

Connect Four AI Project

User and Developer Guide

Goal

This program simulates and trains an Artificial Neural Network (ANN) player to improve its performance in Connect Four by comparing its evaluations against a baseline player (MinMaxPlayer).

File Overview

- **main.cpp**: The core training logic and simulation loop.
 - **PlayerClass.h**: Defines the Player, ANNPlayer, MinMaxPlayer, and AIPlayer classes.
 - **BoardClass.h**: Defines the Board class that handles the Connect Four grid and game logic.
 - **PlayerImplementation.cpp**: Implementation for the functions declared in PlayerClass.h.
 - **BoardImplementation.cpp**: Implementation for the functions declared in BoardClass.h.
-

Key Concepts

- **Board State**: A grid representing current moves in the Connect Four game.
 - **ANNPlayer**: A machine learning-based player that evaluates board positions and learns from outcomes.
 - **MinMaxPlayer**: A traditional AI player using the MinMax algorithm to play optimally.
 - **AIPlayer**: A simple player used to simulate valid random moves in a game.
 - **Training**: The ANN is trained using error-based backpropagation on sample games.
-

Program Flow

1. **Startup**
 - Outputs welcome message.
 - Seeds the random number generator with the current time.
2. **Random Board Generation**
 - `determineRandom()` creates a legal game board state after a random number of moves (1–41).
 - Ensures the board is not in a terminal state (no winner or draw).
3. **Simulate Baseline Performance**

- runSample(board) simulates set number of games between two MinMaxPlayer AIs from the same board state.
- Returns the win percentage for the player using token +1 (i.e., Player 1).
- 4. **Train ANN**
 - For each random board, the ANNPlayer evaluates the board.
 - runANN() compares the ANN's evaluation to the actual win percentage and trains the network.
 - Training is repeated indefinitely or across a set number of iterations for a series of random board states.
- 5. **Display Results**
 - playerTest.displayWeights() prints the final ANN weights after training.
- 6. **Continuous Training**
 - A commented-out loop in main allows continuous training using newly generated random boards.

Main Function Summaries

| Function | Purpose |
|--------------------|--|
| determineCurrent() | Determines whose turn it is based on board tokens. |
| runSample() | Simulates 100 games and returns Player 1's win rate. |
| runANN() | Trains the ANN by comparing its evaluation to expected win rate. |
| determineRandom() | Builds a valid random board after simulating random moves. |
| writeError() | Logs test number and error value to data.csv. |

PlayerClass Function Summaries

| Function | Purpose |
|-----------------|---|
| getMove() | Virtual move function that returns move's column for each player. |
| minimax() | Minimax implementation to evaluate best possible move. |
| Softmax() | Softmax implementation to add randomness to move determination. |
| evaluateboard() | Virtual heuristic function for minMax and ANN players. |
| ANNfunction() | Performs forward pass for ANN board evaluation. |
| backPropagate() | Performs backward pass for ANN training. |

BoardClass Function Summaries

| Function | Purpose |
|---------------------|---|
| displayBoard() | Writes board to screen using created format. |
| makeMove() | Adds specified token to current board state. |
| resetBoard() | Reset board to empty state (all 0s). |
| loadBoardFromFile() | Reads board state in from binary file specified in function definition. |
| saveboardToFile() | Saves board state to binary file specified in function definition. |

Creating Players

- Creating players requires two parameters, player token and amount of lookahead.

HumanPlayer(± 1 , 0) Human player does not use lookahead, default to 0.

AIPlayer(± 1 , 0) AI player does not use lookahead, default to 0.

MinMaxPlayer(± 1 , 4) Create MinMax Player with four lookahead.

ANNPlayer(± 1 , 6) Create ANN Player with six lookahead.

Notes

- ANNPlayer is trained via **supervised learning**, not reinforcement learning.
- It is evaluated using MinMaxPlayer as a benchmark for expected performance.
- All training error values are logged in data.csv for later analysis.