Homework 4 - Tic tac toe

The homework has been done with jupyter notebook. The whole homework is done using a numpy library. Matplotlib is used for visualization.

The notebook file contains the code and the results.

The homework flow is as follows

- Have a matrix of size = NxN
    - I have taken N as 3, but that is not necessary.
- Initialize the matrix with zeroes.
    - The values in the matrix will be represented by integers: 0, 1, 2.
        - 0: empty cell represented by ' '.
        - 1: cell occupied by symbol 'O'.
        - 2: cell occupied by symbol 'X'.
    - 'X' or 'O' can be chosen by the player at initialization step by providing a choice, defaults to 'x'
    - The other symbol will be chosen for the bot.
- Have a property of winner, initialized by None.
- Have a method to reset the board.
- Have a method which lets a user play by plotting a symbol of 'X' or 'O' only! anywhere within the matrix.
- Calculates if there is a winner after each symbol is plotted.
    - A win is defined by any row, column or diagonal being filled with the same symbol, with the symbol as the winner.
- If there is a winner, prints a message for the same.

How the program is constructed,

It should have a data-structure which represents a tic-tac-toe board.

It should support the 'X' or 'O' symbol addition into the board.

It should be able to declare a winner, in case there is one.

This means it should be able to calculate if any row, column or diagonal has same values throughout.

The constructor takes care of points (1), (2) and creates room for making (3) possible,

Create the board property, and set it up with 3x3 matrix full of zeroes, we are going forward with a matrix as the data-structure for representing the tic-tac-toe board.

Establish relationships between symbols to be used on boards with numbers for easier calculation of winner.

Assign the symbols to a player and a bot.

Initialize a winner to be None as no one has won yet.

After setting up our board, we need means to play. out of the three methods that you see, two are just wrappers around the first.

If a player (bot or human) enters their symbol to be placed somewhere in the board. Insert the corresponding integer instead. For 'X' insert a 2 and for 'O' insert a 1, otherwise let it be a blank cell.

have_same_val

- axis int: The direction along which operations are to be performed. Can have a value of 0 or 1 only.

    - 0 means row

    - 1 means column

- item_x int: The row of the matrix in which item has been inserted.

- item_y int: The column of the matrix in which the item has been inserted.

- item int: The latest integer inserted into the matrix at row-index = item_x, and column-index = item_y.

left_diagonal_has_same_values

- item_x int: The row of the matrix in which item has been inserted.

- item_y int: The column of the matrix in which the item has been inserted.

- item int: The latest integer inserted into the matrix at row-index = item_x, and column-index = item_y.

right_diagonal_has_same_values

- item_x int: The row of the matrix in which item has been inserted.

- item_y int: The column of the matrix in which the item has been inserted.

- item int: The latest integer inserted into the matrix at row-index = item_x, and column-index = item_y.

cols_have_same_values

Check if any of the columns have same values

- item_x int: The row of the matrix in which item has been inserted.

- item_y int: The column of the matrix in which the item has been inserted.

- item int: The latest integer inserted into the matrix at row-index = item_x, and column-index = item_y.

rows_have_same_values

Check if any of the rows have same values

- item_x int: The row of the matrix in which item has been inserted.

- item_y int: The column of the matrix in which the item has been inserted.

- item int: The latest integer inserted into the matrix at row-index = item_x, and column-index = item_y.

element_diagonal_has_same_value

Check if any of the diagonals have same values

- item_x int: The row of the matrix in which item has been inserted.

- item_y int: The column of the matrix in which the item has been inserted.

- item int: The latest integer inserted into the matrix at row-index = item_x, and column-index = item_y.

is_game_over

Check if the game is over, which is defined by a row, column or diagonal having

the same values as the latest inserted integer item

- item_x int: The row of the matrix in which item has been inserted.

- item_y int: The column of the matrix in which the item has been inserted.

- item int: The latest integer inserted into the matrix at row-index = item_x, and column-index = item_y.

is_winning_move

Check if the last move was a winning move, which is defined by a row, column or diagonal having

the same values as the latest inserted integer item

- item_x int: The row of the matrix in which item has been inserted.

- item_y int: The column of the matrix in which the item has been inserted.

- item int: The latest integer inserted into the matrix at row-index = item_x, and column-index = item_y.

is_stale

Checks if there is no vacant space on the board

player_move

The method which facilitates insertion of values into the board matrix.

- input_symbol: 'X' or 'O'

- item_x int: The row of the matrix in which item has been inserted.

- item_y int: The column of the matrix in which the item has been inserted.

Human user Play

The method exposed to a human user

facilitates insertion of values into the board matrix.

- input_symbol: 'X' or 'O'

- item_x int: The row of the matrix in which item has been inserted.

- item_y int: The column of the matrix in which the item has been inserted.

bot_play

The method exposed to a bot

facilitates insertion of values into the board matrix.

- input_symbol: 'X' or 'O'

- item_x int: The row of the matrix in which item has been inserted.

- item_y int: The column of the matrix in which the item has been inserted.

Agent

An agent is a problem solver.

It should perform actions like:

- plotting a symbol on the tic-tac-toe board if it is vacant.

- Remember which states are more profitable than the others.

- Explore better states

- Exploit for maximum profit

- exploration_rate: A floating point number < 1

which defines the agents probability to explore.

- learning_rate: Used for assessing the value of intermediate

states during temporal difference learning.

- discount_factor: The factor by which a reward must be reduced

to be passed on for intermediate states

get_serious

Quit exploring states and start exploiting

Use this if you want to play with the agent.

learn_by_temporal_difference

Implementation of the temporal difference formula.

set_state

Stores the action performed for a given state

on_reward

Assign rewards to actions performed on intermediate states.

select_move

Choose from exploration and exploitation.

explore_board

Find an empty cell from the board

exploit_board

Find the best action for the given state

optimize_bot

Punish or Reward the bot with respect to the agent that wins the game

A sample of Exploration and exploitation

```
epoch: 1
explore
(0, 2)

             |   | X
           -----------
             |   |
           -----------
             |   |

explore
(1, 0)

             |   | X
           -----------
            X|   |
           -----------
             |   |
```

```
explore
(2, 0)

     |   | X
-----------
 X |   |
-----------
 X |   |

explore
(0, 1)

     | X | X
-----------
 X |   |
-----------
 X |   |


explore
(2, 1)

     | X | X
-----------
 X |   |
-----------
 X | X |

explore
(0, 0)

 X | X | X
-----------
 X |   |
-----------
 X | X |

Winner is: X
----------------------------------------------------------------------------------------------------
epoch: 2
exploit
State rewards
[[  0.00000000e+00   0.00000000e+00   4.65661287e-72]
 [  0.00000000e+00   0.00000000e+00   0.00000000e+00]
 [  0.00000000e+00   0.00000000e+00   0.00000000e+00]]
(0, 2)

     |   | X
-----------
     |   |
-----------
     |   |
```

```
explore
(0, 1)

   | x | x
-----------
   |   |
-----------
   |   |

explore
(1, 0)

   | x | x
-----------
 x |   |
-----------
   |   |


explore
(2, 1)

   | x | x
-----------
 x |   |
-----------
   | x |

explore
(0, 0)

 x | x | x
-----------
 x |   |
-----------
   | x |

Winner is: x
```