Homework 1 - CSCI 6650: Intelligent Agents

Padam Jung Thapa (2623560)

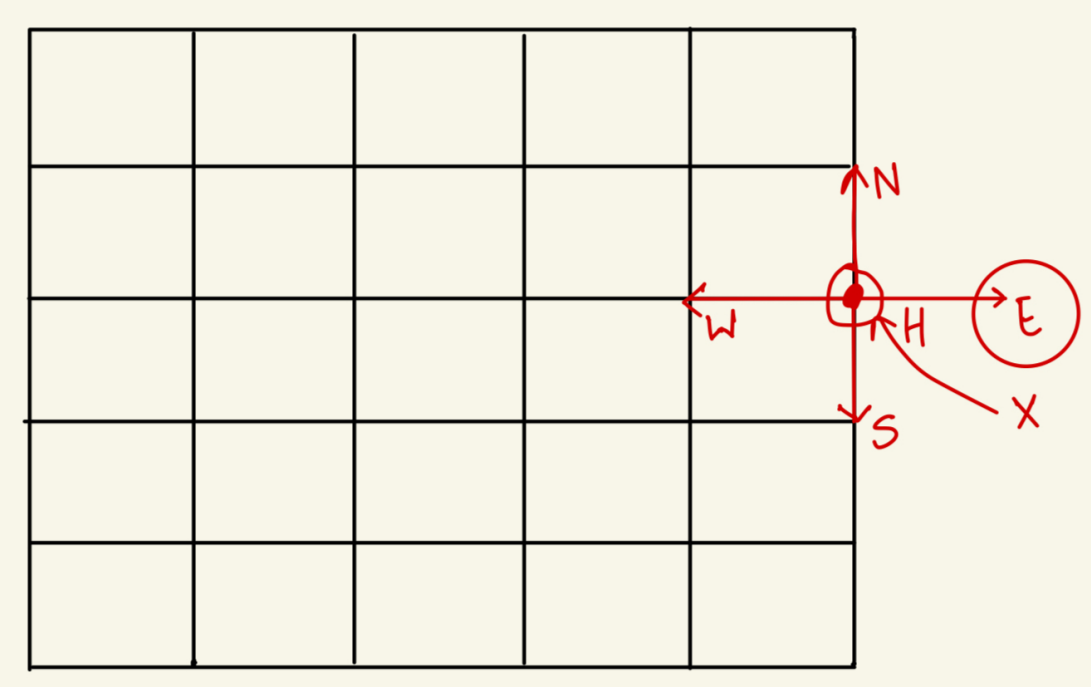
January 17, 2024

Problem Statement

Given a robot operating within a bounded rectangular grid world, the state of the robot at any time t is represented by a point x_t in the grid. The robot can perform actions to change its state, and each action is represented as a vector u_t in the grid world's vector space. The task is to determine the set of valid actions that will ensure the robot remains within the bounds of the grid when a new action is applied.

The state transition for the robot is defined by the function $f(x, u) = x + u$, where x is the current state and u is the action applied. The outcome of this function is a new state x_{t+1}. The objective is to verify whether the new state x_{t+1}, resulting from the application of action u_t to state x_t, remains inside the grid world. The focus is particularly on actions that have the potential to move the robot outside of the grid, such as an eastward move when the robot is at the eastern boundary.

The challenge is to use the concept of determinants in the solution. The determinant is used to calculate the area spanned by vectors in a matrix. In this scenario, it is necessary to determine if the point x_{t+1} falls outside the rectangular grid by computing determinants that involve the vectors defining the grid and the action in question.



Solution

Given a robot in a grid world, we represent the robot's current position as a point $\vec{x_t} = (x_t, y_t)$ and an action by the vector $\vec{u_t} = (u_x, u_y)$. The new position of the robot after taking the action is given by the state transition function:

$$\vec{x_{t+1}} = \vec{x_t} + \vec{u_t}$$

To determine if the new position $\vec{x_{t+1}}$ is outside the grid, we represent the grid by two orthogonal vectors $\vec{v_1}$ and $\vec{v_2}$, corresponding to the width and height of the grid, respectively. The corners of the grid can be represented by the origin $\vec{o}$ and the vectors $\vec{v_1}$, $\vec{v_2}$, and $\vec{v_1} + \vec{v_2}$.

The determinant of two vectors in the plane gives the area of the parallelogram formed by these vectors. For the robot to remain within the grid after the action, the new position must be expressible as a linear combination of $\vec{v_1}$ and $\vec{v_2}$ such that:

$$\vec{x_{t+1}} = \alpha\vec{v_1} + \beta\vec{v_2}$$

where α and β are scalars.
The robot's new position is within the grid if $0 \leq \alpha \leq 1$ and $0 \leq \beta \leq 1$. To find α and β, we can set up the following system of linear equations:

$$\begin{bmatrix} v_{1x} & v_{2x} \\ v_{1y} & v_{2y} \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} x_{t+1_x} \\ x_{t+1_y} \end{bmatrix}$$

The determinant of the matrix formed by the grid vectors $\vec{v_1}$ and $\vec{v_2}$ must not be zero for the system to have a unique solution. If the determinant is zero, the vectors are collinear and do not form a grid.
To check if the point $\vec{x_{t+1}}$ after the action is inside the grid, we can solve for scalars α and β such that:

$$\begin{bmatrix} v_{1x} & v_{2x} \\ v_{1y} & v_{2y} \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} x_{t+1_x} \\ x_{t+1_y} \end{bmatrix}$$

Solving for α and β gives us:

$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} v_{1x} & v_{2x} \\ v_{1y} & v_{2y} \end{bmatrix}^{-1} \begin{bmatrix} x_{t+1_x} \\ x_{t+1_y} \end{bmatrix}$$

If both α and β are within the range of $[0, 1]$, then the robot remains inside the grid. If either scalar is outside of this range, the robot's action would take it outside of the grid boundary.

Corresponding Python Script:

```python
import numpy as np
# Correcting the previous approach and calculating determinants
# Define a function to calculate the determinant indicating the position of the point relative to the sides of the rectangle
def check_robot_position_with_determinant(A, B, C, D, x_t, u_t):
    """
    Calculate the determinants to determine if the robot's next position x_t+1 will be outside the grid.

    Args:
    A, B, C, D: Coordinates of the rectangle's corners.
    x_t: Current position of the robot.
    u_t: Action taken by the robot.

    Returns:
    A tuple with two values:
        - The determinant for the triangle formed by the points A, x_t, and x_t+1.
        - The determinant for the triangle formed by the points C, x_t, and x_t+1.
    """
    # Calculate the next position of the robot
    x_t_plus_1 = x_t + u_t

    # Matrix for triangle A, x_t, and x_t+1
    matrix_A_xt_xt1 = np.array([
        [A[0], A[1], 1],
        [x_t[0], x_t[1], 1],
        [x_t_plus_1[0], x_t_plus_1[1], 1]
    ])

    # Matrix for triangle C, x_t, and x_t+1
    matrix_C_xt_xt1 = np.array([
        [C[0], C[1], 1],
        [x_t[0], x_t[1], 1],
        [x_t_plus_1[0], x_t_plus_1[1], 1]
    ])
```

```
34
35      # Calculate the determinants
36      det_A_xt_xt1 = np.linalg.det(matrix_A_xt_xt1)
37      det_C_xt_xt1 = np.linalg.det(matrix_C_xt_xt1)
38
39      return det_A_xt_xt1, det_C_xt_xt1
40
41  # Given points
42  A = np.array([0, 0])
43  B = np.array([5, 0])
44  C = np.array([5, 5])
45  D = np.array([0, 5])
46  x_t = np.array([5, 3])
47  u_t = np.array([1, 0])
48
49  # Calculate determinants
50  det_A_xt_xt1, det_C_xt_xt1 = check_robot_position_with_determinant(A, B, C, D, x_t, u_t)
51
52  det_A_xt_xt1, det_C_xt_xt1
```

- The calculated determinants are:
  - Determinant for triangle A, x_t, and x_{t+1}: -2.9999999999999996
  - Determinant for triangle C, x_t, and x_{t+1}: 2.000000000000001
- The function takes the following arguments:
  - A, B, C, D: The coordinates of the rectangle's corners.
  - x_t: The current position of the robot.
  - u_t: The action taken by the robot.
- The function first calculates the next position of the robot by adding the action to the current position. Then, it calculates two determinants: one for the triangle formed by points A, x_t, and $x_t + 1$, and another for the triangle formed by points C, x_t, and $x_t + 1$.
- The signs of these determinants indicate the position of $x_t + 1$ relative to the sides of the rectangle. A positive determinant means the point is on the same side as the corner, while a negative determinant means it's on the opposite side.
- If both determinants are positive, the next position is inside the grid. If either determinant is negative, the next position is outside the grid, and the robot's action is considered invalid.
- In the example provided, the robot is initially at (5, 3) and takes a move to the right (action [1, 0]). The determinants are calculated for triangles with corners A and C. If both determinants are positive, the next position will be inside the grid.
- The code defines a function 'check_robot_position_with_determinant' to determine if the robot's next position will be outside a rectangular grid. It calculates determinants of triangles formed by the robot's current and next positions with corners A and C of the rectangle. Positive determinants indicate the next position is on the same side as the corner, while negative determinants indicate it's on the opposite side. Both determinants being positive means the next position is inside the grid.

Interpretation:

Determinant for triangle A, x_t, and $x_t + 1$ is negative: This indicates that the next position $x_t + 1$ is on the opposite side of the line passing through A and x_t. This means the robot is moving towards the left side of the rectangle. Determinant for triangle C, x_t, and $x_t + 1$ is positive: This indicates that the next position $x_t + 1$ is on the same side as the line passing through C and x_t. This means the robot is not crossing the top boundary of the rectangle.

Conclusion:

Since the determinant for triangle A is negative, the robot's next position would be outside the grid if continued on its current trajectory. The robot's action to move right would need to be adjusted to prevent it from going outside the grid.

Other Notes:

- **What is the significance of the determinant of a matrix? How does it relate to the stretching or squashing of objects in linear transformations? What does the determination of our transformation matrix mean?**

  - The determinant of a matrix, particularly a square matrix, is a value that provides important information about the matrix and the linear transformation it represents. Key Points:
  1. Scale Factor: Indicates how much a linear transformation changes the area or volume. A larger determinant means more stretching, while a smaller one indicates compression.
  2. Orientation Change: The sign of the determinant shows if the transformation preserves (positive sign) or reverses (negative sign) the orientation of shapes.
  3. Singular or Non-Singular: A zero determinant means the matrix is singular (non-invertible), meaning it doesn't have an inverse. This corresponds to a transformation that squashes the space into a lower dimension (e.g., a plane being squashed into a line or a point), resulting in a loss of information. While a non-zero determinant indicates it is non-singular (invertible) matrix.
  4. Linear Independence: The determinant helps in determining whether the vectors that make up the matrix are linearly independent. A zero determinant indicates that the vectors are linearly dependent.

  - The determination of a transformation matrix, in the context of linear algebra and geometry, refers to finding the specific matrix that, when applied to a geometric object (like a point, vector, or shape), transforms it in a certain way. This transformation can include translation, rotation, scaling, shearing, or any combination of these. Also refers to how much area does the transformation will have.
  Here, The determinant of a 2x2 matrix can be interpreted geometrically as the area of a parallelogram.

- **What is State Estimation Problem?**
  - The State Estimation Problem involves determining the current state (like position or speed) of a system using imperfect or incomplete measurements. This is crucial in scenarios where direct measurement of the state is not possible or where the measurements are noisy, indirect, or uncertain. Techniques like Kalman Filters, Particle Filters, Bayesian Estimators are used to estimate the most probable state of the system based on available data, while accounting for uncertainties and measurement errors. This problem is commonly found in robotics, navigation systems, and other areas involving dynamic systems. For instance, in robotics, it's used for localization (determining a robot's position), in GPS for tracking positions, in aerospace for flight control, and in process control for monitoring industrial processes.