AI LAB 1

TOY PROBLEM





Code-

tictactoe.py:

import math

import copy

X = "X"

O = "O"

EMPTY = None

def initial\_state():

"""

Returns starting state of the board.

"""

return [[EMPTY, EMPTY, EMPTY],

[EMPTY, EMPTY, EMPTY],

[EMPTY, EMPTY, EMPTY]]

def player(board):

"""

Returns player who has the next turn on a board.

"""

xCounter = 0

oCounter = 0

for i in range(0, len(board)):

for j in range(0, len(board[0])):

if board[i][j] == X:

xCounter += 1

elif board[i][j] == O:

oCounter += 1

if xCounter > oCounter:

return O

else:

return X

def actions(board):

"""

Returns set of all possible actions (i, j) available on the board.

"""

possibleActions = set()

for i in range(0, len(board)):

for j in range(0, len(board[0])):

if board[i][j] == EMPTY:

possibleActions.add((i, j))

return possibleActions

def result(board, action):

"""

Returns the board that results from making move (i, j) on the board.

"""

# Create new board, without modifying the original board received as input

result = copy.deepcopy(board)

result[action[0]][action[1]] = player(board)

return result

def winner(board):

"""

Returns the winner of the game, if there is one.

"""

# Check rows

if all(i == board[0][0] for i in board[0]):

return board[0][0]

elif all(i == board[1][0] for i in board[1]):

return board[1][0]

elif all(i == board[2][0] for i in board[2]):

return board[2][0]

# Check columns

elif board[0][0] == board[1][0] and board[1][0] == board[2][0]:

return board[0][0]

elif board[0][1] == board[1][1] and board[1][1] == board[2][1]:

return board[0][1]

elif board[0][2] == board[1][2] and board[1][2] == board[2][2]:

return board[0][2]

# Check diagonals

elif board[0][0] == board[1][1] and board[1][1] == board[2][2]:

return board[0][0]

elif board[0][2] == board[1][1] and board[1][1] == board[2][0]:

return board[0][2]

else:

return None

def terminal(board):

"""

Returns True if game is over, False otherwise.

"""

if winner(board) is not None or (not any(EMPTY in sublist for sublist in board) and winner(board) is None):

return True

else:

return False

#return True if winner(board) is not None or (not any(EMPTY in sublist for sublist in board) and winner(board) is None) else False # noqa E501

def utility(board):

"""

Returns 1 if X has won the game, -1 if O has won, 0 otherwise.

"""

if terminal(board):

if winner(board) == X:

return 1

elif winner(board) == O:

return -1

else:

return 0

# Check how to handle exception when a non terminal board is received.

def minimax(board):

"""

Returns the optimal action for the current player on the board.

"""

if terminal(board):

return None

else:

if player(board) == X:

value, move = max\_value(board)

return move

else:

value, move = min\_value(board)

return move

def max\_value(board):

if terminal(board):

return utility(board), None

v = float('-inf')

move = None

for action in actions(board):

# v = max(v, min\_value(result(board, action)))

aux, act = min\_value(result(board, action))

if aux > v:

v = aux

move = action

if v == 1:

return v, move

return v, move

def min\_value(board):

if terminal(board):

return utility(board), None

v = float('inf')

move = None

for action in actions(board):

# v = max(v, min\_value(result(board, action)))

aux, act = max\_value(result(board, action))

if aux < v:

v = aux

move = action

if v == -1:

return v, move

return v, move

Runner.py:

import pygame

import sys

import time

import tictactoe as ttt

pygame.init()

size = width, height = 600, 400

# Colors

black = (0, 0, 0)

white = (255, 255, 255)

screen = pygame.display.set\_mode(size)

mediumFont = pygame.font.Font("OpenSans-Regular.ttf", 28)

largeFont = pygame.font.Font("OpenSans-Regular.ttf", 40)

moveFont = pygame.font.Font("OpenSans-Regular.ttf", 60)

user = None

board = ttt.initial\_state()

ai\_turn = False

while True:

for event in pygame.event.get():

if event.type == pygame.QUIT:

sys.exit()

screen.fill(black)

# Let user choose a player.

if user is None:

# Draw title

title = largeFont.render("Play Tic-Tac-Toe", True, white)

titleRect = title.get\_rect()

titleRect.center = ((width / 2), 50)

screen.blit(title, titleRect)

# Draw buttons

playXButton = pygame.Rect((width / 8), (height / 2), width / 4, 50)

playX = mediumFont.render("Play as X", True, black)

playXRect = playX.get\_rect()

playXRect.center = playXButton.center

pygame.draw.rect(screen, white, playXButton)

screen.blit(playX, playXRect)

playOButton = pygame.Rect(5 \* (width / 8), (height / 2), width / 4, 50)

playO = mediumFont.render("Play as O", True, black)

playORect = playO.get\_rect()

playORect.center = playOButton.center

pygame.draw.rect(screen, white, playOButton)

screen.blit(playO, playORect)

# Check if button is clicked

click, \_, \_ = pygame.mouse.get\_pressed()

if click == 1:

mouse = pygame.mouse.get\_pos()

if playXButton.collidepoint(mouse):

time.sleep(0.2)

user = ttt.X

elif playOButton.collidepoint(mouse):

time.sleep(0.2)

user = ttt.O

else:

# Draw game board

tile\_size = 80

tile\_origin = (width / 2 - (1.5 \* tile\_size),

height / 2 - (1.5 \* tile\_size))

tiles = []

for i in range(3):

row = []

for j in range(3):

rect = pygame.Rect(

tile\_origin[0] + j \* tile\_size,

tile\_origin[1] + i \* tile\_size,

tile\_size, tile\_size

)

pygame.draw.rect(screen, white, rect, 3)

if board[i][j] != ttt.EMPTY:

move = moveFont.render(board[i][j], True, white)

moveRect = move.get\_rect()

moveRect.center = rect.center

screen.blit(move, moveRect)

row.append(rect)

tiles.append(row)

game\_over = ttt.terminal(board)

player = ttt.player(board)

# Show title

if game\_over:

winner = ttt.winner(board)

if winner is None:

title = f"Game Over: Tie."

else:

title = f"Game Over: {winner} wins."

elif user == player:

title = f"Play as {user}"

else:

title = f"Computer thinking..."

title = largeFont.render(title, True, white)

titleRect = title.get\_rect()

titleRect.center = ((width / 2), 30)

screen.blit(title, titleRect)

# Check for AI move

if user != player and not game\_over:

if ai\_turn:

time.sleep(0.5)

move = ttt.minimax(board)

board = ttt.result(board, move)

ai\_turn = False

else:

ai\_turn = True

# Check for a user move

click, \_, \_ = pygame.mouse.get\_pressed()

if click == 1 and user == player and not game\_over:

mouse = pygame.mouse.get\_pos()

for i in range(3):

for j in range(3):

if (board[i][j] == ttt.EMPTY and tiles[i][j].collidepoint(mouse)):

board = ttt.result(board, (i, j))

if game\_over:

againButton = pygame.Rect(width / 3, height - 65, width / 3, 50)

again = mediumFont.render("Play Again", True, black)

againRect = again.get\_rect()

againRect.center = againButton.center

pygame.draw.rect(screen, white, againButton)

screen.blit(again, againRect)

click, \_, \_ = pygame.mouse.get\_pressed()

if click == 1:

mouse = pygame.mouse.get\_pos()

if againButton.collidepoint(mouse):

time.sleep(0.2)

user = None

board = ttt.initial\_state()

ai\_turn = False

pygame.display.flip()