CIS667 Semester Project Instructions

Overview

In this project you will implement an AI system that plays Gomoku. The details of the AI system are up to you, but it must conform to a specific API that pits your AI against a baseline. You must submit your AI code as well as a brief write-up describing how it works; both will factor into your grade.

Deadline

The code and write-up are both due at 11:59pm on the last day of the semester, Tuesday December 19. ***There will be no extensions, because I need to submit grades soon after.***

Submission Instructions

Prepare a single zip archive named project.zip containing the following files:

- submission.py (this contains your AI implementation)
- report.pdf (this contains your write-up)

Optionally you can include additional Python modules with custom helper functions that you import in submission.py. You can also include data files (pickle, csv, json, etc.) that are loaded by your AI, such as optimized neural network weights. However:

- If you use machine learning, your submitted AI data should be optimized already; the instructor will not re-run the training.
- All other project scripts distributed with this assignment will be replaced with the instructor's copies, so you should not modify them and do not need to include them in your submission.

Code Structure

The code.zip archive available with this assignment has the following structure. ***You should not modify any of these files except for submission.py***, as they will be replaced with the instructor's copies anyway.

- gomoku.py: This module contains the implementation of the Gomoku game, including methods to list valid actions for every state, perform an action in a given state, and calculate the score in a game state.
- compete.py: This module runs a full game between two players. Each player can be controlled by a human or various automated policies.
- performance.py: This module runs several games between your AI and the baseline policy. The plots at the end visualize the final scores and run times of each AI. It saves the results in a file named perf.pkl.
- policies/: The modules in this sub-directory include various policies that can be selected for each player.
    - human.py: This policy is human-controlled
    - random.py: This policy chooses actions uniformly at random
    - minimax.py: This policy chooses actions using an augmented Minimax Search
    - submission.py: This policy will use your AI implementation

API Description

The rules of Gomoku are the same as in HW2, but the implementation is different. The max player uses the symbol “x” and the min player uses the symbol “o.” Internally, the board is represented by a 3D NumPy array with shape (3, board_size, board_size) and entries that are either 0 or 1. The leading dimension corresponds to the status of each position: whether it is empty, occupied by the min player, or occupied by the max player. Specifically, if position (r,c) is empty, then state[0,r,c] == 1, otherwise state[0,r,c] == 0. Similarly, if the min player has a mark at position (r,c), then state[1,r,c] == 1, otherwise state[1,r,c] == 0. Likewise for the max player, in state[2,r,c]. One example 3x3 state is as follows:

Text representation	state[0,:,:]	state[1,:,:]	state[2,:,:]
..x	110	000	001
.o.	101	010	000
x..	011	000	100

A Gomoku game is parameterized by board size and a win size. The board is a square and board size is the side length of the square. The win size is the number of marks that must be placed in a row to win. For example, the standard 15x15 Gomoku domain has a board size of 15, and a win size of 5 (meaning 5-in-a-row to win).

The game score rewards winners for finishing more quickly. The sign of the score indicates whether min or max won. The magnitude of the score is the number of empty positions left at the end of the game (plus 1).

Actions are represented by a tuple (r,c) which indicates that the current player will put their mark at row r and column c. Row and column indices are zero-based.

Like HW2, game scores are determined by correlating the board with win patterns (horizontal, vertical, diagonal, anti-diagonal). Unlike HW2, the correlations are not calculated from scratch at every state, but incrementally updated every time an action is performed, which makes the code run faster. If you want to use it in your AI (which is optional), you can access the correlations using the state.corr attribute. The comments in gomoku.py provide more detail about this attribute. You are free to use it, but your code should not modify it.

The comments in gomoku.py also provide more information on the available helper methods on gomoku states, which return things like current player, score, valid actions, or the new state after an action is performed.

For examples of an AI using the gomoku helpers, check the code and comments in policies/minimax.py.

Implementation Guidelines

Each policy, including the one you must implement, is a Python class with the following methods:

- __init__(self, board_size, win_size): This must method initialize the policy for the given Gomoku game parameters. This initialization can also optionally do things like load lookup tables or optimized neural network weights.
- __call__(self, state): This method must return a valid action in the given state. This is the entry point to your AI method.

You can run a game between two policies by running compete.py on the command line with options to specify board size, win size, and each player's policy. For example, the command

```
python compete.py -b 15 -w 5 -x Human -o Minimax
```

will run a game on a 15x15 board with 5 in a row to win, where you control the max player, and the min player selects actions randomly. For each policy you must write the exact name of the policy class (case-sensitive). The available policies are Human, Random, Minimax, and Submission (which is yours). Press Ctrl-C, or a similar command in your operating system, to interrupt the script at any time and terminate early before the game is over.

Run the script performance.py to play your AI against the Minimax baseline. The baseline is the max player and gets to go first. Your AI is the min player and goes second. This means that your AI wins when the final score is negative. Multiple games are played, and the final game scores and run times of each game are recorded. The statistics of these metrics will be used to calculate your grade on the coding portion. To receive a decent grade, your AI should perform relatively well against the baseline, both in terms of average final score, and average time per turn. The better your AI is relative to the baseline, the higher your grade will be.

35% of your project grade is based on your AI's performance against the baseline. If your AI takes more than 10x as long as the baseline, you may receive a performance grade of zero.

If your code crashes, you will receive a performance grade of zero. It is recommended that after uploading submission.py, you download it from blackboard into a fresh directory and run it using the original copies of the other files, to make sure it behaves as expected and does not crash.

Write-up Guidelines

The write-up should be 1 page single-spaced, not including the bibliography. A second page for the bibliography is acceptable. You do not need a title, but the names and NetIDs of all students should be at the top of the page. The rest of the page should summarize how your AI is designed at a conceptual, mathematical, and algorithmic level.

65% of your project grade is based on the writeup. That 65% will be distributed according to the following rubric:

Format and style (5%)

- Include names and NetIDs at top
- Use proper spelling and grammar
- Use bullet-point lists sparingly; most of the write-up should use complete sentences and paragraphs.
- Be thorough enough to use the full page (single-spaced), but concise enough to fit within one page.
- Include at least one citation for each AI method and each code library you use. Any bibliography format is fine as long as it is consistent and widely used (e.g. MLA, APA, etc.). Each reference in your bibliography should also have an in-text citation. https://scholar.google.com makes it easy to search for relevant references and obtain their citations in a format of your choice. You can also Google "citing numpy," "citing scipy," "citing pytorch," etc., to see their preferred citations. For AI methods (not code libraries), books and research articles are acceptable sources, but websites are not.

Novelty (10%)

Your method should involve some level of creativity beyond what was covered in lecture examples and the baseline code.

- If you only change some hyperparameters (e.g., depth limits, exploration coefficients in the UCT formula, etc.), you will receive a low novelty score.
- You will receive a good novelty score for a more substantial algorithmic modification, such as designing a custom minimax evaluation function or child selection formula in MCTS.
- For full novelty credit, you should make a more fundamental change, such as a custom neural network architecture or a different tree search algorithm we did not cover in class.

Method (30%)

You should clearly explain your method so that the reader can form a reasonable understanding even without reading your code. Use mathematical formulas, short pseudocode, or small diagrams and figures as needed to most efficiently and clearly convey your approach. Some important details that should be covered are:

- If you used a custom minimax evaluation function, what was its formula or logic?
- If you used a custom child selection in MCTS, what was its formula or logic?
- If you used a tree search other than minimax or MCTS, what was its pseudocode?
- If you used a neural network, what are its architectural diagram and hyper-parameters? What is the formula for its loss function? What was the learning curve (plot of loss vs number of gradient updates) during its training?

Discussion (20%)

Conclude with a discussion of your AI's performance. Did it perform better or worse than you expected? What is your best hypothesis as to why the performance did not match expectations, or what the main limiting factor in performance was? If you were to continue working on this project in the future, what experiments could you do to test your hypothesis and what changes would you make to the implementation to further improve performance? What was the most difficult challenge in this project?

Academic Integrity

If any violation of academic integrity is suspected on your project, it will be reported to the university, and may result in a zero grade on the project. It is your responsibility to review and understand the university's academic integrity policy. Violations include, but are not limited to:

- Copying text from the internet into your write-up without proper attribution, even if you make some modifications to it. Most of the text should be your own writing, but any copied text must be copied exactly verbatim (without modifications), and it must be surrounded by quotation marks, and it must be indented relative to the rest of the text, and immediately before or after the quote, you must state the original author's name and include a citation to the associated reference in your bibliography.
- Copying code from the internet without proper attribution. Any code that you did not author yourself must have its source acknowledged in your bibliography and in a comment in the relevant code files themselves.

- Fabricating results in your report. All data-points that you include in any plot or describe in the text must have been generated from actually running your code.