Project Report: Connect Four AI

Project Description

Objective
The primary goal of this project is to develop a reinforcement learning model to play the Connect Four game. The model utilizes a Q-learning algorithm to enable an artificial intelligence (AI) system to learn optimal moves against a human player or another AI based opponent through repeated gameplay.

Implementation
The game is programmed in Python, utilizing the NumPy library for array manipulations, which is critical for maintaining the game state. The AI leverages a Q-table to store and retrieve values corresponding to the quality of actions taken from specific states.

The core of the code involves:

- **Board Management**: Functions to initialize the game board, display the board, and identify valid moves.
- **Game Mechanics**: Functions to place discs in the board columns, check for a winning game state, and simulate player moves.
- **AI Logic**: Using Q-learning, where the AI selects actions based on a balance of exploration of new moves and exploitation of known strategies. The Q-table is updated continually based on the reward schema defined for winning, losing, drawing, or making a move.
- **Persistence**: Using the \`pickle\` library, the Q-table is saved and loaded from disk to preserve learning between sessions.

Data
The data utilized in this project is the game state of the Connect Four board, which is represented as a 2D NumPy array. Each element of the array represents a cell on the board, which can be empty, filled by the human player or filled by the AI.

CONTRIBUTIONS

Novelty

The novelty of this project lies in the application of Q-learning in a game that requires strategic depth beyond simple move prediction. Unlike standard implementations:

- **Predictive Blocking**: The AI not only plays to win but also actively predicts and blocks the opponent's winning moves.

- **Dynamic Exploration**: The balance between exploration and exploitation is dynamically adjusted based on gameplay, enhancing the AI's learning curve.

Other Contributions

While leveraging NumPy for array manipulation and `pickle` for data persistence, the implementation of the Q-learning algorithm and the integration of game-specific heuristics were developed from scratch.

- **Custom Reward System**: Tailored rewards and penalties that align with the game's strategic nature, influencing the AI’s learning priorities.
- **State-Action Mapping**: The design of a Q-table that effectively maps board states to potential actions, enabling rapid decision-making.
- **Performance Optimization**: Implementing efficiency improvements in the game state evaluation to speed up the AI's decision-making process.