Theory and Implementation of linear regression

Mengyu Huang
Ningbo Xiaoshi High School

***Abstract*-Linear regression refers to the mathematical technique of fitting given data to a function of a certain type. It is best known for fitting straight lines. In this paper, we explain the theory behind linear regression and illustrate this technique with a real world data set. This data relates the earnings of a food truck and the population size of the city where the food truck sells its food.**

Keywords*-*Linear regression; Fitting straight lines; Food truck

I. Introduction

Assume we own a used car dealership that sells only cars of the same model and year so it is reasonable to assume that the price of each car depends only on the number of miles the car has. We just acquire a car with 55,000 miles on it and wonder the price at which we should sell it. Answering this question would be easy if we had a function $y = f(x)$ that gives us the selling price y as a function of the miles on the car x. We would simply replace x by 55,000 in the formula $y = f(x)$ to obtain the selling price y. However, we do not have the function $f(x)$. In fact, such a function may not exists. What we do have is our past experience. Assume we have sold 5 cars whose miles and selling prices are listed on Table 1. The question becomes: Given our past experience summarized on Table 1, at what price should we sell a car with 55,000 miles? Of course this questions may have many answers, after all, we are free to set any asking price, but Linear Regression [1], the subject of this article, is a technique that provides an answer.

The idea behind Linear Regression, as applied to the example of the above paragraph, can be summarized as follows: Since we do not know the relationship $y = f(x)$, in fact, such a relation may not exists, we approximate this relation by a linear relation $y = ax + b$. Linear Regression uses the data we have, Table 1, to compute a and b. The explanation of how to carry out this computation is the subject of this article. For our example of Table 1, we obtain $a = -0.05735$ and $b = 13{,}580$. Once a and b are computed, it is a matter of replacing x by 55,000 in the equation $y = ax + b$ to obtain the selling price from the Linear Regression technique. We obtain a selling price of $10,426.

Table 1: Miles and prices of the cars sold.

Miles	Price ($)
20,000	13,000
30,000	11,000
45,000	11,500
60,000	9,500
70,000	10,000

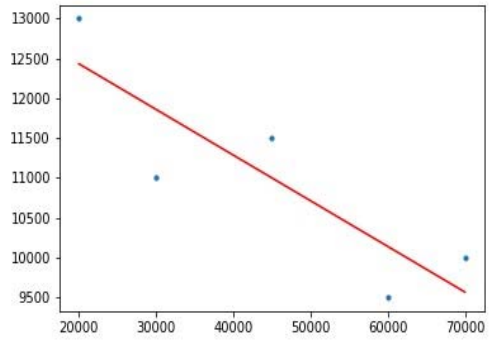

Figure 1: Scatter plot of the data in Table 1 in blue and plot of the line that best fits this data in red.

Figure 1 shows a scatter plot, in blue, of the data of Table 1. The horizontal axis is the miles and the vertical axis is the selling price. This figure also shows of plot of the line $y = ax + b$ in red. The reader will agree that this line approximates the data well. In fact, this line is the line that *best fits the data*. As we will see in this article, Linear Regression makes the meaning of *best fits the data* precise.

In Section 2 we state our problem in general terms. In Section 3 we explain how the slope a and y-intercept b of *the line $y = ax + b$ that best fits a given data set* are obtained from the data set. As learned in Section 3, the computation of the of a and b requires the solution of a linear systems of equations. We explain in Section 4 what a linear system of equation is. In Section 5 we explain what vectors and matrices are. In Section 6 we introduce the notion of augmented matrix of a linear system. In sections 7 and 8 we describe an algorithm to solve linear systems. This algorithm is called Gaussian Elimination. In Section 9 we show our implementation of the Gaussian Elimination Algorithm in the programming language Python.

Sections 10 to 14 are devoted to the derivation of the equations that the slope a and y-intercept b of the line that best fits the data set satisfy. We start in Section 10 by explaining the notion of distance between vectors. In Section 11 we discuss the minima of concave quadratic functions. In Section 12, we introduce the Data vector $\mathbf{y}$ and the Predicted vector $\bar{\mathbf{y}}$. In Section 13, we discuss the error we make when we approximate a data set by a straight line. In Section 14, we derive the equations that the slope a and y-intercept b of the line that best fits the data set satisfy. In sections 15 we describe how we implement the theory discussed thus far in the programming language Python. We provide a real world example in Section 16, and we finish with a small discussion in Section 17.

978-1-7281-9481-3/20/$31.00 ©2020 IEEE
DOI 10.1109/CVIDL51233.2020.00-99

II. Statement of the problem

We now state our problem in a general setting. Assume we are given a data set (x_1,y_1), (x_2,y_2),...,(x_n,y_n), where x_1, x_2,...,x_n, y_1, y_2,..., y_n are all known numbers. Each pair (x_i,y_i) (for $1 \leq i \leq n$) is called a point. Our goal is to find the numbers a and b such that the line $y = ax + b$ *best approximates* the given data. The meaning of *best approximates* will be made precise later in this paper.

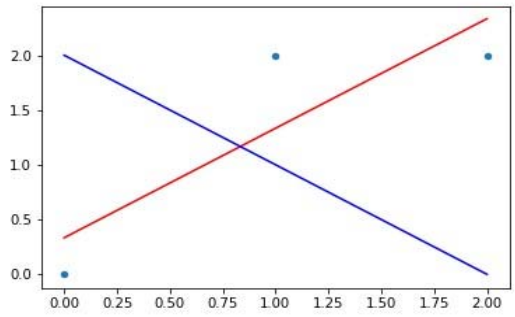

Figure 2: The blue dots are the scatter plot of the data set (0,0), (1,2) and (2,2). The blue line is the line $y = -x + 2$. The line that best fits this data is in red. It is the line $y = x + (1/3)$.

In our car sells example, the data, that is shown in Table 1, consists of the five points: (20,000,13,000), (30,000,11,000), (45,000,11,500), (60,000,9,500) and (70,000,10,000). Thus, x_1 = 20,000, x_2 = 30,000, x_3 = 45,000, x_4 = 60,000, x_5 = 70,000, y_1 = 13,000, y_2 = 11,000, y_3 = 11,500, y_4 = 9,500, y_5 = 10,000 and n = 5.

To illustrate the concepts we will introduce, from now on, we will consider a second example of a data set with only three points and smaller numbers. This data set is the set of points (0,0), (1,2) and (2,2). Figure 2 shows these three points in blue. It also shows two lines. The line in red is given by the equation $y = x + (1/3)$ and the line in blue is given by the equation $y = -x + 2$. Clearly, the red line approximates better this data. In fact, as we will later see, this line is the line that best approximates this data.

All our implementation will be in the programming language Python. We include our codes in this article. To avoid confusion, we enclose all our python codes in boxes. The code below creates, saves and displays Figure 2.

```
import matplotlib.pyplot as plt f1 =
plt.figure()
plt.scatter([0,1,2],[0,2,2],s = 20) plt.plot([0,2],[0
+ 1/3,2 + 1/3], 'r') plt.plot([0,2],[−1 ∗ 0 + 2,−1 ∗
2 + 2], 'b') f1.savefig('f1.png') plt.show()
```

III. Equations to find the slope and y-intercept of the line that best fit a given data set

Any straight line is given by an equation of the form $y = ax + b$, where a and b are numbers that are called the slope and y-intercept respectively. The slope a and the y-intercept b of the line that best fit the data $(x_1,y_1),(x_2,y_2),...,(x_n,y_n)$, are the solutions of the following system of two linear equations

$$(x_1^2 + x_2^2 + \ldots + x_n^2)a + (x_1 + x_2 + \ldots + x_n)b = x_1y_1 + x_2y_2 + \ldots + x_ny_n \quad (1)$$

$$(x_1 + x_2 + \ldots + x_n)a + nb = y_1 + y_2 + \ldots + y_n. \quad (2)$$

The reason why the a and b are the solutions to these equations will be explained later in this article.

We illustrate the use of these equations with the example of the data set being the three points: (0,0),(1,2),(2,2). In this case, x_1 = 0, x_2 = 1, x_3 = 2, y_1 = 0, y_2 = 2, y_3 = 2 and n = 3. The coefficients of the Equations (1) and (2) are

$$x21 + x22 + x23 = 5\ x_1 + x_2 + x_3 = 3\ x1y1 + x2y2 + x3y3 = 6\ y_1 + y_2 + y_3 = 4.$$

Thus, the System of Equations (1) and (2) becomes

$$5a + 3b = 6 \quad (3)$$

$$3a + 3b = 4. \quad (4)$$

The solution to this system is $a = 1$ and $b = 1/3$, which gives us the line $y = x+(1/3)$, the red line plotted in Figure 2. Note that to find the slope a and the y-intercept b we had to solve a system of two linear equations with two unknowns. The explanation of what a linear system of equations is and a method to find its solution, in the case where the number of equations equal the number of unknowns, are given in the next few sections.

IV. Systems of linear equations

Following standard practice, we denote by R the set of real numbers. The symbol $\in$ means belongs to. For example, "5 $\in$ R" reads "five belongs to the set of real numbers", which is the same as saying that 5 is a (real) number. As an other example, "$x \in$ R" means that x is a number. If we write $x,y \in$ R, it means that both x and y are numbers. A linear equation is an equation of the form

$$a_1x_1 + a_2x_2 + \ldots + a_nx_n = b,$$

where $a_1,a_2,...,a_n,x_1,x_2,...,x_n,b \in$ R. The numbers $a_1,...,a_n,b$ are known. a_i is called the coefficient multiplying x_i and b is called the right hand side. The numbers $x_1,x_2,...,x_n$ are the unknowns, also called variables. The goal is to find the set of numbers $x_1,...,x_n$ that satisfy the equation. For example,

$$3x_1 + 4x_2 = 1$$

is a linear equation. In this case, a_1 = 3, a_2 = 4 and b = 1. x_1 = −1 and x_2 = 1 is a solution of this equation because the equation is satisfied if x_1 is replaced by −1 and x_2 is replaced by 1.

A system of linear equations is a set of linear equations. Thus, it is of the form

$$a11x1 + a12x2 + ... + a1nxn = b1\ a21x1 + a22x2 + ... + a2nxn = b2$$
$$\text{.....} \quad (5)$$
$$ak1x1 + ak2x2 + ... + aknxn = bk.$$

where $a_{ij}, x_i, b_i \in \mathrm{R}$ for all i and j. The numbers a_{ij}, b_i are called coefficients.

The System (5) is a system with n variables and k equations. The goal is to find the set of numbers $x_1,...,x_n$ that satisfy all k equations simultaneously. For example,

$$3x_1 + 4x_2 = -1 \quad (6)\ x_1 - 2x_2 = 3$$

is a system of two linear equations with two unknowns. In this case, $a_{11} = 3$, $a_{12} = 4$, $a_{21} = 1$, $a_{22} = -2$, $b_1 = -1$ and $b_2 = 3$. $x_1 = 1$ and $x_2 = -1$ is a solution of the system because it is a solution to each equation.

V. Vectors and matrices

A vector is a list of n numbers inside parenthesis and separated by comas. Thus, if **x** is a vector, **x** is of the form

$$\mathbf{x} = (x_1, x_2, ... x_n),$$

where $x_1,...,x_n \in \mathrm{R}$. We say that n is the number of components of **x**. We also say that **x** is an $n-$vector. We say that x_i is the i^{th} component of **x**. We usually denote vectors with a lower case bold face letter and its components with the same letter, not in bold face, and a subscript. For example,

$$\mathbf{x} = (1, -2, 0)$$

is the three vector where $x_1 = 1$, $x_2 = -2$ and $x_3 = 0$. We use the standard notation R^n for the set of $n-$vectors.

A matrix is set of numbers ordered in a finite rectangular grid. Thus, if A is a matrix, A is of the form

$$a11 \quad a12\ a22 \quad ... \quad a1n,\ a2n\ ...,$$
$$a21\ A = ... \quad ... \quad ... \quad akn$$
$$ak1 \quad ak2 \quad ...$$

where $a_{ij} \in \mathrm{R}$ for all i and j. We say that a_{ij} are the entries or components of A. For the matrix in the above Equation, n is the number of columns of A and k is the number of rows of A. We say that A is a $k \times n$ matrix. We usually denote matrices with upper case letters and its components with the same letter but in lower case and two sub-indices. The first sub-index denotes the row of the entry and the second the column. For example,

$$A = \begin{bmatrix} 1 & 3 & 0 \\ -2 & 4 & -1 \end{bmatrix}$$

is a 2×3 matrix. In this case, $a_{11} = 1$, $a_{12} = 3$, $a_{13} = 0$, $a_{21} = -2$, $a_{22} = 4$ and $a_{23} = -1$. We also use the notation $\mathrm{R}^{k \times n}$ for the set of all $k \times n$ matrices.

VI. Augmented matrix of a linear system

The augmented matrix of the linear system (5) is

$$\left[\begin{array}{cccc|c} a_{11} & a_{12} & \dots & a_{1n} & b_1 \\ a_{21} & a_{22} & \dots & a_{2n} & b_2 \\ \vdots & \vdots & & \vdots & \vdots \\ a_{k1} & a_{k2} & \dots & a_{kn} & b_k \end{array}\right].$$

For example, the augmented matrix of the system (6) is

$$\left[\begin{array}{cc|c} 3 & 4 & -1 \\ 1 & -2 & 3 \end{array}\right].$$

Note that we display a vertical line in augmented matrices. This is not essential, but it helps us remember that the matrices come from a linear systems of equations, and the columns to the right of the vertical line are form by the right hand side of the equations.

VII. Elementary operations on augmented matrices

An elementary operation on an augmented matrix is one of the following operations:

1. Multiplying a row by a non-zero number.
2. Adding to a row a multiple of an other row.
3. Swapping two rows.

Elementary operations do not change the set of solutions. For example, if we multiply the first row of the augmented matrix of the system (6) by 3, we get

$$\left[\begin{array}{cc|c} 9 & 12 & -3 \\ 1 & -2 & 3 \end{array}\right]. \quad (7)$$

The system (6) has the same set of solutions as the system with augmented matrix (7).

VIII. Gaussian elimination

We restrict our focus to systems of n equations with n unknowns in the case when the system has exactly one solution. We will describe an algorithm, called Gaussian elimination, to find that solution. The algorithm proceeds as follows:

1. If $n = 1$, we have only one equation with one unknown. More precisely, our system is simply the equation $a_{11}x_1 = b_1$, whose augmented matrix is $[a_{11}\ |b_1]$. Thus, the solution is $x_1 = b_1/a_{11}$ and we terminate the algorithm (note that we are assuming $a_{11}\ 6= 0$).

   If $n > 1$, continue with Step 2.

2. Find a row whose first entry value is larger than or equal to, in absolute value, the first entries of the other rows.

3. Swap the row found in Step 2 step with the first row.

4. Divide the *new* first row by its first entry.

5. For $2 \leq k \leq n$, subtract from the k^{th} row the first row times the first element of the k^{th} row. After this step, the augmented matrix has the form

$$\left[\begin{array}{cccc|c} 1 & \star & \dots & \star & \star \\ 0 & \star & \dots & \star & \star \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \star & \dots & \star & \star \end{array}\right],$$

where? simply means that any number could be in that location.

6. Apply recursively the algorithm to solve the system that results from removing the first rowand first column of the augmented matrix above. This gives the values of $x_2,x_3,\dots,x_n$.

7. Find the value x_1 using the first equation of the the augmented matrix above and the values for $x_2,x_3,\dots,x_n$ found in the 6th step.

To illustrate the algorithm described above, we now solve the system

$$x_1 + x_3 = -1 \; -2x_1 + x_2 = 1 \; x_2 - 2x_3 = 0.$$

We first construct the augment matrix

$$\left[\begin{array}{ccc|c} 1 & 0 & 1 & -1 \\ -2 & 1 & 0 & 1 \\ 0 & 1 & -2 & 0 \end{array}\right]. \tag{8}$$

In this case $n = 3 > 1$, so we move to Step 2.

The Row 2 is the row whose first entry value is largest in absolute value.

We move to Step 3.

We swap Row 1 and Row 2

$$\left[\begin{array}{ccc|c} -2 & 1 & 0 & 1 \\ 1 & 0 & 1 & -1 \\ 0 & 1 & -2 & 0 \end{array}\right]$$

and move to Step 4.

We divide Row 1 of this new matrix by −2 to get

$$\left[\begin{array}{ccc|c} 1 & -1/2 & 0 & -1/2 \\ 1 & 0 & 1 & -1 \\ 0 & 1 & -2 & 0 \end{array}\right]$$

and move to Step 5.

To Row 2 (of the last matrix), we subtract 1 times Row 1

$$\left[\begin{array}{ccc|c} 1 & -1/2 & 0 & -1/2 \\ 0 & 1/2 & 1 & -1/2 \\ 0 & 1 & -2 & 0 \end{array}\right] \tag{9}$$

and move to Step 6.

We now apply the algorithm recursively to the system that results from removing the first row and first column of this last matrix

$$\left[\begin{array}{cc|c} 1/2 & 1 & -1/2 \\ 1 & -2 & 0 \end{array}\right].$$

Now $n = 2 > 1$. The second row is the row whose first entry value is largest in absolute value. Thus, we swap Row 1 and Row 2

$$\left[\begin{array}{cc|c} 1 & -2 & 0 \\ 1/2 & 1 & -1/2 \end{array}\right].$$

To Row 2 we subtract 1/2 times Row 1

$$\left[\begin{array}{cc|c} 1 & -2 & 0 \\ 0 & 2 & -1/2 \end{array}\right] \tag{10}$$

We apply the algorithm recursively to the system that results from removing the first row and first column, i.e.

$$\left[\begin{array}{c|c} 2 & -1/2 \end{array}\right].$$

Now $n = 1$ and we can immediately get that $x_3 = (-1/2)/2 = -1/4$. Next, we use the first row in (10) to get that $x_2 = 0 - (-2)x_3 = 0 - (-2)(-1/4) = -1/2$. Finally, we use the first row in (9) to get that $x_1 = -1/2 - (-1/2)x_2 = -1/2 - (-1/2)(-1/2) = -3/4$.

IX. Implementation of Gaussian elimination in Python

Vectors are represented with lists in Python. The vector **x** is represented with the list

$$x = [x_1,x_2,\dots,x_n] = [x[0],x[1],\dots,x[n-1]].$$

Note that in Python the index of the first element of a list is 0, not 1.

Matrices are represented with lists of lists. Each list is a row. In general,

$$A = [[a_{11},a_{12},\dots,a_{1n}],[a_{21},a_{22},\dots,a_{2n}],\dots,[a_{k1},a_{k2},\dots,a_{kn}]].$$

Thus, $A[0] = [a_{11},a_{12},\dots,a_{1n}] = [A[0][0],A[0][1],\dots,A[0][n-1]]$, $A[1] = [a_{21},a_{22},\dots,a_{2n}] = [A[1][0],A[1][1],\dots,A[1][n-1]]$, ..., $A[k-1] = [a_{k1},a_{k2},\dots,a_{kn}] = [A[k-1][0],A[k-1][1],\dots,A[k-1][n-1]]$. For example, the augmented matrix (8) is represented in Python as the list of lists $A = [[1,0,1,-1],[-2,1,0,1],[0,1,-2,0]]$.

In this Section we show the code of a Python function that we call ”gaussian”, that implements the Gaussian Elimination algorithm in Python. The input of ”gaussian” is the augmented matrix of the system and thus, a list of lists. This function” gaussian” works only in the case when the system has only one solution. Solutions of systems are vectors, and since we represent vectors with lists in Python, our function” gaussian” returns a list.

Our function ”gaussian” will make use of the Python function defined in the code below, that we call ’maxrow’, that takes as input a matrix A (i.e. a list of lists), and returns the index of the row whose first entry is largest in absolute value. For example, maxrow([[1,0,1,−1],[−2,1,0,1],[0,1,−2,0]]) returns the integer 1.

```
def maxrow(A): n
    = len(A) i = 0
    x = abs(A[0][0]) j = 1
    while j < n : if
    abs(A[j][0]) > x : i = j
                x = abs(A[j][0])
        j = j + 1
    return i
```

In the box below we display the function" gaussian".

```
def gaussian(A): n
    = len(A) if n
    == 1 :
        return [A[0][1]/A[0][0]]
    i = maxrow(A)
    A[0],A[i] = A[i],A[0]
    j = 0
    x = A[0][0] while j
    <= n :
        A[0][j] = A[0][j]/x j = j
        + 1
    k = 1 while k <
    n : x = A[k][0]
        j = 0 while j
      <= n :
            A[k][j] = A[k][j] − x ∗ A[0][j] j = j
            + 1
        k= k + 1 B =
      [] i = 1 while i
      < n :
        B.append(A[i][1 :]) i =
        i + 1
    y = gaussian(B) x =
    A[0][n] i = 0 while i
    < len(y) :
        x = x − A[0][i + 1] ∗ y[i] i = i
        + 1
    return [x] + y
```

As an example, we tested our code with the system with augmented matrix (8). In fact, gaussian([[1,0,1,−1],[−2,1,0,1],[0,1,−2,0]]) returns the vector [−0.75,−0.5,−0.25], which is the right solution, $x_1 = -3/4$, $x_2 = -1/2$, $x_3 = -1/4$.

X. Distances between vectors

Note that 2-vectors are identified with points in the plane, as is illustrated in Figures 1 and 2. Thus, it makes sense to talk about the distance between two 2-vectors, $\mathbf{x} = (x_1,x_2)$ and $\mathbf{y} = (y_1,y_2)$. This distance can be found using Pythagorean Theorem, and is given by the formula $\sqrt{(y_1 - x_1)^2 + (y_2 - x_2)^2}$. For example, the distance between the vectors (1,3) and (2,2) is $\sqrt{(2-1)^2 + (2-3)^2} = 2$.

The notion of distance between vectors can be extended to vectors with any number of components. More precisely, the distance between the n-vectors $\mathbf{x} = (x_1,x_2,...,x_n)$ and $\mathbf{y} = (y_1,y_2,...,y_n)$ is $\sqrt{(y_1 - x_1)^2 + (y_2 - x_2) + ... + (y_n - x_n)^2}$. For example, the distance between the vectors $(1\sqrt{},3,4)$ and $(2,2,-1)$ is $\sqrt{(2-1)^2 + (2-3)^2 + (-1-4)^2} = 27$.

XI. Concave quadratic functions and their minima

A quadratic function is a function $f(x)$ of the form $f(x) = ax^2 + bx + c$, where a, b and c are known numbers, and x is the independent variable. For example, $f(x) = 3x^2-5x+2$ is a quadratic function, and in this case, $a = 3$, $b = -5$ and $c = 2$.

The quadratic function $f(x) = ax^2 + bx + c$ is said to be concave if $a > 0$. In our derivation of Equations (1) and (2), we will use the fact that the minimun of a quadratic concave function of the form $f(x) = ax^2 + bx + c$ is attained at the value

$$x = -\frac{b}{2a.} \tag{11}$$

The validity of the statement in the last paragraph can be easily understood as follows. We first pull a as a common factor and then complete squares to get that

$$f(x) = ax^2 + bx + c = a\left(x^2 + \frac{b}{a}x + \frac{c}{a}\right) = a\left(x + \frac{b}{2a}\right)^2 + c - \frac{b^2}{4a} = a\left(x + \frac{b}{2a}\right)^2 + \frac{(4ac - b^2)}{4a}.$$

Since the second term of the right hand side, $(4ac-b^2)/(4a)$, is constant, and first term, $(x+b/(2a))^2$, is non-negative and becomes zero for $x = -b/(2a)$, we see that, in fact, the minimun of $f(x)$ is attained for the value of x given in Equation (11). In the example of $f(x) = 3x^2 - 5x + 2$, the minimun is attained at $x = 5/6$. This is illustrated in Figure 3, where we graph the function $y = 3x^2 - 5x + 2$ in red, and with a solid blue circle, we show where the point $(5/6, 3(5/6)^2 - 4)$, where the minimun of the function occurs.

Figure 3: Graph of the function $y = 3x^2 - 5x + 2$ in red, and minimum point $(5/6, 3(5/6)^2 - 4)$, in blue.

XII. The Data vector y and the Predicted vector $\bar{y}$

The second components of our given data form an n-vector, that we denote with **y** and we call the Data vector **y**, i.e. $\mathbf{y} = (y_1, y_2, \ldots, y_n)$. A straight line $f(x) = ax + b$ that we consider as approximation of our data, gives us, for each x_i, predicted values of the second component

$\bar{y}_i = ax_i + b$. We denote the vector with components as $\bar{y}_i$ as $\bar{\mathbf{y}}$, and we call it the Predicted vector $\bar{\mathbf{y}}$. i.e. $\bar{\mathbf{y}} = (ax_1 + b, ax_2 + b, \ldots, ax_n + b)$.

For example, if the given data were (0,0),(1,2),(2,2), the Data vector would be $\mathbf{y} = (0,2,2)$ and, for the straight line $f(x) = x + 1/3$, the Predicted vector would be $\bar{\mathbf{y}} = (0+1/3, 1+1/3, 2+1/3) = (1/3, 4/3, 7/3)$.

XIII. Error in the prediction

The closer the Predicted vector $\bar{\mathbf{y}}$ to the Data vector **y**, the better the approximation is. Thus, we call the square of the distance between these two vectors the error. We denote this error by J. More precisely,

$$J = (ax_1 + b - y_1)^2 + (ax_2 + b - y_2)^2 + \ldots + (ax_n + b - y_n)^2. \quad (12)$$

Our goal is to find the values of a and b that minimize the error J. This gives precise meaning to the phrase *the line that best fits the data*.

XIV. Minimizing the error

To find the values of a and b that minimize the error J, we first regard b as a fixed number, and a as a variable. Note that $(ax_i + b - y_i)^2 = x_i^2 a^2 + 2x_i(b - y_i)a + (b - y_i)^2$. Thus, adding over all values of i we get that

$$J = \left(x_1^2 + x_2^2 + \ldots + x_n^2\right) a^2 + 2\left[x_1(b - y_1) + x_2(b - y_2) + \ldots + x_n(b - y_n)\right] a + \\ + \left[(b - y_1)^2 + (b - y_2)^2 + \ldots + (b - y_n)^2\right].$$

This is a quadratic function of a. Thus, from equation (11), we see that the minimun of the error J is attained for the value of a equal to

$$a = -\frac{2\left[x_1(b - y_1) + x_2(b - y_2) + \ldots + x_n(b - y_n)\right]}{2\left(x_1^2 + x_2^2 + \ldots + x_n^2\right)}.$$

Once the terms are rearrange, it can be easily seen that the last equation is in fact Equation (1).

The validity of Equation (2) is shown similarly. More precisely, we now regard a as a fixed number and b as the variable. We note that $(ax_i + b - y_i)^2 = b^2 + 2(ax_i - y_i)b + (ax_i - y_i)^2$ and add these expressions for all i to get

$$J = nb^2 + 2\left[(ax_1 - y_1) + (ax_2 - y_2) + \ldots + (ax_n - y_n)\right] b + \\ + \left[(ax_1 - y_1)^2 + (ax_2 - y_2)^2 + \ldots + (a_n - y_n)^2\right].$$

Applying Equation (11) again, we obtain that the minimun of the error J is attained for the value of b equal to

$$b = -\frac{2\left[(ax_1 - y_1) + (ax_2 - y_2) + \ldots + (ax_n - y_n)\right]}{2n}$$

Once the terms are rearrange in the last equation, we obtain Equation (2).

XV. Finding the line that best fits the data with Python

Below is the Python code that computes the slope a and the y-intercept b of the line that best fits the data. This code requires several functions that we created. The first function, that we called 'sum lis', takes as input a vector (list of numbers), and gives as output the sum of the components of the vector. For example, sum lis([0,2,2]) returns the number 4.

```
def sum lis(z) : s
    = 0 for a in
    z :
        s = s + a
    return s
```

The next function, that we call 'sum sqr lis', takes as input a vector (list of numbers), and gives as output the sum of the squares of the components of the vector. For example, sum lis([0,2,2]) returns the number 8.

```
def sum sqr lis(z): r
    = 0 for a in z :
        r = r + a * *2
    return r
```

The inner product of two vectors with the same number of components $\mathbf{x} = (x_1,x_2,...,x_n)$ and $\mathbf{y} = (y_1,y_2,...,y_n)$ is denoted by $\mathbf{x} \cdot \mathbf{y}$ and is the sum of the products of the components of the vectors. More precisely, $\mathbf{x}\cdot\mathbf{y} = x_1y_1+x_2y_2+...+x_ny_n$. The next function, that we call 'mul add', takes as input two vectors (list of numbers), and gives as output their inner product. For example, mul add([0,2,2],[1,−1,3]) returns the number 4.

```
def mul add(u,w): r =
    0
    n = len(u) i = 0 while i <
    n : r = r + u[i] * w[i] i =
    i + 1
    return r
```

Recall that our data is a set of n vectors with two components, $(x_1,y_1),(x_2,y_2),...,(x_n,y_n)$. From these n vectors, we can construct two vectors, the vector $\mathbf{x} = (x_1,x_2,...,x_n)$, and the vector $\mathbf{y} = (y_1,y_2,...,y_n)$. The vector $\mathbf{y}$ is the vector that we have called the Data vector $\mathbf{y}$. The next function, that we call equ, takes as input the vectors $\mathbf{x}$ and $\mathbf{y}$, and gives as output the augmented matrix of the system of equations (1) and (2), the equations that will need to be solved to find the line that best fits the data. Note that the function 'equ' uses functions that we have previously defined.

```
def equ(x,y):
    return [[sum sqr lis(x),sum lis(x),mul
    add(x,y)],[sum lis(x),len(x),sum lis(y)]]
```

For example, if our data is the set of vectors (0,0), (1,2), (2,2), the vectors $\mathbf{x}$ and $\mathbf{y}$ are $\mathbf{x} = (0,1,2)$ and $\mathbf{y} = (0,2,2)$ and the equ([0,1,2],[0,2,2]) returns [[5,3,6],[3,3,4]], which agrees with the system of equations (3) and (4) that we previously found for this example when we work it out by hand. Recall that we represent matrices, and in particular augmented matrices, as list, where each element of the list is a list itself that is a row of the matrix.

Finally, the function 'slo int', whose code is displayed below, takes as input the data vectors $\mathbf{x}$ and $\mathbf{y}$, as described for the function 'equ', and returns two numbers, a, b, which are the slope and the y-intercept of the line that best fits the data.

```
def slo int(x,y): r =
    gaussian(equ(x,
    y))
    return r[0],r[1]
```

For example, when applied to the data (0,0), (1,2), (2,2), which gives the $\mathbf{x} = (0,1,2)$ and $\mathbf{y} = (0,2,2)$, we have that slo int([0,1,2],[0,2,2]) returns the numbers 1,0.3333333, which really is 1,1/3, which means that the line that best fits this data is $y = x + 1/3$. This agrees with the line we found solving by hand the system of equation (3) and (4).

XVI. Applications to a real world data set

We now apply the techniques we have described to analyze an example with several data points. Our example is in a file that we called data.txt. Each line in this file contains two numbers. The first number is the population of a city in 10,000s. Thus, if the first number of a line if 7, it means that the city corresponding to that line has a population of 70,000. The second number in each line contains the profit of a food truck over a period of time T in 10,000 dollars in the corresponding city. For example, if a line contains the two numbers 7, 8.5, it means that the city has a population of 70,000 and the food truck made a profit of $85,000 over a period of time T

Below we show the lines of code we wrote to read the file data.txt. These lines of code create two lists called $\mathbf{x}$ and $\mathbf{y}$. These are our data vectors. More precisely, each element of $\mathbf{x}$ is the population

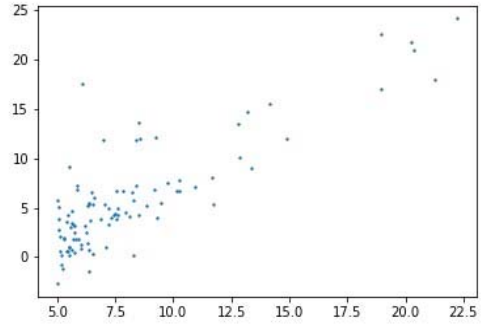

Figure 4: Scatter plot of the data in the file data.txt.

of a city in 10,000s, and each element of $\mathbf{y}$ is the profit of the food truck in the corresponding city over a period of time T in $10,000s.

```
with open('data.txt') as f:
    a0 = f.read()
a1  =  a0.split('\n')
a1.pop() x = [] y = []
for a2 in a1:
    c = a2.split(',')
    x.append(float(c[0]))
    y.append(float(c[1]))
```

The next few lines of code creates a scatter plot of the data

```
import matplotlib.pyplot as plt f2 = plt.figure() plt.scatter(x, y, s=2)
plt.show() f2.savefig('f2.png')
```

After running the lines of code in the box above, we display and save in a file called f2.png Figure 4. This is a figure with the data points. The horizontal axis is the *x*-axis and the vertical axis is the *y*-axis.

We now apply our code to find the line that best fits a given data to our example. This is simply done with the lines of code below.

```
a,b = slo int(x,y)
```

The line that best fits this data is $y = ax+b$. In this application we obtain $y = 1.1930336441895983x-$

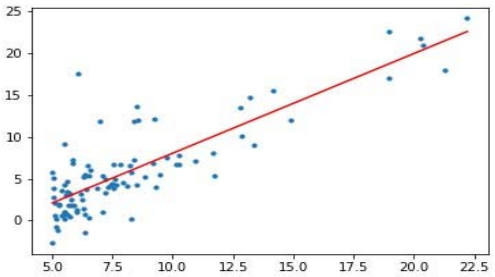

Figure 5: Scatter plot of the data in the file data.txt, in blue, and plot of the line that best fits this data.

3.895780878311899. This line, together with a scatter plot of the data, is displayed in Figure 5.

XVII. Conclusions

Linear Regression is a widely used technique used in many branches of science and technology. It is a core topic in Machine Learning and Data Science, two very popular fields that have found a wide range of applications. This article is a self-contained description of Linear Regression, the required Linear Algebra, and the required Python for its implementation.

References

[1] Douglas C Montgomery, Elizabeth A Peck, and G Geoffrey Vining. *Introduction to linear regression analysis*, volume 821. John Wiley & Sons, 2012.

[2] Czepiel, Scott A. *Maximum likelihood estimation of logistic regression models: theory and implementation*.Conference Proceedings, 2002.

[3] Yan, Xin and Su, Xiaogang.. *Linear regression analysis: theory and computing.*World Scientific, 2009