Introduction to Artificial Intelligence

Lab 3 (Week 3) -Intelligent Agents: Part 1 - Agents and environments

2023 - 2024

18 February 2024

Objectives

- Using Python OOP to model and set up real-world problems (chess game as an example).
- Implement simple actions under certain circumstances.

Task 01: 8-PUZZLE GAME

The 8-puzzle is represented as a 3x3 grid with numbers 1 through 8 and an empty space denoted by 0. The aim is to place each number in its corresponding position, as shown in Figure 1.

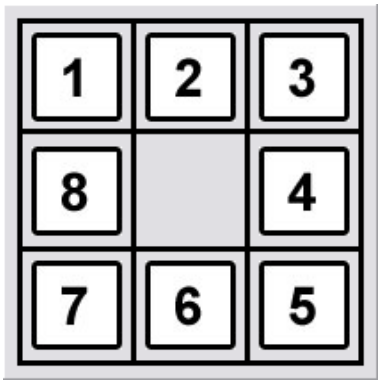

Figure 1: 8-Puzzle board

Step 1- Design the 8-Puzzle Environment:

As we mentioned, the 8-Puzzle Environment is depicted as a 3x3 grid. Each square on the puzzle is defined by a number between 0 and 9. The puzzle has an initial state, reflecting a randomly arranged set of squares, and a goal state, denoting the desired final configuration that a player aims to reach.

Step 2- Player representation

The main role of the player is to move the blank tile (square) to achieve the goal state. The player begins by identifying the position of the blank tile (0). Based on this position, the player decides on a move from the following possibilities: up, down, left, or right. The player must ensure that the chosen move does not move the blank tile outside the puzzle boundaries.

Step 3- Movement Rules:

- The player can move a numbered tile into the empty square, thus swapping the positions of the tile and the empty square.
- Legal moves are limited to adjacent squares, a tile can move up, down, left, or right into the empty square.
- The player cannot move a tile diagonally or skip over other tiles.
- Moves that would take a tile outside the puzzle boundaries are not allowed.

To do:

1. Using the object-oriented representation, define a square.
2. Using the object-oriented representation, define the Eight Puzzle (Environment).
3. Enhance your code by integrating a function that visually represents the current configuration of the Eight Puzzle. Please ensure that blank tiles are displayed as empty squares, while the other cells should be represented with their respective numbers.
4. Define a player in the Eight Puzzle game (Agent).
5. Implement the function that returns the possible moves based on the position of the blank tile.
6. Test your game with one player that moves the blank tile randomly (up to 20 moves) while displaying the puzzle at each iteration.
7. What did the agent do ?

Task 02: CHESS GAME

The chess game environment is a classic 8x8 board, where two players engage in a strategic battle using various chess pieces. Each piece has specific movements, and the ultimate goal is to checkmate the opponent's king. The environment includes the following components and features:

Step 1- Preparing the environment

1.1- Chessboard Representation:

As shown in Figure 2, the chessboard is a square grid of 8 `x` 8 cells. In which each cell may contain a chess piece(King, Queen, Rook, Knight, Bishop, Pawn) or remain empty.

Figure 2: Chess board

1.2- Chess pieces

Each specific chess piece is defined by a name(type): King(K), Queen(Q), Rook(R), Knight(K), Bishop(B), Pawn(P), and it has a specific color(White or Black). Once the chess piece is created, its color and name cannot be changed.

Step 2- Player representation

Each player in the game is defined by a name and the color of the chess pieces he/she/it wants to manage.

Step 3- Rules to move a chess piece

In this exercise, we focus on the two following rules:

- Each piece cannot move outside the boundaries of the chessboard.
- Each player has pieces of a distinct color. The player can move any piece regardless of its color(the player who has control over the white pieces is not allowed to move the back pieces).
- The player is not allowed to move a piece to a square occupied by a piece of the same color.

To do:

1. Using the object-oriented representation, define the chess piece.
2. Using the object-oriented representation, define the chessboard.
3. Enhance your code by incorporating a function that visually represents the current configuration of the chessboard. Please be mindful that white pieces should be displayed using uppercase letters, while black pieces should be represented in lowercase letters.
4. Define a player in the chess game.
5. Implement a function that returns all possible moves for a given piece. Consider the current state of the board, the position of the piece, and the rules for each type of piece.
6. Test your game with one player that moves the pieces randomly.

Extra-Tasks:

Task 3: An aware agent

This task involves implementing an aware agent represented by a 2D agent, `Env_Aware_Agent`, that is environment-aware and capable of sensing obstacles by visual observation. The environment is structured with obstacles denoted by '|' and flat surfaces represented by '_'.

Figure 3: An aware Agent Environment

The agent's behavior is designed to dynamically respond to the environmental features:

- When encountering an obstacle, the agent performs a jump, changing its position by 2.
- In the absence of obstacles, the agent moves forward one position at a time.
- The environment is constrained by a specified limit.

The `Environment` class serves to initialize the environment, randomly distributing obstacles and flat surfaces. The agent, implemented by `Env_Aware_Agent`, utilizes its 'move' method to navigate through the environment based on real-time observations of obstacles.

The task's core functionality is outlined in the initial response, where the agent dynamically decides between jumping and walking, adjusting its position accordingly. Throughout the simulation, the agent provides detailed feedback on its position, ensuring an interactive and responsive interaction within the defined environment.

Task 4: Update the CHESS GAME

Update the CHESS GAME environment to include the following rules:

- **King:** it moves one square in any direction. It cannot move into a square that is under attack.
- **Queen:** it moves any number of squares horizontally, vertically, or diagonally.
- **Rook:** it moves any number of squares horizontally or vertically.
- **Bishop:** it moves any number of squares diagonally.
- **Knight:**it moves in an L-shape: two squares in one direction and one square perpendicular.
- **Pawn:** Moves forward one square, but captures diagonally. In addition, on its first move, a pawn has the option to move forward two squares. And they capture differently than they move.