Project Proposal: AI Learning Agent for the Game of Battleship

Jordan Ebel (jebel), Kai Wan (kaiw)

Task Definition

We propose to implement a game-playing agent for a modified version of the game of Battleship.

The game of Battleship is played on a square grid (traditionally 10-by-10). A number of ships, each 1 square wide and between 1 to 5 squares long, are placed on the grid either horizontally or vertically with no overlap. The ships are hidden from the player, who each turn makes a guess and selects a square to strike. The player is then informed whether the strike resulted in a hit or a miss. The game continues until a player has successfully hit all the squares occupied by ships (i.e. sunk all the ships).

In this proposal, we plan to make some specific modifications to the game as follows:

- The game will be played by a single player (AI agent), who selects one square of the board to strike each turn.
- The game board will be of size m-by-m. We intend to test our implementation on various sizes of m, including the traditional 10 but also smaller and larger sizes.
- The number and lengths of the ships will differ from the traditional arrangement (5 ships of lengths 5, 4, 3, 3, 2). We intend to test various ship numbers and sizes and observe the impact on our results.
- The player (AI) will have no prior knowledge of the number and sizes of the ships.
- Optionally, we can extend our project to include a separate algorithm for placing ships. With this extension, we can support full two participant gameplay.

Evaluation

The main metric for game-playing proficiency will be the player's hit-rate, i.e. the ratio of strikes taken that resulted in a hit. Equivalently, a player should attempt to sink all ships on the board in as few strikes as possible.

Approach

Our goal is to implement an AI agent that will learn to become proficient at playing our modified version of Battleship. More specifically, we require our implementation to meet these requirements:

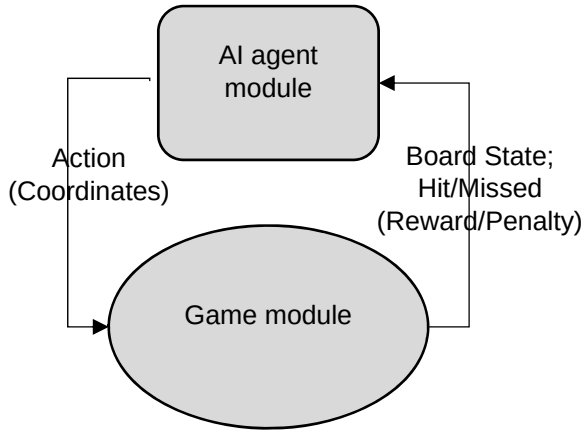
- Our implementation shall have no prior knowledge or assumptions about the game of Battleship. It will not rely on any handcrafted features that reflect game knowledge.
- Our implementation shall rely on a reward-based feedback to learn the optimal way to play the game through reinforcement learning.

We believe these requirements present a worthy challenge and also makes an interesting opportunity to compare against non-learning, rules-based approaches (See Baselines and Oracles). The main difficulty of the project lies in the large state-space, which increases exponentially with the game board size. The challenge, then, is to find ways to search the state space more quickly and efficiently.

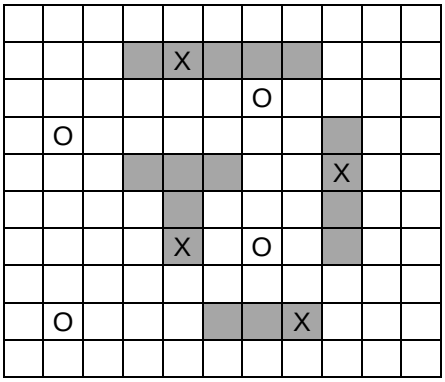
Modeling

Our system will consist of a learning agent module and a game module, which serves as the environment with which the agent interacts. The only input from the agent to the game will be actions, which are simply coordinates on the game board that the agent selected to strike. The feedback output from the game to the agent consists of an indication of hit or miss, along with a corresponding reward or penalty.

The states of the game are each represented by a complete game board of m x m squares. Each square may be in one of the following statuses: Unexplored, hit, or missed.



System and interaction between agent and the Battleship game



Example game board representing one possible state. Shaded squares contain parts of hidden ships. Squares with 'X' represents hits; those with 'O' represent misses

Baselines and Oracles

We plan to use several non-learning approaches as baselines and oracles to compare with our learning-based AI agent:

- A random algorithm can simply select squares at random to strike each turn. It is simple but not very interesting. We can use this to represent the absolute lower bound on effectiveness.
- A hunt-and-target algorithm is a simple implementation that improves on the random algorithm. At first, it also selects squares at random to strike. But after a strike results in a hit, it will target the squares adjacent to the hit square during subsequent turns. This makes a good benchmark to compare against the AI algorithm.
- Humans can play the game of Battleship reasonably well, and a human player may serve as good benchmark for comparison.
- The theoretical upper bound to effectiveness would be 100% hit rate, i.e. all strikes result in a hit and the game is completed in the minimum possible number of turns. We can consider an oracle algorithm that has knowledge of ship placements to represent this upper bound.

Literature Review

We have found several similar attempts to implement a Battleship algorithm, but each have somewhat different goals, approaches and results.

- C. Compton, J. Liu, N. Stanzione. "Battleship: A Hit or Miss Affair". http://www.cs.uml.edu/ecg/uploads/AIfall14/compton_liu_stanzione.pdf

Attempts to implement a learning based AI that was not very successful due to large state space.

- "Battleships - A Game Playing Agent". http://gritslab.gatech.edu/Pickem/wp-content/uploads/2012/12/cs6601_project_1_proposal.pdf

Attempts to implement a search problem without learning.

- N. Berry. "Battleship". DataGenetics. http://www.datagenetics.com/blog/december32011/index.html

Developed a very effective Battleship algorithm, but is a non-AI solution that incorporates very specific knowledge of game mechanics.