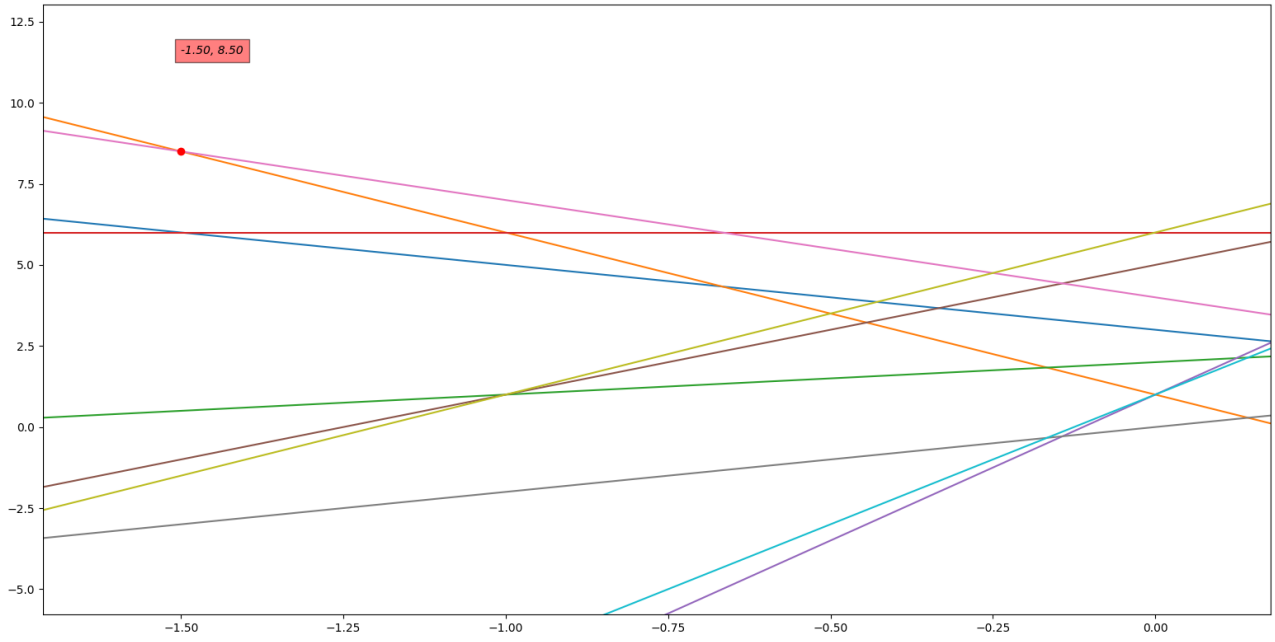
Mathematics of AI

Fereshteh Baradaran

Assignment #1



Introduction

In this assignment, we had to identify the minimum of the maximum values of a set of affine functions, commonly referred to as the Minimax problem. The main task was to analyze linear functions, represented as f(x) = ax + b, and determine the point where their maximum values reach the lowest possible point. To tackle this problem, I implement a **sweeping algorithm**, a technique renowned for its efficiency in computational geometry in processing linear functions.

Implementation

The implementation of this assignment is carried out in Python. The complete source code is available in a GitHub repository that can be accessed at the following link.

- countingSort.py:

  This implements the counting sort algorithm, which is used to sort the affine functions in *main.py*. Counting Sort, a linear time complexity sorting algorithm, is typically used for non-negative numbers. To handle both positive and negative values the original array is divided into two distinct arrays, one for positive and the other for negative values, and then applying Counting Sort to each separately. For negative values, a transformation to positive values is performed. This approach not only maintains the algorithm's linear O(n) complexity but also makes it ideal for efficiently sorting the slopes of affine functions.

- Main.py:

  This file contains the core logic for generating affine functions, sorting them, and implementing the algorithm to find the minimum of the maximum values. The input gathering section prompts the user to enter the number of affine functions and their coefficients. The visualization segment utilizes matplotlib to plot the affine functions for a visual understanding of their distribution and intersection points. The implementation of the Minimax problem solution is the key part of this file, where it calculates the minimum of the maximum of these affine functions. It'll be explained in detail in the following paragraphs.

  Starting with the Minimum Slope: The algorithm begins by considering the line with the minimum slope. This is based on the sorted order of the lines, where they are arranged from the smallest to the largest slope. Starting with the line of minimum slope is strategic because the intersection points with this line will often be critical in determining the minimum of the maximum values of these functions.

Finding the First Intersection and Min of Max: The algorithm then finds the first intersection of this line with the other lines. This intersection point is crucial as it potentially represents a change in the maximum value of the set of functions at different intervals. At this intersection point, the algorithm calculates f(x), the value of the function at this x-coordinate. This value is initially set as the minimum of the maximum values.

Updating the Current Line and Iterative Process: After finding the first intersection, the algorithm updates the current line to be the line that it intersected with. It then continues to find the next intersection of this new current line with the other lines. At each intersection, the algorithm evaluates f(x) for the new intersection point.

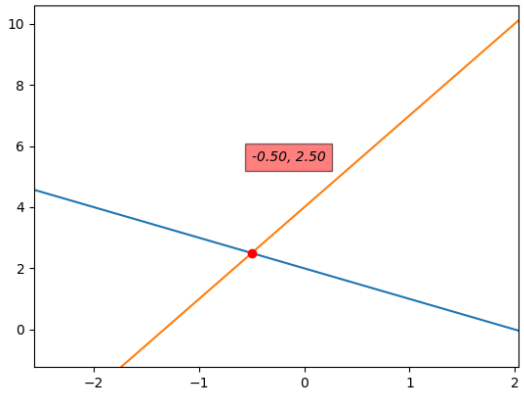
Comparing and Updating the Min of Max: If the value of f(x) at the new intersection point is lower than the current minimum of the maximum values, the min of max value is updated to this new value. This process continues until one of two conditions is met:

- The x-coordinate of the current line's intersection points with all other lines occurs before the x-coordinate of the minPoint.
- The min max value is less than the value of f(x) at the intersection point.

The final min of max value represents the minimum of the maximum values of all the affine functions at their intersection points.

Examples:

1. Affine Functions:
    a. -x + 2
    b. 3x + 4



2. Affine Functions:
    a. -2x + 3
    b. -5x + 1
    c. x + 2
    d. 6
    e. 9x + 1
    f. 4x + 5
    g. -3x + 4
    h. 2x
    i. 5x + 6
    j. 8x + 1