PROJECT REPORT

Submitted by:

Sampada Talwar	NetID: *satalwar*
WeiDong Wang	NetID: *wwang43*
Jasmir Pinakin Kharwar	NetID: *jkharwar*
Manha Garg	NetID: *magarg*

GitHub repository:

https://github.com/sampadatalwar/CIS667_AI_Project_Chess

CIS 667 Introduction To Artificial Intelligence

Fall 2021

Syracuse University,
Syracuse,
New York - 13244

Introduction:

Chess is a board game played by two players. The present form of the game originated in southern Europe in the late 15th century, and evolved from a similar and much older game of Indian origin. Today, chess is one of the most popular games in the world, played by millions of people around the world.

Chess is an abstract strategy game with no hidden information. Played on a square chessboard with 64 squares arranged in an 8x8 grid. Initially, each player (one white and the other black) controls 16 pieces: 1 king, 1 queen, 2 rooks, 2 knights, 2 bishops, and 8 pawns. The goal of the game is to checkmate the opponent's king, who is immediately attacked ("checked") and has no way to escape. There are also several ways a game can end in a draw.

We've devised a mini version of the game where a player can play 4x4, 6x6, 8x8 and 7x7 grids. The pieces in a 6x6 grid for each black and white player consists of 4 rooks, 6 pawns, 1 king and 1 queen. With respect to rules of the game, in any grid the player wants to play there is always 1 king and 1 queen, so depending on a grid that player is playing the number of pawns and rooks increases.

Types of AI:

We've created 3 AIs with which a player can play a single player game. Also two modes of AIs can compete against each other where the game would run for some time without involvement of any player.

Baseline AI:

This AI takes its decision uniformly at random from the available choices. The decisions taken by this AI are chosen at random and will not take into consideration the abilities of the human user and consequences of the move.

Tree Based AI:

Minimax is a basic decision-making algorithm used in artificial intelligence. It uses intelligence, game theory and many other areas such as statistics, philosophy, etc. to maximize profits by minimizing possible losses in worst case scenarios. It was originally formulated for the zero-sum game theory, where players make alternating or simultaneous moves. This algorithm was later extended to more complex games and general decision making which involved uncertainty. Consider the case when an opponent makes mistakes that help the AI (a"good case"), but they can also choose actions very bad for the AI (a "bad case"), or as bad as possible for the AI (the "worst case"). The minimax algorithm gets the best out of the worst case possible outcome for AI. In the event of an unforeseen error by the player, the AI can perform much better than the results guaranteed by the minimax algorithm.

For this project we've used the Minimax algorithm which would try to find an optimal move from the tree of moves that can be made in a chessboard. In this the optimal moves are looked upon from a tree in a bottom to top direction (from the leaves) by altering between min() and max() function. That is, if an AI is playing the black team on a chess board then it would try to optimize the outcome by calculating the max position score while during an opponent's turn it would try to minimize your outcome by calculating min position score.

The scoring of the game is done with the perspective of AI, so that an AI can maximize the final game score. Positive final game scores are good for AI and bad for a player playing the game, while negative final game scores are good for the player and bad for AI.

An AI playing chess based on the minimax algorithm is associated with the "Max" player, while a player of this game is associated with the "Min" player. The basic structure of Minimax algorithm is as follows:

```
function minimax(node, depth, maximizingPlayer) is
    if depth = 0 or node is a terminal node then
        return the heuristic value of node
    if maximizingPlayer then
        value := −∞
        for each child of node do
            value := max(value, minimax(child, depth − 1, FALSE))
        return value
    else (* minimizing player *)
        value := +∞
        for each child of node do
            value := min(value, minimax(child, depth − 1, TRUE))
        return value
```

Neural Network AI:

Neural networks, also known as artificial neural networks, are computer systems based on t he interpretation of biological neural networks that make up animals. brain. Neural networks are built on top of a set of nodes called neurons, just like neurons in a biological brain. Each connection can transmit one signal to other neurons as well as the synapses of the brain. An artificial neuron receives a signal, then processes it, then transmits the signal to the neurons connected to it. The connections are called edges, and the signal at each connection is a real number. Results of each neuron are computed using a nonlinear function of its inputs. All neurons and edges have adjusted weights as the model learns. The weight increases or decreases the signal strength at a connection. The neural network model

is trained to reduce the square root mean square error to zero. Normally, neurons are assembled into a set of layers, but if the model is not complex then just a single layer. Separate layers perform separate transformations on their inputs. Signals move from the first layer to the last layer to provide the final output.

Chess Implementation Rules:

We've devised a mini version of a 2 player chess game where a player can play 4x4, 6x6, 8x8 and 7x7 grids. The pieces in a 6x6 grid for each black and white player consists of 4 rooks, 6 pawns, 1 king and 1 queen. With respect to rules of the game, in any grid the player wants to play there is always 1 king and 1 queen, so depending on a grid that player is playing the number of pawns and rooks increases.

The game can be played in following ways:

Human vs Human:

In this mode, both players are human controlled and they are prompted as and when their turn arrives. When a player's turn arrives, he/she is asked to input which piece is desired to be moved, and to which position. These inputs are then validated using the function possible_moves_of_player() and the game proceeds accordingly. In this mode, it depends on the skill set of players that which player wins.

AI vs Human:

In this mode, Player 1 is AI and Player 2 is human controlled. The AI determines it's best move using minimax function that evaluates a tree of depth five to determine the most optimal move. The human controlled player works in the standard way as described. When the player's turn arrives, he/she is asked to input which piece is desired to be moved, and to which position. These inputs are then validated using the function possible_moves_of_piece() and the game proceeds accordingly.

AI vs AI:

In this mode, both the players are AI controlled. Each player tries to make the best move using maximising its own score. This uses the minmax() function to find the most optimal move in it's turn.

The rules of the chess pieces are as follows:

King: A king can move one square in any direction (horizontally, vertically, or diagonally), unless the square is already occupied by a friendly piece.

Queen: The queen can move in any direction on a straight or diagonal path. The queen cannot "jump" over any piece on the board, so its movements are restricted to any direction of unoccupied squares. The queen can be used to capture any of your opponent's pieces on the board.

Rook: The rook moves horizontally or vertically, through any number of unoccupied squares (see diagram). The rook cannot jump over pieces. As with captures by other pieces, the rook captures by occupying the square on which the enemy piece sits.

Pawn: Pawn chess pieces can only directly forward one square, with two exceptions. First, Pawns can move directly forward two squares on their first move only. Secondly, Pawns can move diagonally forward when capturing an opponent's chess piece.

Scoring Rules:

Pawn: If one kills a pawn of an opposite player by any chess piece then they'll get 30 points.

Rook: If one kills a rook of an opposite player by any chess piece then they'll get 60 points.

Queen: If one kills a queen of an opposite player by any chess piece then they'll get 80 points.

King: If one kills a king of an opposite player by any chess piece then they'll get 100 points and would result in the end of the game, where the player who kills an opposition king would win.

How to Play:

1. Run the PlayChess.py file using the python compiler or from the command line.
2. Enter the dimensions of the board you want to play on. (4v4, 6v6, 8v8)
3. Choose the mode you want to play, i.e whether you want human vs human game, AI vs AI game or AI vs human game.
4. If you select the mode that involves AI, select the version of AI that you want to use for the game. (Tree-based AI(Minimax), Neural Network, Baseline AI). Then you'll see the board and all the moves that AI chooses from the list of moves.

5. If you select the mode involving human interaction, then a chess board would be displayed to you from the grid that you selected earlier with all the possible moves. Possible moves will be a list of matrix indexes, where you choose one move. (Note: The move that you'll select would be 0 indexed, i.e if you want to select the 6th move from the list then you enter 5 because all the moves displayed start at the index of zero.)
6. If you select AI vs Human mode, you make the move from possible moves and the selection would be reflected back on the grid that you selected at the start of the game. After your move, AI would select its move accordingly and the result would be reflected back on the grid.

Bibliography:

1. Minimax - Wikipedia
2. Wholesale Chess
3. GitHub - luweizhang/chess-ai: chess game and AI built with python. (in progress)
4. Minimax Algorithm in Game Theory | Set 4 (Alpha-Beta Pruning) - GeeksforGeeks
5. Chess - Wikipedia
6. Artificial neural network - Wikipedia