Artificial Intelligence and Machine Learning Fundamentals

Activity 2: Teaching the Agent to Realize Situations When It Defends Against Losses

Follow these steps to complete the activity:

1. Create a function **player_can_win** such that it takes all moves from the board using the **all_moves_from_board** function and iterates over it using a variable **next_move**. On each iteration, it checks if the game can be won by the sign, then it return true else false.

```
def player_can_win(board, sign):
next_moves = all_moves_from_board(board, sign)
for next_move in next_moves:
if game_won_by(next_move) == sign:
return True
return False
```

2. We will extend the AI move such that it prefers making safe moves. A move is safe if the opponent cannot win the game in the next step.

```
def ai_move(board):
new_boards = all_moves_from_board(board, AI_SIGN)
for new_board in new_boards:
if game_won_by(new_board) == AI_SIGN:
return new_board
safe_moves = []
for new_board in new_boards:
if not player_can_win(new_board, OPPONENT_SIGN):
safe_moves.append(new_board)
return choice(safe_moves) if len(safe_moves) > 0
else \
new_boards[0]
```

3. You can test our new application. You will find the AI has made the correct move.
4. We will now place this logic in the state space generator and check how well the computer player is doing by generating all the possible games.

```
def all_moves_from_board( board, sign ):
```

5. We will now place this logic in the state space generator and check how well the computer player is doing by generating all the possible games.

```
def all_moves_from_board(board, sign):
move_list = []
for i, v in enumerate(board):
if v == EMPTY_SIGN:
new_board = board[:i] + sign + board[i+1:]
move_list.append(new_board)
if game_won_by(new_board) == AI_SIGN:
return [new_board]
if sign == AI_SIGN:
safe_moves = []
for move in move_list:
```

```
        if not player_can_win(move, OPPONENT_SIGN):
        safe_moves.append(move)
        return safe_moves if len(safe_moves) > 0 else \
        move_list[0:1]
        else:
        return move_list
```

6. Count the possibilities that as possible.

```
        count_possibilities()
```

The output is as follows:
step 0. Moves: 1
step 1. Moves: 9
step 2. Moves: 72
step 3. Moves: 504
step 4. Moves: 3024
step 5. Moves: 5197
step 6. Moves: 18606
step 7. Moves: 19592
step 8. Moves: 30936
First player wins: 20843
Second player wins: 962
Draw 20243
Total 42048

We are doing better than before. We not only got rid of almost 2/3 of possible games again, but most of the time, the AI player either wins or settles for a draw. Despite our effort to make the AI better, it can still lose in 962 ways. We will eliminate all these losses in the next activity.