Informal Documentation

Gemmin Sugiura

May 27, 2025

Table of Contents

Here is a list of explanations of the models used for the AIs for this project.

- **Rock Paper Scissors (RPS)**: Markov Chains
- **Stone Game**: Dynamic Programming
- **Checkers**: Alpha Beta Pruning
- **Wordle**: Information Theory

1 Rock Paper Scissors

The model uses Markov chains. An m-th order Markov chain assumes the next move depends on the previous m moves. Formally, for an opponent's move sequence $X_1, X_2, \ldots, X_m$, we have $P(X_{t+1} = x \mid X_t, X_{t-1}, \ldots, X_{t-m+1})$, where a state $h = X_t, X_{t-1}, \ldots, X_{t-m+1}$ denotes the last m moves. For state h, since $P(x|h) \propto C(h, x)$, where $C(h, x)$ is the counts of observing (h, x). Therefore, we can track the counts for transitions.

To enhance the model, it employs ensemble learning. Multiple Markov chains of varying orders generate their predictions. The model with the highest win rate is picked. The win rate is computed over a sliding window of size of the focus length hyperparameter. During the first several rounds of RPS, the system prioritizes stochastic sampling, in which the exploration rate is controlled by the exploration rate hyperparameter.

2 Stone Game

In the classic 21 stones game with moves $\{1, 2, 3\}$, we analyze the probability of AI victory under different player strategies. When the player plays optimally, the outcome is deterministic based on the starting configuration and the starting player. In any configuration, the starting player is either winning or losing. Usually, there should be a fair coin flip to determine which player goes first, but I manipulated the first-move selection process so that the first-move selection has a 0.75 chance of favoring the AI and a 0.25 chance of an actual coin flip. Thus, the probability that an optimal player will win is only 0.125.

For a random player, the winning probability decreases significantly. Even when the player gets the theoretically winning first move, they must play optimally throughout the entire game. Since the player chooses randomly among valid moves at each turn, their probability of making the correct choice at any given position is $\frac{1}{3}$ (assuming all three moves are typically available). Over the course of an average game lasting approximately 7 moves, the probability of the player maintaining optimal play becomes $(1/3)^k$ where k is the number of critical decision points. This exponential decay in optimal play probability, combined with the strategic first-move manipulation, results in the observed $0.125 \times (1/3)^7 = \frac{1}{17496}$ overall winning probability for the player.

3 Checkers

Alpha-beta pruning is a search algorithm optimization technique that significantly reduces the number of nodes evaluated in the minimax tree for adversarial games like checkers. The algorithm maintains two values, α (the best value the maximizing player can guarantee) and β (the best value the minimizing player can guarantee), which represent bounds on the possible outcomes. When $\beta \leq \alpha$, the algorithm can safely prune entire subtrees because the opponent would never allow the game to reach those states. In our checkers implementation, the AI uses a depth-limited search with alpha-beta pruning to explore game states up to 4 moves ahead.

The utility function serves as the critical evaluation mechanism that assigns numerical scores to board positions, enabling the AI to make informed decisions about move quality. Our implementation uses a comprehensive scoring system that considers multiple strategic factors: piece count differential (regular pieces worth 1 point, kings worth 5 points), positional advantages (pieces closer to promotion receive bonuses), mobility (number of available moves), and center control (pieces occupying central squares receive positional bonuses). The evaluation function is calculated as $U(s) = \sum_i w_i \cdot f_i(s)$, where $f_i(s)$ represents different board features such as material balance, king promotion potential, and strategic positioning, while w_i represents their respective weights.

4 Wordle

Information theory quantifies uncertainty and the amount of information contained in random events. For a discrete random variable X with outcomes x_i occurring with probabilities $p_i = P(X = x_i)$, the entropy $H(X)$ measures the expected information content: $H(X) = -\sum_i p_i \log_2 p_i$, where the logarithm base determines the unit (bits for base 2). Entropy represents the average number of bits needed to encode the outcomes optimally. The information content of a specific outcome x_i is $I(x_i) = -\log_2 p_i = \log_2 \frac{1}{p_i}$, meaning rare events carry more information than common ones. This formulation ensures that: (1) information is non-negative, (2) certain events ($p_i = 1$) carry zero information, and (3) information combines additively for independent events.

The entropy $H(X)$ achieves its maximum value when all outcomes are equally likely, i.e., when $p_i = \frac{1}{n}$ for all i, yielding $H_{\max}(X) = \log_2 n$. This maximum entropy principle underlies optimal decision-making strategies: to minimize expected uncertainty, choose actions that maximize information gain. In the context of word-guessing games, each guess partitions the remaining possible words into groups based on feedback patterns, and the **information gain** equals the entropy of this partition: $IG = \sum_j \frac{|G_j|}{|W|} \log_2 \frac{|W|}{|G_j|}$, where W is the set of remaining words and G_j are the groups formed by different feedback patterns. The optimal guess maximizes this expected information gain, systematically reducing uncertainty about the target word until convergence.