From EECS 216 (Sophomore Signals & Systems). You will use this in a few HW sets. See later warning.

Linear Regression or Least Squares Fit of Functions to Data

Prof. J.W. Grizzle, Univ. of Michigan

We discuss the process of fitting functions to data, which is usually called *regression*. A typical problem goes as follows. We are data as shown in Table I, which is also plotted in Figure 1. It is clear that the data do NOT lie exactly on a straight line. How can we **approximately** fit a straight line to the data? In particular, how can we do the fit in such a way as to **minimize the error**, in a given sense?

TABLE I
DATA FOR OUR FIRST LINEAR REGRESSION.

i	x_i	y_i
1	1	4
2	2	8
3	4	10
4	5	12
5	7	18

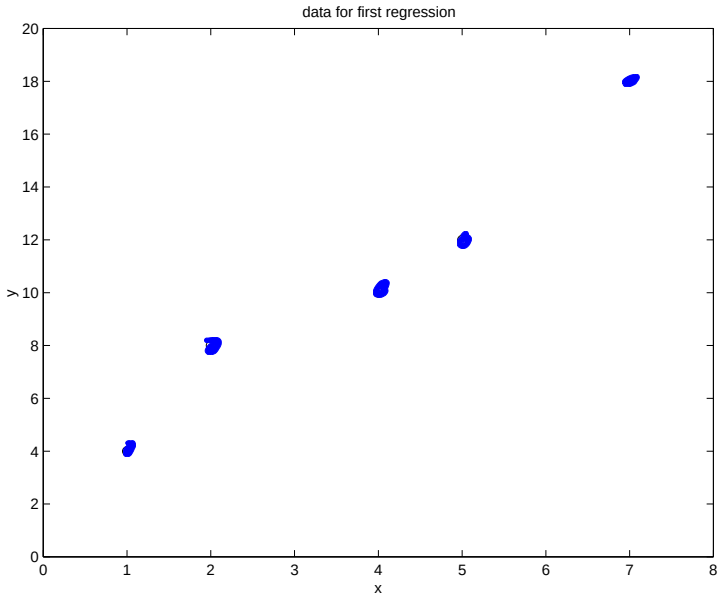

Fig. 1. Plot of data in Table 1.

Let's suppose that we want to fit a linear model

$$y = mx + b.$$

For $1 \leq i \leq N$, define

$$\hat{y}_i = mx_i + b,$$

where N is the number of data points (five in the case of Table I). Define the i-th error term to be

$$e_i = y_i - \hat{y}_i$$

and the total squared error to be

$$E_{tot} = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2.$$

Note that since $\hat{y}_i$ depends on m and b, so does E_{tot}.

We select the coefficients m and b in the model so as to minimize E_{tot} as a function of m and b. This is called a **Least Squared Error** fit, or a **Least Squares** fit.

Using basic calculus, we choose m and b so that

$$\frac{\partial E_{tot}}{\partial m} = 0$$
$$\frac{\partial E_{tot}}{\partial b} = 0$$

This yields two equations in the two unknowns. Depending on the particular problem , deriving these equations is more or less tedious. One way to do this is to grind out the partial derivatives, and produce the equations for m and b. We will use a more sophisticated but EQUIVALENT method that is MUCH EASIER to use in practice.

Write out the equations $y_i = mx_i + b$, $i = 1, \cdots, N$ in matrix form. That is, note that

$$y_i = mx_i + b = \begin{bmatrix} x_i & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix}$$

Now, do this for $i = 1, \cdots, N$

$$\underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}}_{Y} = \underbrace{\begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_N & 1 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} m \\ b \end{bmatrix}}_{\alpha}$$

in order to arrive at the **(over determined)** equation $Y = A\alpha$. For our example, the various quantities are

$$Y = \begin{bmatrix} 4 \\ 8 \\ 10 \\ 12 \\ 18 \end{bmatrix} \qquad A = \begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 4 & 1 \\ 5 & 1 \\ 7 & 1 \end{bmatrix} \qquad \alpha = \begin{bmatrix} m \\ b \end{bmatrix}$$

Theorem (to be proven in lecture): If $\det(A^T A) \neq 0$, then the least squares solution to $Y = A\alpha$ is given by

$$\hat{\alpha} = (A^T A)^{-1} A^T Y$$

That is

$$\hat{\alpha} = \underset{\alpha \in \mathbb{R}^2}{\operatorname{argmin}} \; ||Y - A\alpha||^2$$

where $|| \cdot ||^2$ is the sum of the squares of the elements of the vector.

Remark: This will be a special case of the *Normal Equations*!

On the Robotics PhD Qualifying Exam, you must be able to derive this result from the Normal Equations.

Computations in Matlab

```
x=[1     2     4     5     7]'

x =

     1
     2
     4
     5
     7

 Y=[ 4     8     10     12     18]'

Y =

     4
     8
    10
    12
```

```
    18

 A=[x,ones(5,1)]

A =

     1     1
     2     1
     4     1
     5     1
     7     1

 det(A'*A)

ans =

   114

 theta_hat=inv(A'*A)*A'*Y

theta_hat =

    2.1228
    2.3333

 m=theta_hat(1)

m =

    2.1228

 b=theta_hat(2)

b =

    2.3333

 plot(x,Y,'o',x,m*x+b)
 axis([0 8 0 20])
 gtext('y = mx + b')
 xlabel('x')
 ylabel('y')
```

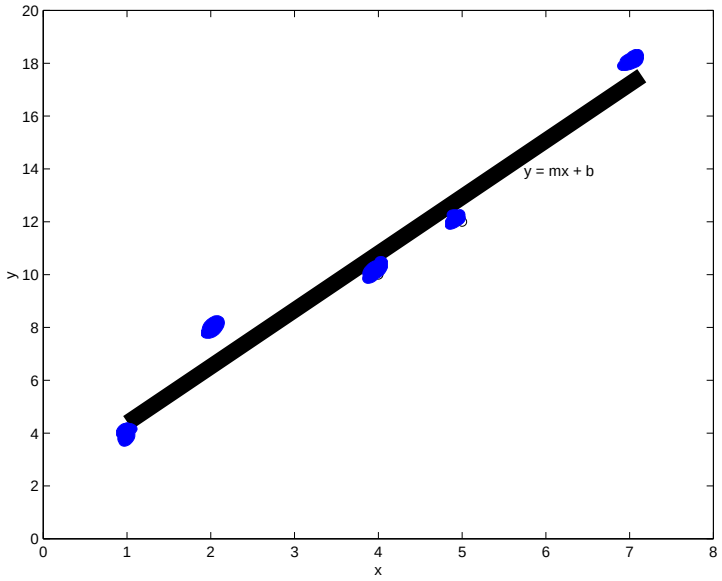

Fig. 2. Plot of data from Table 1 along with the least squares fit.

Now, all of this can be applied for any **model** that depends linearly on its unknown coefficients. For example, consider the data of Table II, which is plotted in Figure 3.

TABLE II
DATA FOR OUR SECOND LINEAR REGRESSION.

i	x_i	y_i
1	0	1.0
2	0.25	1.0
3	0.5	1.5
4	0.75	2.0
5	1.0	3.0
6	1.25	4.25
7	1.5	5.5
8	1.75	7.0
9	2.0	10.0

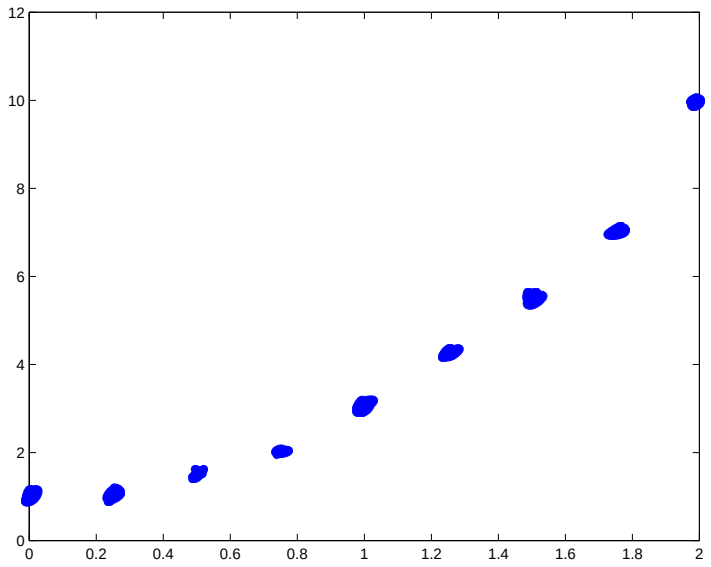

Fig. 3. Plot of data in Table 2.

Let's choose a model of the form

$$y = c_0 + c_1 x + c_2 x^2,$$

where here x^2 does mean x to power 2 (squared). Note that even though the model is nonlinear in x, it is linear in the unknown coefficients c_0, c_1, c_2. This is what is important!!! Just as before, define $\hat{y}_i = c_0 + c_1 x_i + c_2 x_i^2$, the i-th error

term to be

$$e_i = y_i - \hat{y}_i$$

and the total squared error to be

$$E_{tot} = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2.$$

To find the coefficients $c_0,\ c_1,\ c_2$ that minimize E_{tot}, we write out the model in matrix form:

$$y_i = c_0 + c_1 x_i + c_2 (x_i)^2 = \begin{bmatrix} 1 & x_i & (x_i)^2 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \end{bmatrix}.$$

Doing this for $i = 1, \cdots, N$ yields

$$\underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}}_{Y} = \underbrace{\begin{bmatrix} 1 & x_1 & (x_1)^2 \\ 1 & x_2 & (x_2)^2 \\ \vdots & \vdots & \\ 1 & x_N & (x_N)^2 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} c_0 \\ c_1 \\ c_2 \end{bmatrix}}_{\alpha}$$

which gives us the equation $Y = A\alpha$. For our second example, the various quantities are

$$Y = \begin{bmatrix} 1.0 \\ 1.0 \\ 1.5 \\ 2.0 \\ 3.0 \\ 4.25 \\ 5.5 \\ 7.0 \\ 10.0 \end{bmatrix} \qquad A = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0.25 & 0.0625 \\ 1 & 0.5 & 0.25 \\ 1 & 0.75 & 0.5625 \\ 1 & 1.0 & 1.0 \\ 1 & 1.25 & 1.5625 \\ 1 & 1.5 & 2.25 \\ 1 & 1.75 & 3.0625 \\ 1 & 2.0 & 4.0 \end{bmatrix} \qquad \alpha = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \end{bmatrix}$$

Computations in Matlab

```
 x=[0:.25:2]';
 Y=[ 1 1 1.5 2 3 4.25 5.5 7 10]';
 A=[ones(9,1),x,x.^2]

A =

    1.0000         0         0
    1.0000    0.2500    0.0625
    1.0000    0.5000    0.2500
    1.0000    0.7500    0.5625
    1.0000    1.0000    1.0000
    1.0000    1.2500    1.5625
    1.0000    1.5000    2.2500
    1.0000    1.7500    3.0625
    1.0000    2.0000    4.0000

 det(A'*A)

ans =

   40.6055

 alpha_hat=inv(A'*A)*A'*Y

alpha_hat =

    1.0652
   -0.6258
    2.4545

 c0=alpha_hat(1)

c0 =

    1.0652

 c1=alpha_hat(2)

c1 =

   -0.6258

 c2=alpha_hat(3)
```

```
c2 =

    2.4545
```

```
plot(x,y,'o',x,c0+c1*x+c2*x.^2)
axis([0 2 0 12])
gtext('y = c_0+c_1x + c_2 x^2')
xlabel('x')
ylabel('y')
title('Least square fit of a quadratic to data')
```

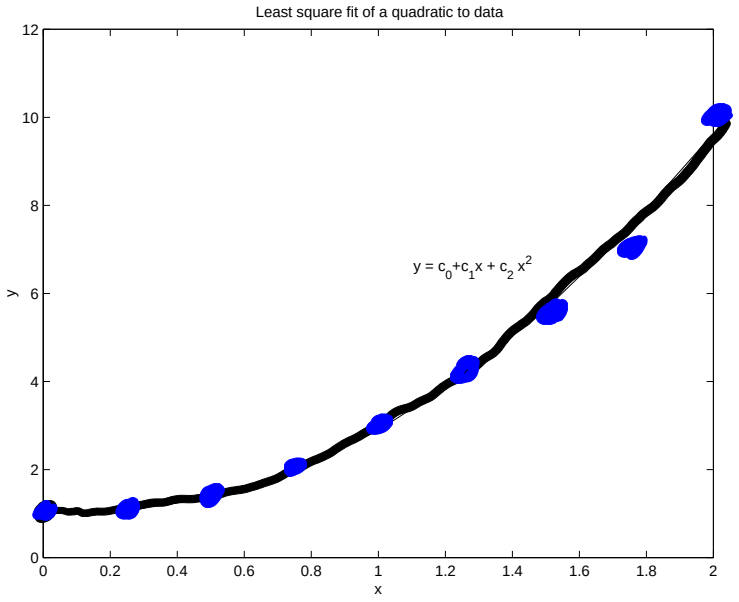

Fig. 4. Plot of data from Table 2 along with the least squares fit of a quadratic function to the data.