477 HW3

Aditya Jadhav

September 16, 2023

1 Abstract

In this study, we explore the problem of line fitting using least squares methods in MATLAB. Two approaches, non-homogeneous and homogeneous least squares, are employed to fit lines to noisy data points. We compare these methods based on their Root Mean Square (RMS) errors under two different models: vertical distance and perpendicular distance. The paper also provides insight into why different least squares approaches result in varying levels of accuracy. The results are validated using real-world data set

2 Introduction

The purpose of this study is to investigate 2 approaches of line fitting through least squared error — non-homogeneous and homogeneous linear least square approach. Line fitting is often used in regression and many varieties of application to fit many data points across a single linear classifier .

We begin by using MATLAB to load the data sets, which consist of noisy data points assumed to lie on a line. After this initial step, the paper presents the mathematical formulas for non-homogeneous and homogeneous least squares methods and subsequently applies them to the data. The main metrics for comparison are the slope, intercept, and RMS errors calculated using both vertical and perpendicular distances to the fitted line.

We further discuss and try to find inssights into why the RMS errors might differ between the two approaches and under different models.

We then lastly touch upon image projection and how we can roughly determine if an image is in perspective or not by extending parallel lines in the image and checking for convergence.

All computations and visualizations were performed using MATLAB, and this report was compiled using LaTeX in OverLeaf. It took approximately 3 hours to conduct the experiments and 2 more hours to write this report, totaling 5 hours of work.

3 Experiments

3.1 Line fitting using Non Homogenous Least Squared Error

A non-homogenous system has the form [Ax = b] where A is a KxL matrix of coefficients, x is a $Lx1$ vector of unknowns and b is a $Kx1$ non-zero vector of constants. In the non-homogeneous least squares method, we define E as a measure of the "best" fit between our model and the data. Here, U and y are given by the problem and data at hand.

The least squares solution minimizes E, that is, we aim to find x such that $E(x)$ is minimized. Mathematically, this is expressed as:

$$E(x) = \|Ux - y\|_2^2$$

This is equivalent to:

$$E(x) = (Ux - y)^T(Ux - y)$$

The solution x that minimizes $E(x)$ is given by:

$$x = U^\dagger y$$

where $U^\dagger = (U^T U)^{-1}U^T$ is the pseudoinverse of U.

The equation $y = mx + b$ can be rewritten in matrix form as:

$$\begin{pmatrix} x & 1 \end{pmatrix} \begin{pmatrix} m \\ b \end{pmatrix} = y$$

In this representation, we form:

- A matrix U with rows $(x_i, 1)$
- A vector y with elements y_i
- A vector of unknowns $x = (m, b)$

We then use the formula $Ux \approx y$ to solve for the unknowns. For the datapoints in *line2data.txt*, we construct U, x and y to get the line of best fit. We get the intercept as the first value and slope as 2nd value when using \ to solve for $U^\dagger$.

We calculate RMS for vertical distance by just subtracting y value of the datapoint and value of y on the line with the same x value.

To compute the perpendicular distance between each data point and the fitted line, we employed the formula $d_i = ax_i + by_i - d$. In this formula, a and b represent the coefficients of the line equation, x_i and y_i are the coordinates of the data point, and d is the distance from the origin to the line.

It is important to note that we require the vector $[a, b]$ to be a unit vector for the formula to yield accurate perpendicular distances. In other words, $\sqrt{a^2 + b^2}$ should be equal to 1. To achieve this, we normalize the coefficients a and b

by dividing them by $\sqrt{a^2+b^2}$, thereby satisfying the unit vector constraint. In a non-homogeneous least squares fitting, we typically obtain a line in the form $y = mx + b$, where m is the slope and b is the y-intercept. To calculate perpendicular distances, it is often more convenient to work with the general line equation $ax + by + c = 0$.

To convert $y = mx + b$ into $ax + by + c = 0$, we can rearrange it as:

$$-mx + y - b = 0.$$

Here, $a = -m$, $b = 1$, and $c = -b$.

After obtaining a, b, and c, we normalize these coefficients to make $[a, b]$ a unit vector, i.e., we divide them by $\sqrt{a^2+b^2}$, thereby ensuring that the perpendicular distances d_i calculated using $d_i = |ax_i + by_i + c|$ are accurate. We get the following values -

Parameter	Value
Intercept	1.466518
Slope	-0.057488
RMS error	0.949231
RMS Perpendicular Error	0.947666

Table 1: Parameters and Errors for the Non-Homogeneous Line

3.2 Line fitting using Homogenous Least Squared Error

In the homogeneous least squares approach, the goal is to minimize the error contributions arising from the perpendicular distances between each data point and the fitted line. This is different from the non-homogeneous approach, which primarily focuses on minimizing vertical distances.

Mathematically, for each data point, the error contribution is the square of its perpendicular distance to the line. The equation for calculating this perpendicular distance is $d_i = ax_i + by_i + d$, where a, b, and d are the coefficients of the general line equation $ax + by + d = 0$. Here d = 0 as it is homogenous.

The error E in homogeneous line fitting can be represented as:

$$E = \sum (x_i - \bar{x}, y_i - \bar{y}) \cdot (a, b)^2 = ||Un||^2$$

where U is a matrix formed by:

$$U = \begin{pmatrix} x_1 - \bar{x} & y_1 - \bar{y} \\ \vdots & \vdots \\ x_n - \bar{x} & y_n - \bar{y} \end{pmatrix}$$

and n is a vector:

$$n = \begin{pmatrix} a \\ b \end{pmatrix}$$

To find the best-fitting line, we aim to solve $Un = 0$ in a least squares sense, with the constraint that $n = \sqrt{a^2 + b^2} = 1$. This constraint ensures that we are minimizing the perpendicular distance from each point to the fitted line. We aim to minimize the objective function, highlighted as follows:

$$\textbf{IE, minimize} \quad Ux^T Ux = x^T U^T Ux \quad \textbf{where} \quad x^T x = 1$$

The matrix $U^T U$ can be decomposed as $U^T U = VDV^T$.

In MATLAB we do this using - In Matlab, form Y = $U^T U$. Then use eig() to get the eigenvalues and eigenvectors of Y. x is the eigenvector corresponding to the smallest eigenvalue. In Matlab, the smallest eigenvalue is usually first, but it pays to check.

Here intercept will be 0 as it it passes through origin. We again calculate the 2 types of the RMS as we described in Non Homogenous line fitting with a slight difference in a, b and d values. The standard equation for a line in a 2D plane is given as $y = mx + b$, where m is the slope and b is the y-intercept. In the context of homogeneous line fitting, we aim to minimize the perpendicular distance to the line, which is described by the equation $d_i = ax_i + by_i - d$.

To align this with our given equation $y = mx + b$, we perform an algebraic transformation. We rewrite $y = mx + b$ as $y - mx = b$, which can be further rearranged into $mx - y = -b$. In homogenous setting b = 0.

Comparing this with $ax_i + by_i - d$, we find that $a = m$, $b = -1$.

Table 2: Results for Homogeneous Line Fitting

Parameter	**Value**
Slope	1.000000
RMS Error	1.477034
RMS Perpendicular Error	1.044421
Intercept	0

3.3 Comparing between the 2 approaches

In our experiment, we observed that the RMS error for the Homogeneous line fitting was 1.477034, whereas for the Non-Homogeneous line fitting, it was 0.949231. Additionally, the RMS error for perpendicular distances was 1.044421 for the Homogeneous line and 0.947666 for the Non-Homogeneous line. The Non-Homogeneous method yielded lower RMS errors, both in terms of the line fit and the perpendicular distances, suggesting that it may offer a more accurate fit for the given data. However, it is worth noting that the objectives of these two methods are different: the Homogeneous method aims to minimize the perpendicular distances, whereas the Non-Homogeneous method minimizes the vertical distances to the line. Depending on the specific requirements of the task, one method may be more appropriate than the other. The homogenous is used to get a line passing from origin which minimizes the perpendicular error

and vice versa for non homogenous. Both give a very well fitted line and I believe their usefulness depends on the distribution of datapoints.

Here is the plot of the lines gotten from homogenous and non-homogenous line fitting. Also we throw in the line we get from the REGRESSION function in MATLAB. WE see that the non-homogenous and REGRESSION function almost give us the same line.

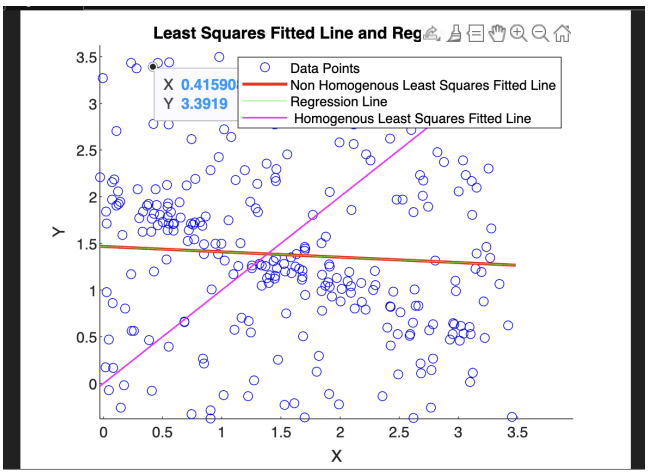

Figure 1: Distribution of datapoints on a XY plane and the output lines we get from homogenous and non homogenous line fitting. Note the overlap of the line we get from Non-homogenous line fitting and the one we get from the REGRESSION function.

3.4 Finding whether an object is in perspective using interpolation

When parallel lines in the image meet at a single point, it tells us that the perspective is correct, and the image was captured from a realistic viewpoint. This effect is often due to the camera's lens mimicking the way our eyes see the world in three dimensions.

The fact that different sets of parallel lines meet at different points is another confirmation of a correct perspective. This suggests that the lines are vanishing into the distance as they should in a three-dimensional space.

To figure out if the building image is in perspective, I drew lines over it as shown in Figure 2. I used two tests from class to check. The first test is to see if lines that should be parallel meet at a single point. The yellow and blue lines on the image meet this rule. But the red lines don't meet at a point because they are flat to the camera.

The second test is to see if different sets of parallel lines meet at different points. The yellow and blue lines do this too. They meet at what we call the horizon. So, based on these tests, the building image is mostly in perspective.

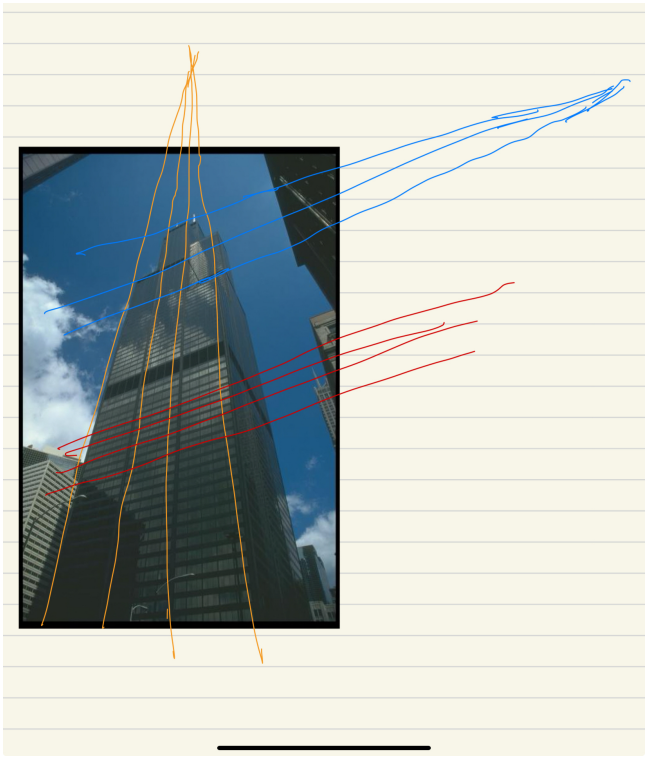

Figure 2: As we see here the vertical yellow lines seem to converge on a single spot. As do the blue horizontal lines which converge onto a single point when extended. But the red horizontal lines do not converge onto a single point. Probably because they are parallel to the camera plane.

3.5 Conclusion

We now have succesfully discussed and experimented many MATLAB functions to get output lines after doing homogenous and non homogenous line fitting usign least squared error. We have learnt about which type of RMS error gives us better suited to which implementation of the line fitting. We also know now about how to identify whether objects are in perspective using the extrapolation of parallel lines in an image and checking for convergence.

4 Bibliography

- https://www.overleaf.com/learn/latex/Matrices
- https://www.overleaf.com/learn/latex/Inserting_Images
- `https://www.overleaf.com/learn/latex/How_to_Write_a_Thesis_in_LaTeX_(Part_3)`

- https://www.overleaf.com/learn/latex/Learn_LaTeX_in_30_minutes
- matlab.mathworks.com