Experiment-1: Tic-Tac-Toe with Minimax

Based on Python Implementation and Documentation

Prof. Gaurav Parashar
Department of Computer Science & Engineering
KIET Group of Institutions, Ghaziabad

07/08/2025

Overview

Information

Problem Statement

Develop an AI to play Tic-Tac-Toe using the Minimax algorithm.

Implementation Details

Represent game as 3x3 board, apply recursive minimax logic with pruning.

Plan

This program is divided into two phases Game Setup with human vs human and human vs AI.

PHASE-1

Game Setup

- 3x3 Board using NumPy Array
- Player 1 Symbol: `X`
- Player 2 Symbol: `O`
- Goal: Place three of the same symbols in a row, column, or diagonal

Step 1: Initialization

- Create empty 3x3 board with '-' symbols
- Assign symbols: `p1s = 'X'`, `p2s = 'O'`

Step 2: Game Loop

- Loop for 9 turns (maximum in 3x3 grid)
- Even turn: Player 1 (`X`)
- Odd turn: Player 2 (`O`)

Step 3: Place Symbol

- Prompt player to input row and column (1-3)
- Validate input: in bounds and cell is empty
- Place symbol and display updated board

Step 4: Check Win

- Check if player symbol appears in:
  - Any row (check_rows)
  - Any column (check_cols)
  - Any diagonal (check_diagonals)
- If true, declare the winner and end game

Step 5: Draw Condition

- After 9 turns, if no player has won
- Check both players for win one last time
- If none, declare the result as Draw

Function: place(symbol)

- Input row and column from user
- Validate and place symbol
- Print updated board

Function: check_rows, check_cols, check_diagonals

- Each checks if symbol appears three times
- Returns `True` if win is detected

Function: won(symbol)

- Combines row, column, and diagonal checks
- Used after every move to check win

Conclusion

- A simple turn-based 2-player game
- Efficient input-validation and win-checking
- Demonstrates control flow, loops, and conditionals

Thank You

Questions?

PHASE-2

Python Code with Documentation I

```
1  import numpy
2  #My initial Setup of the Game
3  board = numpy.array([['-','-','-'],['-','-','-'],['-','-','-']]) #
       Board is 3x3
4  p1s='X' # Symbol for Player 1
5  p2s='O' # Symbol for Player 2
6  def place(symbol): # 3. or 7. Place the symbol on the board
7      print(numpy.matrix(board)) # 3.1 print the board
8      while(1): # 3.2 Take input until the inputs are correct
9          row = int(input('Enter row - 1 or 2 or 3: ')) # 3.3 Take
               row input
10         col = int(input('Enter Column - 1 or 2 or 3: ')) # 3.4 Take
                col input
11
12         if row>0 and row<4 and col>0 and col<4 and board[row-1][col
               -1]=='-': # 3.5 row and col must lie between 0 to 3 and
                the board should be empty
13             break # 3.6 if everything is correct then break the
                   loop
14         else: # 3.7 Otherwise ask user to input again
15             print("Invalid Input please enter again...")
```

Python Code with Documentation II

```
16      board[row-1][col-1]=symbol # 3.8 if row and cols are correct
            then placethe symbol at the desired palce
17      print(numpy.matrix(board)) # 3.9 print the board with the
            placed symbol
18
19  def check_rows(symbol):# 4.1a Check the symbol is in three
        consecutive rows or not
20      for r in range(3): # 4.1a.1 Check all three rows, start from 0
            until row 2
21          count=0 # 4.1a.2 set counter to 0, to test whether it is a
                winning move
22          for c in range(3): # 4.1a.3 Iterate through all the cols
                (0..2)
23              if board[r][c] == symbol: # 4.1a.4 if the symbol on the
                     board is the one that is supplied by user then
24                  count=count+1 # 4.1a.5 increase the counter
25          if count==3: # 4.1a.6 If the counter is 3 then it means it
                is a winning move by the player with the symbol
26              print(symbol, ' won') # 4.1a.7 Place the symbol X won
                    or O won
27              return True # 4.1a.8 Return True if the player has won
28      return False # 4.1a.9 Return Flase in all the other cases
```

Python Code with Documentation III

```
29
30 def check_cols(symbol): # 4.1b Check the symbol is in three
       consecutive cols or not. The logic is same first we start with
       col 0 and all rows
31     for c in range(3): # 4.1b.1 Check the symbol is in three
           consecutive cols or not
32         count=0 # 4.1b.2 set counter to 0, to test whether it
               is a winning move
33         for r in range(3): # 4.1b.3 Iterate through all the
               rows (0..2)
34             if board[r][c] == symbol: # 4.1b.4 if the symbol on
                   the board is the one that is supplied by user
                   then
35                 count=count+1 # 4.1b.5 increase the counter
36         if count==3: # 4.1b.6 If the counter is 3 then it means
               it is a winning move by the player with the symbol
37             print(symbol, ' won') # 4.1b.7 Place the symbol X
                   won or O won
38             return True # 4.1b.8 Return True if the player has
                   won
39     return False # 4.1b.9 Return Flase in all the other cases
40
```

Python Code with Documentation IV

```
41 def check_diagonals(symbol):  # 4.1c Check the symbol is in three
       diagonal places
42     if board[0][2]==board[1][1] and board[1][1]==board[2][0] and
           board[1][1]==symbol: # 4.1c.1 First winning diagonal move
43         print(symbol, " won") # 4.1c.2 Place the symbol X won or O
               won
44         return True # 4.1c.3 Return True if the player has won
45     if board[0][0]==board[1][1] and board[1][1]==board[2][2] and
           board[0][0]==symbol: # 4.1c.4 Second winning digonal move
46         print(symbol, " won")  # 4.1c.5 Place the symbol X won or O
                won
47         return True # 4.1c.6 Return True if the player has won
48     return False # 4.1c.7 Return Flase in all the other cases
49
50 def won(symbol): # 4 or 8. check the winning move
51     return check_rows(symbol) or check_cols(symbol) or
           check_diagonals(symbol) # 4.1 Winning move can be either
           three consecutive rows or cols or any diagonal
52     #these functions return True in case it is a winning move
           otherwise False
53
54 def play():
```

Python Code with Documentation V

```
55      for turn in range(9): # 1. Number of total turns as the size of
            board is 3x3
56          if turn%2==0:  # 2. All even chances 0,2,4 are of X's and
                odd chances 1,3,5 are of O's
57              print('X Trun') # 2.1 Print it is an X's Turn
58              place(p1s) # 3. Place the X's symbol on the board
59              if won(p1s): # 4. After placing the symbol check
                    whether this move is winning move or not
60                  break   # 5. If X's move is winning then come of
                        the the game
61              else: # 6. Else check if it is O's turn
62                  print('O Trun') # 6.1 Print is is an O's Turn
63                  place(p2s) # 7. Place the O's symbol on the  board
64                  if won(p2s): # 8. After placing the symbol check
                        whether this move is winning move or not
65                      break  # 9. If O's move is winning then come of
                            the the game
66      if not(won(p1s)) and not(won(p2s)): # 10. if there is no move
            that it is a winning move
67          print('Draw') # 11. then it is a draw
68
69  if __name__ == "__main__":
```

Python Code with Documentation VI

```
70      play()
```