**Python Program based on Reinforcement Learning(Tictactoe Game)**

Code is already given, there will be a human move and computer move.

**Task 1:**

Declare a variable that decides the board layout (for example 3 implies 3 x 3 board) Make changes to the code such that the opponent (who plays 'O') is automated using a probability transition function.

**Details**

1. Model the state space and action space for Tictactoe game.
2. Create an arbitrary probability transition function
3. Make use of the above probability transition function to decide the move of the opponent(computer)

In this part, human move will be performed by yourself using the mouse itself. Only the opponent move is automated.

You can use any probability distribution function (uniform or normal distribution)to make choice of computer move.

**Task 2** (Computer move is automated using probability transition function in the same way like Task-1)

1. Create an arbitrary policy
2. Make changes to the code to follow that policy (This means the mouse click is no more required. The code will follow the policy to find human move)

Using some random choice human move will be performed.

**Task 3:**

Find an optimal policy for human move using RL algorithms: (Computer move is automated using probability transition function in the same way like Task-1)

1. Policy iteration algorithm
2. Value iteration algorithm
3. Q learning algorithm

All the modifications has to be done in this code(No new code should be written).

All the tasks has to be implemented as a separate python program.

**Code:**

|  |  |
| --- | --- |
|  | #importmodules |
|  | import pygame |
|  | from pygame.locals import \* |
|  |  |
|  | pygame.init() |
|  |  |
|  | screen\_height = 300 |
|  | screen\_width = 300 |
|  | line\_width = 6 |
|  | screen = pygame.display.set\_mode((screen\_width, screen\_height)) |
|  | pygame.display.set\_caption('Tic Tac Toe') |
|  |  |
|  | #define colours |
|  | red = (255, 0, 0) |
|  | green = (0, 255, 0) |
|  | blue = (0, 0, 255) |
|  |  |
|  | #define font |
|  | font = pygame.font.SysFont(None, 40) |
|  |  |
|  | #define variables |
|  | clicked = False |
|  | player = 1 |
|  | pos = (0,0) |
|  | markers = [] |
|  | game\_over = False |
|  | winner = 0 |
|  |  |
|  | #setup a rectangle for "Play Again" Option |
|  | again\_rect = Rect(screen\_width // 2 - 80, screen\_height // 2, 160, 50) |
|  |  |
|  | #create empty 3 x 3 list to represent the grid |
|  | for x in range (3): |
|  | row = [0] \* 3 |
|  | markers.append(row) |
|  |  |
|  |  |
|  |  |
|  | def draw\_board(): |
|  | bg = (255, 255, 210) |
|  | grid = (50, 50, 50) |
|  | screen.fill(bg) |
|  | for x in range(1,3): |
|  | pygame.draw.line(screen, grid, (0, 100 \* x), (screen\_width,100 \* x), line\_width) |
|  | pygame.draw.line(screen, grid, (100 \* x, 0), (100 \* x, screen\_height), line\_width) |
|  |  |
|  | def draw\_markers(): |
|  | x\_pos = 0 |
|  | for x in markers: |
|  | y\_pos = 0 |
|  | for y in x: |
|  | if y == 1: |
|  | pygame.draw.line(screen, red, (x\_pos \* 100 + 15, y\_pos \* 100 + 15), (x\_pos \* 100 + 85, y\_pos \* 100 + 85), line\_width) |
|  | pygame.draw.line(screen, red, (x\_pos \* 100 + 85, y\_pos \* 100 + 15), (x\_pos \* 100 + 15, y\_pos \* 100 + 85), line\_width) |
|  | if y == -1: |
|  | pygame.draw.circle(screen, green, (x\_pos \* 100 + 50, y\_pos \* 100 + 50), 38, line\_width) |
|  | y\_pos += 1 |
|  | x\_pos += 1 |
|  |  |
|  |  |
|  | def check\_game\_over(): |
|  | global game\_over |
|  | global winner |
|  |  |
|  | x\_pos = 0 |
|  | for x in markers: |
|  | #check columns |
|  | if sum(x) == 3: |
|  | winner = 1 |
|  | game\_over = True |
|  | if sum(x) == -3: |
|  | winner = 2 |
|  | game\_over = True |
|  | #check rows |
|  | if markers[0][x\_pos] + markers [1][x\_pos] + markers [2][x\_pos] == 3: |
|  | winner = 1 |
|  | game\_over = True |
|  | if markers[0][x\_pos] + markers [1][x\_pos] + markers [2][x\_pos] == -3: |
|  | winner = 2 |
|  | game\_over = True |
|  | x\_pos += 1 |
|  |  |
|  | #check cross |
|  | if markers[0][0] + markers[1][1] + markers [2][2] == 3 or markers[2][0] + markers[1][1] + markers [0][2] == 3: |
|  | winner = 1 |
|  | game\_over = True |
|  | if markers[0][0] + markers[1][1] + markers [2][2] == -3 or markers[2][0] + markers[1][1] + markers [0][2] == -3: |
|  | winner = 2 |
|  | game\_over = True |
|  |  |
|  | #check for tie |
|  | if game\_over == False: |
|  | tie = True |
|  | for row in markers: |
|  | for i in row: |
|  | if i == 0: |
|  | tie = False |
|  | #if it is a tie, then call game over and set winner to 0 (no one) |
|  | if tie == True: |
|  | game\_over = True |
|  | winner = 0 |
|  |  |
|  |  |
|  |  |
|  | def draw\_game\_over(winner): |
|  |  |
|  | if winner != 0: |
|  | end\_text = "Player " + str(winner) + " wins!" |
|  | elif winner == 0: |
|  | end\_text = "You have tied!" |
|  |  |
|  | end\_img = font.render(end\_text, True, blue) |
|  | pygame.draw.rect(screen, green, (screen\_width // 2 - 100, screen\_height // 2 - 60, 200, 50)) |
|  | screen.blit(end\_img, (screen\_width // 2 - 100, screen\_height // 2 - 50)) |
|  |  |
|  | again\_text = 'Play Again?' |
|  | again\_img = font.render(again\_text, True, blue) |
|  | pygame.draw.rect(screen, green, again\_rect) |
|  | screen.blit(again\_img, (screen\_width // 2 - 80, screen\_height // 2 + 10)) |
|  |  |
|  |  |
|  | #main loop |
|  | run = True |
|  | while run: |
|  |  |
|  | #draw board and markers first |
|  | draw\_board() |
|  | draw\_markers() |
|  |  |
|  | #handle events |
|  | for event in pygame.event.get(): |
|  | #handle game exit |
|  | if event.type == pygame.QUIT: |
|  | run = False |
|  | #run new game |
|  | if game\_over == False: |
|  | #check for mouseclick |
|  | if event.type == pygame.MOUSEBUTTONDOWN and clicked == False: |
|  | clicked = True |
|  | if event.type == pygame.MOUSEBUTTONUP and clicked == True: |
|  | clicked = False |
|  | pos = pygame.mouse.get\_pos() |
|  | cell\_x = pos[0] // 100 |
|  | cell\_y = pos[1] // 100 |
|  | if markers[cell\_x][cell\_y] == 0: |
|  | markers[cell\_x][cell\_y] = player |
|  | player \*= -1 |
|  | check\_game\_over() |
|  |  |
|  | #check if game has been won |
|  | if game\_over == True: |
|  | draw\_game\_over(winner) |
|  | #check for mouseclick to see if we clicked on Play Again |
|  | if event.type == pygame.MOUSEBUTTONDOWN and clicked == False: |
|  | clicked = True |
|  | if event.type == pygame.MOUSEBUTTONUP and clicked == True: |
|  | clicked = False |
|  | pos = pygame.mouse.get\_pos() |
|  | if again\_rect.collidepoint(pos): |
|  | #reset variables |
|  | game\_over = False |
|  | player = 1 |
|  | pos = (0,0) |
|  | markers = [] |
|  | winner = 0 |
|  | #create empty 3 x 3 list to represent the grid |
|  | for x in range (3): |
|  | row = [0] \* 3 |
|  | markers.append(row) |
|  |  |
|  | #update display |
|  | pygame.display.update() |
|  |  |
|  | pygame.quit() |