Line of Best Fit using Matlab

We solve for the line of best fit two ways- One way is to use Matlab's built in solver, and the other is to use the normal equations. Let's review the techniques first.

For notation, let's suppose that our data is stored as one vector holding the x coordinate and one vector holds the y coordinate (which we'll call target values). For example, if the data points are $(1,0),(2,0),(3,1)$, then $x=[1,2,3]^T$ and $t=[0,0,1]^T$.

To construct the matrices and vectors, recall that given the model $y=mx+b$, applying the data:

$$\begin{array}{rl} mx_1+b & =t_1 \\ mx_2+b & =t_2 \\ \vdots & \vdots \\ mx_p+b & =t_p \end{array} \quad \Rightarrow \quad \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_p & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} = \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_p \end{bmatrix} \quad \Rightarrow \quad A\mathbf{c}=\mathbf{t}$$

Here's a sample run in Matlab. It's easiest to edit this in Matlab's editor and save it as an $m-$file (for example, `LineFit01.m`). To run it in Matlab, type the file name in the command window.

```
%Example 1, p 420
numpts=4;
x=[2 5 7 8]'; t=[1 2 3 3]';
A=[x ones(numpts,1)];

c=A\t;  % c is the least squares solution

xTest=linspace(0,8);   %Create test data
yTest=c(1)*xTest+c(2); %Create the line
plot(x,t,'k*',xTest,yTest,'k-');  % Plot the data with the line
```

Which outputs a graph that like that shown in Figure .

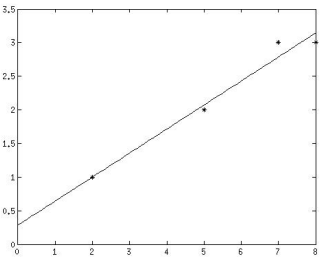

Figure 1: The output of the first Matlab example showing the line of best fit.

The solution from the normal equations is given by first multiplying both sides of our equation by A^T:

$$A\mathbf{c}=\mathbf{t} \quad \Rightarrow \quad A^TA\mathbf{c}=A^T\mathbf{t} \quad \Rightarrow \quad \mathbf{c}=(A^TA)^{-1}A^T\mathbf{t}$$

In Matlab, this would look like:

```
%Example 1 using the normal equation
numpts=4;
x=[2 5 7 8]'; t=[1 2 3 3]';
A=[x ones(numpts,1)];

c=inv(A'*A)*A'*t;  % c is the least squares solution

xTest=linspace(0,8);   %Create test data
yTest=c(1)*xTest+c(2); %Create the line
plot(x,t,'k*',xTest,yTest,'k-');  % Plot the data with the line
```

Solving from a Linear Model

Similar to the line of best fit, we might want to fit a polynomial of degree n to our data.

You might recall from algebra that it takes two points for a line (or degree 1 polynomial), three points for a parabola (or degree two polynomial), and generally it takes $n+1$ points to define a polynomial of degree n.

Suppose we have 100 points. We certainly should not try to fit a polynomial of degree 99- such a polynomial would have wild swings (we'll try an example below). Rather, suppose we think a parabola should fit our data pretty nicely- Then we look for a polynomial of degree 2:

$$t = a_0 + a_1 x + a_2 x^2$$

Building the matrix will be similar to the previous one:

$$\begin{array}{rl} a_0 + a_1 x_1 + a_2 x_1^2 & = t_1 \\ a_0 + a_1 x_2 + a_2 x_2^2 & = t_2 \\ \vdots & \\ a_0 + a_1 x_p + a_2 x_p^2 & = t_p \end{array} \quad \Rightarrow \quad \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ \vdots & \vdots & \\ 1 & x_p & x_p^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_p \end{bmatrix}$$

Or the model may be something like:

$$t = a_0 \sin(x) + a_1 \sin(2x) + a_2 \cos(x) + a_3 \cos(3x)$$

And even though this looks nonlinear, it is a linear problem in a_0, a_1, a_2 and a_3. The matrix equation would be:

$$\begin{bmatrix} \sin(x_1) & \sin(2x_1) & \cos(x_1) & \cos(3x_1) \\ \sin(x_2) & \sin(2x_2) & \cos(x_2) & \cos(3x_2) \\ \vdots & \vdots & & \\ \sin(x_p) & \sin(2x_p) & \cos(x_p) & \cos(3x_p) \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_p \end{bmatrix}$$

Here's the example done up in Matlab. First, we make up some random noisy data to use, then we build the model.

```
%Make up some noisy data.
x=3*pi*rand(40,1);  %40 random numbers between 0 and 3*pi
x=sort(x);
t=-sin(x)+3*sin(2*x)-0.5*cos(x)+4*cos(3*x)+randn(size(x));

%Create matrix A:
A=[sin(x) sin(2*x) cos(x) cos(3*x)];
c=A\t;

xTest=linspace(0,3*pi)';
yTest=[sin(xTest) sin(2*xTest) cos(xTest) cos(3*xTest)]*c;
plot(x,t,'k.',xTest,yTest,'k-');
```

The output to Matlab is given in Figure .

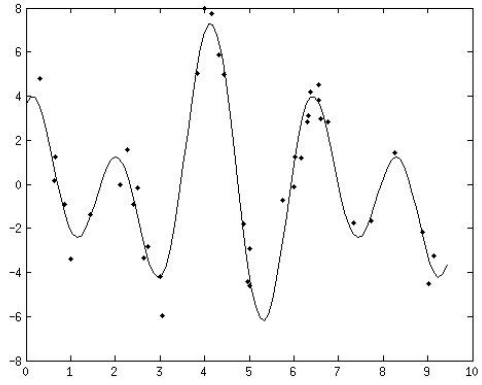

Figure 2: The output of the second Matlab example showing the resulting curve.