Math 364

Name: Akm Nabiul Haque

Report on Gravitational Wave Detection Using Linear Programming

Introduction:

Gravitational waves, predicted by Einstein's theory of relativity, are ripples in spacetime caused by some of the most cataclysmic events in the universe. The detection of these waves requires sensitive instruments and advanced optimization and analysis techniques.

Background:

With his general theory of relativity, Albert Einstein postulated the existence of gravitational waves in 1916. Designed to detect cosmic gravity waves and develop gravitational-wave observations as an astronomical instrument, the Laser Interferometer Gravity-Wave Observatory (LIGO) is a large-scale physics experiment and observatory. Gravitational waves are generated by dynamic systems of massive objects, such as merging black holes. When these waves reach Earth, they are incredibly faint, despite their powerful origins. The Laser Interferometer Gravitational-Wave Observatory (LIGO) detects these waves using a sophisticated arrangement of sensors. The challenge presented by LIGO involved classifying incoming sensor data to ascertain whether it indicates a gravitational wave event. This classification aids astronomers in deciding when to deploy telescopes for observation.

Model Description:

We applied a linear programming model to categorize sensor data into two categories: gravitational wave detection and non-detection, in order to overcome the difficulty presented by LIGO for gravitational wave detection. Based on training data, which consists of sensor readings labeled as either representing a gravitational wave (1) or not (0), the model builds a decision boundary that optimally divides these two groups.

Data Preparation:

20 observations with two sensor readings and a classification (0 or 1) make up the training data. For working with data arrays, the Python numpy package and scipy.optimize are used. The linear programming problem is solved using the linprog approach.

Model Specification:

Objective function:

Maximize the buffer δ, which represents the margin between classifications.

Constrains:

For observations labelled as detection (1): $x_{i,1}a + b + \delta \leq x_{i,2}$

For observations labelled as non-detection (0): $x_{i,1}a + b - \delta \geq x_{i,2}$

Variables:

Here,

a (coefficient affecting the influence of sensor 1)
b (bias term which adjusts the decision boundary vertically),
δ (buffer that ensures a margin between classifications).

Results and Analysis:

The Python package scipy.optimize import linprog was utilized to solve the linear programming model. The ideal values that were attained were:

1.22619048 is the slope (a).
-18.66428571 is the intercept (b).
The buffer (δ) is 19.97261905.

The decision boundary that optimizes the distance between the categorized locations is defined by these coefficients.

Detailed Classification for the Test Data Points:

1. $x_1 = 64.1$, $x_2 = 85.3$
   Predicted $x_2 = 60.1075$
   $x_2 + \delta = 60.1075 + 19.9726 = 80.0801$
   $x_2 - \delta = 60.1075 – 19.9726 = 40.1349$
   Actual $x_2 = 85.3$ is greater than 80.0801, classify as 0.

2. $x_1 = 59.3$, $x_2 = 26$
   Predicted $x_2 = 54.2627$
   $x_2 + \delta = 54.2627 + 19.9726 = 74.2353$
   $x_2 - \delta = 54.2627 – 19.9726 = 32.2901$
   Actual $x_2 = 26$ is less than 34.2901, classify as 1.

3. $x_1 = 84$, $x_2 = 50.9$
   Predicted $x_2 = 84.8074$
   $x_2 + \delta = 84.8074 + 19.9726 = 104.78$
   $x_2 - \delta = 84.8074 – 19.9726 = 64.8343$
   Actual $x_2 = 50.9$ is less than 64.8348, classify as 1.

4. $x_1 = 51.1$, $x_2 = 75.3$
   Predicted $x_2 = 44.4726$
   $x_2 + \delta = 44.4726 + 19.9726 = 64.4452$
   $x_2 - \delta = 44.4726 – 19.9726 = 24.5$
   Actual $x_2 = 75.3$ is greater than 64.4452, classify as 0.

5. $x_1 = 14.8$, $x_2 = 82$
   Predicted $x_2 = 0.1509$
   $x_2 + \delta = 0.1509 + 19.9726 = 20.1235$
   $x_2 - \delta = 0.1509 – 19.9726 = -19.8217$
   Actual $x_2 = 82$ is greater than 20.1235, classify as 0.

6. $x_1 = 68.3$, $x_2 = 78.7$
   Predicted $x_2 = 65.6062$
   $x_2 + \delta = 65.6062 + 19.9726 = 85.5788$
   $x_2 - \delta = 65.6062 – 19.9726 = 45.6336$
   Actual $x_2 = 78.7$ is greater than 85.5788, classify as 0.

7. $x_1 = 19.2$, $x_2 = 80.2$
   Predicted $x_2 = 5.2771$
   $x_2 + \delta = 5.2771 + 19.9726 = 25.2497$
   $x_2 - \delta = 5.2771 – 19.9726 = -14.6955$
   Actual $x_2 = 80.2$ is greater than 25.2497, classify as 0.

8. $x_1 = 19.1$, $x_2 = 8.2$

Predicted $x_2 = 5.0258$
$x_2 + \delta = 5.0258 + 19.9726 = 24.9984$
$x_2 - \delta = 5.0258 – 19.9726 = -14.9468$
Actual $x_2 = 8.2$ is less than 24.9984, classify as 0.

9.
$x_1 = 85.5$, $x_2 = 86.1$
Predicted $x_2 = 86.5776$
$x_2 + \delta = 86.5776 + 19.9726 = 106.55$
$x_2 - \delta = 586.5776 – 19.9726 = 66.6052$
Actual $x_2 = 86.1$ is less than 106.55, classify as 0.

10.
$x_1 = 87.7$, $x_2 = 47.2$
Predicted $x_2 = 88.7863$
$x_2 + \delta = 88.7863 + 19.9726 = 108.759$
$x_2 - \delta = 88.7863 – 19.9726 = 68.8137$
Actual $x_2 = 47.2$ is less than 68.8137, classify as 1.

11.
$x_1 = 27.4$, $x_2 = 0.7$
Predicted $x_2 = 15.5621$
$x_2 + \delta = 15.5621 + 19.9726 = 35.5347$
$x_2 - \delta = 15.5621 – 19.9726 = -4.4105$
Actual $x_2 = 0.7$ is less than 35.5347, classify as 0.

12.
$x_1 = 64.6$, $x_2 = 72$
Predicted $x_2 = 60.6998$
$x_2 + \delta = 60.6998 + 19.9726 = 80.6724$
$x_2 - \delta = 60.6998 – 19.9726 = 40.7272$
Actual $x_2 = 72$ is less than 80.6724, classify as 0.

13.
$x_1 = 83.6$, $x_2 = 28.2$
Predicted $x_2 = 84.3717$
$x_2 + \delta = 84.3717 + 19.9726 = 104.344$
$x_2 - \delta = 84.3717 – 19.9726 = 64.3992$
Actual $x_2 = 28.2$ is less than 64.3992, classify as 1.

14.
$x_1 = 21.5$, $x_2 = 63.9$
Predicted $x_2 = 8.8719$
$x_2 + \delta = 8.8719+ 19.9726 = 28.8445$
$x_2 - \delta = 8.8719 – 19.9726 = -11.1003$
Actual $x_2 = 63.9$ is greater than 28.8445, classify as 0.

15.

$x_1 = 80.5$, $x_2 = 96.4$
Predicted $x_2 = 81.4704$
$x_2 + \delta = 8.8719 + 19.9726 = 101.443$
$x_2 - \delta = 8.8719 – 19.9726 = 61.4976$
Actual $x_2 = 96.4$ is less than 101.443, classify as 0.

16.

$x_1 = 15.1$, $x_2 = 48.2$
Predicted $x_2 = 0.7223$
$x_2 + \delta = 0.7223 + 19.9726 = 20.6949$
$x_2 - \delta = 0.7223 – 19.9726 = -19.2497$
Actual $x_2 = 48.2$ is greater than 20.6949, classify as 0.

17.

$x_1 = 89.5$, $x_2 = 42.3$
Predicted $x_2 = 91.1784$
$x_2 + \delta = 91.1784 + 19.9726 = 111.151$
$x_2 - \delta = 91.1784 – 19.9726 = 71.2056$
Actual $x_2 = 42.3$ is less than 71.2056, classify as 1.

18.

$x_1 = 59$, $x_2 = 2.4$
Predicted $x_2 = 54.0091$
$x_2 + \delta = 54.0091 + 19.9726 = 73.9817$
$x_2 - \delta = 54.0091 – 19.9726 = 34.0365$
Actual $x_2 = 2.4$ is less than 34.0356, classify as 1.

19.

$x_1 = 67.3$, $x_2 = 91.9$
Predicted $x_2 = 64.2698$
$x_2 + \delta = 64.2698 + 19.9726 = 84.2424$
$x_2 - \delta = 64.2698 – 19.9726 = 44.2972$
Actual $x_2 = 91.9$ is greater than 84.2424, classify as 0.

20.

$x_1 = 82.7$, $x_2 = 88.6$
Predicted $x_2 = 83.6428$
$x_2 + \delta = 83.6428 + 19.9726 = 103.615$
$x_2 - \delta = 83.6428 – 19.9726 = 63.6704$
Actual $x_2 = 88,6$ is less than 103.615, classify as 0.

Codes and Outputs:

The linear program is set up with bounds to ensure δ remains non-negative, representing a meaningful buffer. The objective is to minimize the negative of
δ (as linprog performs minimization), effectively maximizing δ. This code is executed in Google Colab.

```
import numpy as np
from scipy.optimize import linprog


labels, features = training_data[:, 0], training_data[:, 1:]


A_ub, b_ub = [], []
for label, (x1, x2) in zip(labels, features):
    if label == 1:
        A_ub.append([x1, 1, 1])
        b_ub.append(x2)
    else:
        A_ub.append([-x1, -1, 1])
        b_ub.append(-x2)


c = [0, 0, -1]
bounds = [(None, None), (None, None), (0, None)]

# LP solution
result = linprog(c, A_ub=A_ub, b_ub=b_ub, bounds=bounds, method='highs')
print("Optimal values:", result.x)
```

```
Optimal values: [  1.22619048 -18.66428571  19.97261905]
```

Result and Analysis:

The model was able to determine the ideal values for a, b, and δ, with the findings demonstrating a substantial buffer that suggests a strong border between the classes. Reliable classifications are produced by this decision boundary, which is effectively created by maximizing the margin between detection and non-detection.

Conclusion:

Based on sensor data, this linear programming model provides a strong and effective tool for categorizing gravitational wave detections. The model is a great help for operations at LIGO and other sites because it can increase the separation margin while maintaining a high degree of forecast certainty.