Tic-Tac-Toe AI Agents Comparison

TEAM MEMBERS:

- JONNALA YAMINI
- SUPRIYA ARETI
- SRIJA VADDELLI

Project Topic:

Implementing multiple AI agents for Tic-Tac-Toe using Minimax, Alpha-Beta Pruning, and Q-Learning

Goal:

Compare their effectiveness in terms of game performance, computational efficiency, and adaptability and identify the most effective AI approach for Tic-Tac-Toe

Statement of Objectives :

Objective 1: Designing and implementing three AI agents using:

Min-Max Algorithm

Alpha-Beta Pruning

Q-Learning (Reinforcement Learning)

Objective 2:

Conduct multiple rounds of gameplay to identify the most effective algorithm based on performance metrics.

Approach:

Techniques:

- Min-Max: Calculate optimal moves through exhaustive search.
- Alpha-Beta Pruning: Prune unnecessary branches in the Min-Max tree to enhance speed.
- Q-Learning: Use reinforcement learning to train an agent capable of adapting over time.

Tools:

- Programming Language: Python
- IDE: Visual Studio Code (VS Code)
- Libraries: NumPy- For handling arrays and mathematical operations, Matplotlib- To visualize performance data and learning curves for Q-learning.

Deliverables:

- Proposal Presentation: Detailed breakdown of objectives, methodology, and evaluation.
- Technical Presentation: Overview of implementation, algorithm comparisons, and insights.
- YouTube Video: Demonstration of the project and its key functionalities.
- GitHub Repository: Documentation, code, resources, and YouTube link.
- Relevance: Each deliverable showcases the project’s objectives, methodology, and results, enabling a comprehensive understanding of the AI agents and their performance.

Evaluation Methodology:

Minimax & Alpha-Beta Pruning:

- Move Accuracy: How often the agent selects the optimal move.
- Execution Time: Measures computational efficiency per move.
- Memory Usage: Quantify memory requirements during gameplay.

Q-Learning:

- Learning Curve Analysis: Plot learning curves across episodes, showing improvements over time.
- Exploration vs. Exploitation Analysis: Track the Q-Learning agent's winning rate as epsilon (in epsilon-greedy strategy) varies, indicating how exploration impacts learning.
- Sample Complexity: Number of episodes needed to reach optimal or near-optimal play.
- Regret: Calculate the difference in rewards from optimal moves across episodes.
- Average Total Rewards per Episode: Evaluate consistency and improvement in performance over time.

Execution Time Comparison: Highlight computational efficiency of each algorithm. Measure and compare execution times for Minimax, Alpha-Beta Pruning, and Q-Learning agents. Test on different board sizes (e.g., 3x3 and 5x5) to show how each algorithm scales.

Manual Comparison: Direct comparison of each agent's gameplay outcomes, identifying strengths and weaknesses across metrics.