Comparative Analysis of Static and Machine Learning Algorithms in Perfect-Information Games

Background

Research Question

This investigation examines the relative performance of static and machine learning (ML) algorithms in perfect-information games. Static algorithms utilize predetermined decision-making processes, while ML algorithms require training data to optimize their parameters. The study encompasses three classic games: Tic-Tac-Toe, Nim, and Connect-4, chosen for their well-defined rules and complete information nature.

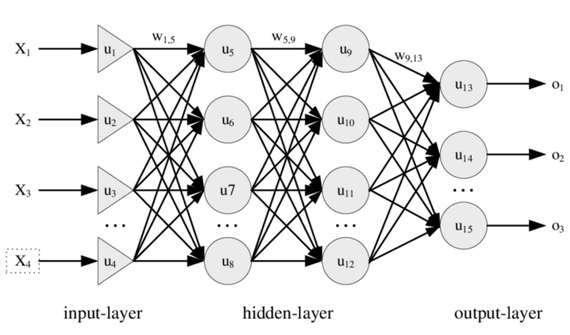

Figure 1. An illustration of a simple multi-layer perceptron (MLP).

Hypothesis

The research posits that static algorithms will demonstrate superior performance compared to their ML counterparts in head-to-head competition. This hypothesis is predicated on the theoretical understanding that static algorithms can achieve optimal play in solved games[1], whereas ML algorithms inherently maintain some degree of probabilistic decision-making even after training. The deterministic nature of static algorithms should therefore confer a competitive advantage in these specific game environments.

Design

Methodology

The experimental design involved implementing open-source implementations of both algorithmic approaches for each game. Q-learning[3] was used as the ML algorithm and minimax[4] tree-search as the static algorithm for all games. The researcher then developed a custom evaluation framework to facilitate head-to-head competitions and record outcomes. The framework, programmed in the Python programming language, handled the training of the ML algorithms, simulation of the games between the two algorithms, and the recording of the scores.[2] Multiple series of 1000 games were conducted, with the ML algorithm's training episodes serving as the primary experimental variable. The framework recorded wins, losses, and draws for subsequent analysis.

Variables

- Independent Variable: Number of training episodes for the ML algorithm
- Dependent Variable: Game outcome metrics (win/loss/draw ratios)

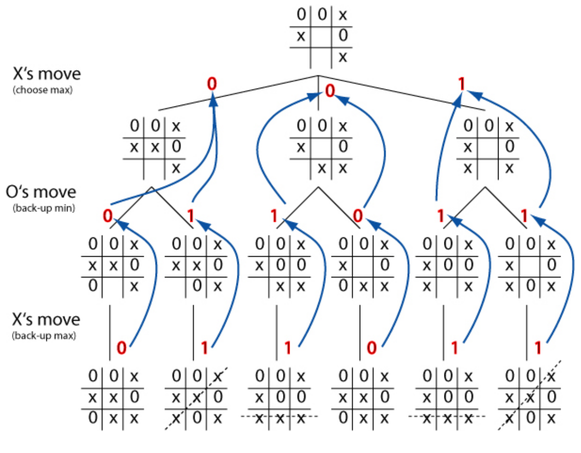

Figure 2. An illustration of a simple minimax search tree.

Data

Tic-Tac-Toe

Training length (episodes)	ML wins	Algorithm wins	Draws
10000	321	0	679
50000	352	0	648
100000	342	0	658
500000	302	0	698
1000000	348	0	652

Nim

Training Length (episodes)	ML Wins	Algorithm Wins
10000	27	973
50000	30	970
100000	34	966
500000	23	977
1000000	23	977

NOTE: Although the original intent was to collect data on Connect-4 as well as Tic-Tac-Toe and Nim, the researcher did not have access to sufficient computing resources to train a model or collect data for Connect-4.

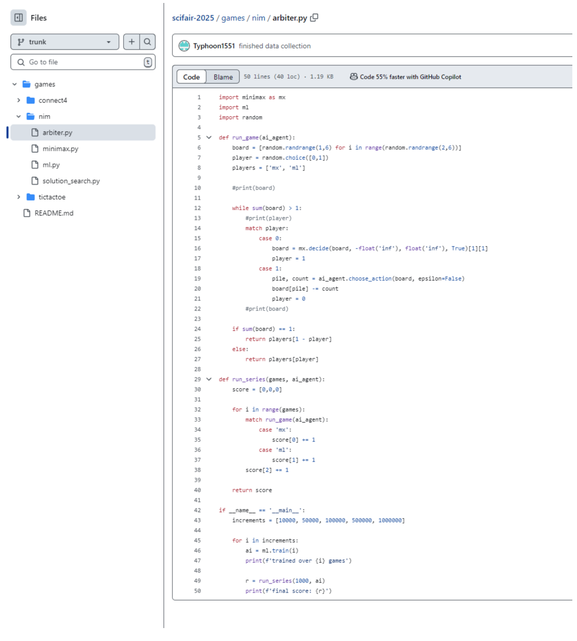

Figure 3. Code sample from this project's Github page.[2]

Conclusions

Analysis

The data yielded contrasting results across different games. In Nim, the results aligned with the initial hypothesis, with the static algorithm (minimax) achieving significantly higher win rates across all training configurations. However, the Tic-Tac-Toe results demonstrated an unexpected pattern, with the ML algorithm achieving complete dominance (100% win rate) across all training levels. This stark disparity suggests potential implementation issues in the static algorithm for Tic-Tac-Toe, rather than a fundamental limitation of the approach.

The inability to collect Connect-4 data represents a significant limitation in the study's comprehensiveness. The contradictory results between Nim and Tic-Tac-Toe, coupled with the missing Connect-4 data, preclude definitive conclusions about the relative efficacy of these algorithmic approaches.

Final Conclusion

While this study provides valuable insights into algorithmic performance in perfect-information games, the conflicting results and incomplete dataset necessitate further investigation. Future research should address the implementation challenges encountered, expand the game selection, and ensure more robust algorithm implementations. Additional metrics such as computational efficiency, decision-making time, and ease of implementation could provide more comprehensive evaluation criteria.

[1] Solved games are games for which a mathematical strategy has been found to always play the most correct move in any given position.

[2] The source code for this project can be viewed at https://github.com/Typhoon1551/scifair-2025

[3] A type of reinforcement learning in which values are assigned to each possible move for a given state.

[4] A type of tree-search algorithm in which lines are simulated until the end of the game and moves are evaluated based on possible outcomes.