a classic approach to AI: Search. Here you will investigate traditional methods of *searching* for good solutions to games. For instance, if you were trying to write an AI to control Pac-Man, what strategies might you employ?

Your task here is the following:

Checkpoint One (C1): Select a (relatively simple) game, somewhere between the complexity of Tic-Tac-Toe and Pac-Man. Some examples include checkers, backgammon, sudoku, connect four, solitaire, solving a maze, othello, or chess.

Checkpoint Two (C2): Pick one central aspect of the AI to implement via search and heuristics. For instance, in Pac-Man, you might write a path-finding algorithm that tells Pac-Man how to find the closest pellet. You will develop a heuristic that encourages good behavior. Implement this aspect.

Checkpoint Three (C3): Develop a method for evaluating the search method and heuristics used therein. How can you determine which ones work and don't work?

3 Expectations

Submit your project path form on the website by Wednesday at 5pm. Changes to the path can be made in rare circumstances up to as late as Thursday at 5pm, but you'll need to talk to Dave during office hours to get it approved.

You will be responsible for writing a short description of your methods and process upon completion of each checkpoint for your project path. These paragraphs will be submitted through a Google Form (available on the website). You must submit each paragraph to complete the project.

3.1 Timeline

Date	*Checkpoint*
W 6/28, 5pm[1]	C1: Submit your project proposal document.
F 6/30, 5pm	First pass at C2.
M 7/3, 5pm	C2 done, first pass at C3.
W 7/5, 5pm	Draft of C4 (ML/MC) or C3 progress (RL/SH path).

The first deadline is strict – you must submit your project proposal document by Wednesday 6/28 at 5pm. The remaining deadlines are flexible; please check in with Dave if you need more time for a particular checkpoint.

3.2 Evaluation

A more detailed rubric will be discussed in class (and made available on the website). In short, you will be evaluated based on three components:

1. *Checkpoints*: Did you complete the checkpoint paragraphs? Did they sufficiently demonstrate your design and scientific process?
2. *Hand-In*: Does your final hand in sufficiently represent your work? Emphasis here is on analyzing your results and reflecting on your methods.
3. *Code*: Does your code work? What did you test? What results did you end up with?

3.3 Hand-In

For posterity, you and your partner will be responsible for handing in your code for running experiments and a write up in a zip file. To hand-in your project, just email Dave the zip file containing these items. The writeup will include:

- Instructions for how to run your code.
- At least two plots or figures showcasing the results of your experiment.
- A conclusion that discusses the results, with some description of what to work on next.

If you need help at any point during the project, don't hesitate to reach out to Dave or Carl – we're here to help!