Linear and Nonlinear Least Squares Regression

C. Alex Simpkins

October 25, 2006

Least squares regression is one useful way to fit a curve to data. The following derivations are from the partial differential equations approach. This can also be derived with linear algebra in a much more abbreviated set of steps, but that approach may be more confusing to those less familiar with linear algebra.

1 Linear Least Squares

1.1 Theory and Derivation

Let us begin by considering a set of pairs of x and y data points.

The concept of least squares is to fit a linear or nonlinear curve which fits that data the best according to some criterion. One such common criterion is the minimization of sum of the squared differences between the actual data and the predicted data due to our least squares line. The error thus defined is given as

$$E = \sum_{i=1}^{M} \left[y_i - y_{LS}(x_i) \right]^2 \tag{1}$$

Where $i = 1, 2, ...M$ is the number of data points, and y_{LS} is the approximating curve's predicted y at the point x_i.

There are several reasons why we square the error and then find the minimum of that function. One of the most notable is to create a function with an attractive bowl shape and a single global minimum with respect to each line fitting parameter.

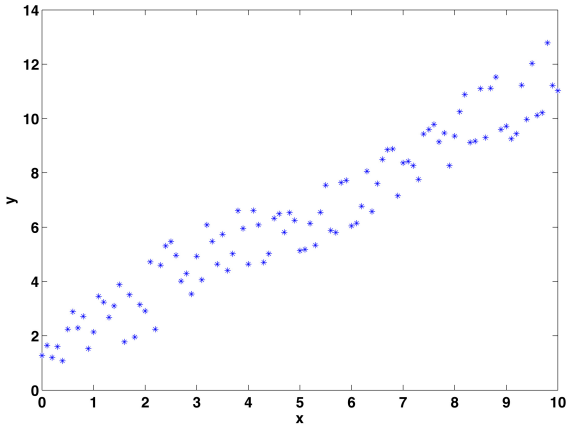

Figure 1: The following command will generate some random data in matlab with a linear relationship: x=0:.1:10; y=x+3*rand(size(x)); plot(x,y,'*');

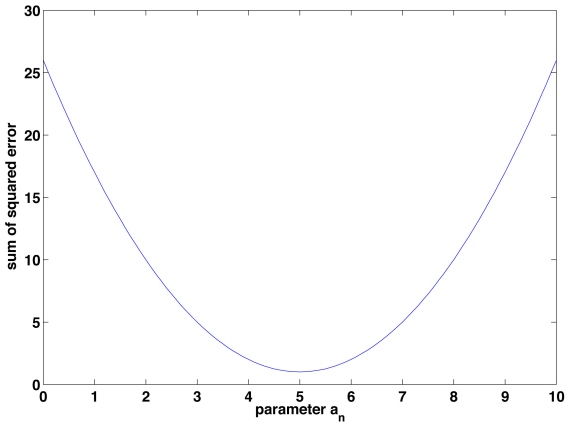

Figure 2: The classic quadratic 'bowl' shape

This serves to make positive and negative errors all positive (ie if y is bigger than y_{LS} at point i, the un-squared error is positive, and if y_{LS} is bigger than y at point i, the un-squared error would be negative- but if we square the error, negative numbers become positive, so we measure the magnitude of the error). A larger negative error is still a bigger error! Thus by looking at the sum of the *squared* error we know when we are reducing error (making our fit better) or increasing error (making our fit worse) as we vary our parameters, and we can define a measure of how to find the 'best fit.'

Going back to high school mathematics courses, we recall that the minimum of a function occurs when the derivative (slope) of that function is zero, and the second derivative is positive (think of it as the bottom of a valley versus the top of a hill - we want a valley shape). We have more than one parameter we can vary, so we have to consider the partial derivative with respect to each parameter.

If we want to fit a line, we have only two parameters, a_0 and a_1 to fit the line defined by

$$y_{LS} = a_0 + a_1 x \tag{2}$$

Thus our error equation becomes, after substituting y_{LS} in,

$$E = \sum_{i=1}^{M} \left[y_i - (a_0 + a_1 x_i) \right]^2 \tag{3}$$

To find the parameters a_0 and a_1 which minimize the error E, we take the partial derivative with respect to each parameter, a_0 and a_1 and set each resulting equation equal to zero.

$$\frac{\partial E}{\partial a_0} = 0 \tag{4}$$

$$\frac{\partial E}{\partial a_1} = 0 \tag{5}$$

$$\frac{\partial E}{\partial a_0} = \frac{\partial}{\partial a_0} \left\{ \sum_{i=1}^{M} \left[y_i - (a_0 + a_1 x_i) \right]^2 \right\} \tag{6}$$

$$\frac{\partial E}{\partial a_1} = \frac{\partial}{\partial a_1}\left\{ \sum_{i=1}^{M} \left[y_i - (a_0 + a_1 x_i)\right]^2 \right\} \tag{7}$$

Which gives us

$$\frac{\partial E}{\partial a_0} = 2\sum_{i=1}^{M} \left[y_i - (a_0 + a_1 x_i)\right](-1) \tag{8}$$

$$\frac{\partial E}{\partial a_1} = 2\sum_{i=1}^{M} \left[y_i - (a_0 + a_1 x_i)\right](-x_i) \tag{9}$$

Which can be simplified by the following steps (note we are working with pairs of equations)

$$\sum_{i=1}^{M} y_i = \sum_{i=1}^{M} a_0 + \sum_{i=1}^{M} a_1 x_i \tag{10}$$

$$\sum_{i=1}^{M} y_i x_i = \sum_{i=1}^{M} a_0 x_i + \sum_{i=1}^{M} a_1 x_i^2 \tag{11}$$

and

$$\sum_{i=1}^{M} y_i = a_0 M + a_1 \sum_{i=1}^{M} x_i \tag{12}$$

$$\sum_{i=1}^{M} y_i x_i = a_0 \sum_{i=1}^{M} x_i + a_1 \sum_{i=1}^{M} x_i^2 \tag{13}$$

Which we can solve by realizing that we have a familiar matrix algebra problem:

$$a_0 M + a_1 \sum_{i=1}^{M} x_i = \sum_{i=1}^{M} y_i \tag{14}$$

$$a_0 \sum_{i=1}^{M} x_i + a_1 \sum_{i=1}^{M} x_i^2 = \sum_{i=1}^{M} y_i x_i \tag{15}$$

$$\begin{bmatrix} M & \sum_{i=1}^{M} x_i \\ \sum_{i=1}^{M} x_i & \sum_{i=1}^{M} x_i^2 \end{bmatrix} \begin{Bmatrix} a_0 \\ a_1 \end{Bmatrix} = \begin{Bmatrix} \sum_{i=1}^{M} y_i \\ \sum_{i=1}^{M} y_i x_i \end{Bmatrix} \tag{16}$$

Which, if we set $A = \begin{bmatrix} M & \sum_{i=1}^{M} x_i \\ \sum_{i=1}^{M} x_i & \sum_{i=1}^{M} x_i^2 \end{bmatrix}$, $x = \begin{Bmatrix} a_0 \\ a_1 \end{Bmatrix}$, and $b = \begin{Bmatrix} \sum_{i=1}^{M} y_i \\ \sum_{i=1}^{M} y_i x_i \end{Bmatrix}$ is equivalent to

$$Ax = b \tag{17}$$

Note that the A, x, and b are not the same variables as the x's and a's in the above equations. We can solve this problem by finding the inverse of A (A^{-1}), and multiplying both sides by A^{-1}. This gives us the parameters of a_0 and a_1:

$$x = A^{-1}b \tag{18}$$

We can find the inverse by Gaussian elimination (or a more efficient algorithm), which will not be covered here. To find it by hand you could use Cramer's Rule to directly solve for a_0 and a_1. If you are unfamiliar with Gaussian elimination it is suggested you review the concept from linear algebra[1].

1.2 Matlab implementation

Performing a linear least squares regression in matlab is very simple using the left matrix divide (type *help mldivide* at the command prompt for more information).

Let's consider a simple case where you have three points and you want to fit a straight line using least squares regression. The points are (1,2) (3,4) (2, 3.5). Then we have $x = [1, 3, 2], y = [2, 4, 3.5]$. If we substitute the points into the equation for a straight line we have

$$a_0 + 1a_1 = 2 \tag{19}$$

$$a_0 + 3a_1 = 4 \tag{20}$$

$$a_0 + 2a_1 = 3.5 \tag{21}$$

Which is an over-constrained problem since there is more information than unknowns. Now we can recognize the canonical linear algebra problem and define $A = \begin{bmatrix} 1 & 1 \\ 1 & 3 \\ 1 & 2 \end{bmatrix}$, $x = \begin{Bmatrix} a_0 \\ a_1 \end{Bmatrix}$, and $b = \begin{Bmatrix} 2 \\ 4 \\ 3.5 \end{Bmatrix}$. Then after typing these into matlab, we can solve for x by writing $x = A \backslash b$ at the command prompt. Matlab promptly solves for x (a_0 and a_1). If you have more points just add more rows to A and b. Now to plot your fit over your data, type plot (x,y, '*') to plot your actual data. Then create another pair of points, xs = [0, 5]; ys = x(1) + x(2)*xs; hold on; plot(xs,ys,'r'). Your results should look like the following:

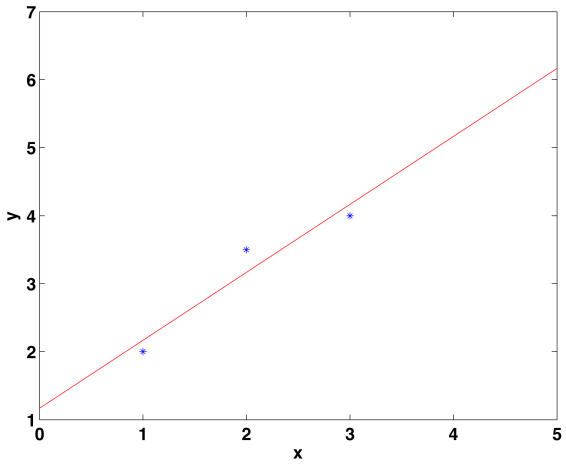

Figure 3: A basic least squares data fit to three points

2 Nonlinear Least Squares

2.1 Theory and Derivation[2]

Not all processes are linear (in fact most real processes are not linear). It is therefore more appropriate in such cases to have nonlinear data fitting methods.

One basic nonlinear function is a polynomial. Consider that we have a set of data $\{x_i, y_i\}$ with $i = 0, 1, ...M$ data points, and we would like to fit a polynomial of order n. The general equation for a polynomial is given by

$$P_n(x) = a_0x^0 + a_1x^1 + a_2x^2 + ... + a_nx^n \tag{22}$$

or more compactly written as

$$P_n(x) = \sum_{k=0}^{n} a_kx^k \tag{23}$$

Let us find once again the least squares approximation of our data using this polynomial fit rather than a linear fit. It should be noted that a linear fit is merely a polynomial of degree one (recall that the degree or order of a polynomial is the number of the highest power to which a variable in the polynomial equation is raised) . To compute the least square polynomial fit we will use the same approach as the linear case. It is important to choose $n < M$ (do not try to fit a polynomial with a higher order than the number of points of data). Let us begin with

$$E = \sum_{i=1}^{M} [y_i - P(x_i)]^2 \tag{24}$$

$$E = \sum_{i=1}^{M} y_i^2 - 2\sum_{i=1}^{M} y_iP(x_i) + \sum_{i=1}^{M} P^2(x_i) \tag{25}$$

$$E = \sum_{i=1}^{M} y_i^2 - 2\sum_{i=1}^{M} y_i \left[\sum_{k=0}^{n} a_k x_i^k\right] + \sum_{i=1}^{M} \left[\sum_{k=0}^{n} a_k x_i^k\right]^2 \tag{26}$$

Which we can simplify to

$$E = \sum_{i=1}^{M} y_i^2 - 2\sum_{k=0}^{n} a_k \sum_{i=1}^{M} y_i x_i^k + \sum_{h=0}^{n}\sum_{k=0}^{n} a_h a_k \left[\sum_{i=1}^{M} x_i^{h+k}\right] \tag{27}$$

Now similar to before, to find the minimum of the error, E, we must find where the partial derivative with respect to each variable is zero:

$$\frac{\partial E}{\partial a_k} = 0 \tag{28}$$

for each $k = 0, 1, 2, ...n$

$$0 = \frac{\partial E}{\partial a_k} = -2\sum_{i=1}^{M} y_i x_i^k + 2\sum_{h=0}^{n} a_h \sum_{i=1}^{M} x_i^{h+k} \tag{29}$$

We now have n+1 equations in n+1 unknown parameters a_k. These are referred to as the normal equations:

$$\sum_{h=0}^{n} a_h \sum_{i=1}^{M} x_i^{h+k} = \sum_{i=1}^{M} y_i x_i^k \tag{30}$$

$k = 0, 1, 2, ...n$

Written in this form we may not initially recognize this equation as being simple to solve. Let us consider this equation in a less short shorthand notation, then take a step back and look for a pattern.

$$a_0 \sum_{i=1}^{M} x_i^0 + a_1 \sum_{i=1}^{M} x_i^1 + a_2 \sum_{i=1}^{M} x_i^2 + ... + a_n \sum_{i=1}^{M} x_i^n = \sum_{i=1}^{M} y_i x_i^0 \tag{31}$$

$$a_0 \sum_{i=1}^{M} x_i^1 + a_1 \sum_{i=1}^{M} x_i^2 + a_2 \sum_{i=1}^{M} x_i^3 + ... + a_n \sum_{i=1}^{M} x_i^{n+1} = \sum_{i=1}^{M} y_i x_i^1 \tag{32}$$

.

.

.

$$a_0 \sum_{i=1}^{M} x_i^n + a_1 \sum_{i=1}^{M} x_i^{n+1} + a_2 \sum_{i=1}^{M} x_i^{n+2} + ... + a_n \sum_{i=1}^{M} x_i^{2n} = \sum_{i=1}^{M} y_i x_i^n \tag{33}$$

If we consider this in a matrix algebra perspective, we see that we can put this in the $Ax = b$ canonical form:

let us define

$$x = \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix} \tag{34}$$

and

$$b = \begin{pmatrix} \sum_{i=1}^{M} y_i \\ \sum_{i=1}^{M} y_i x_i \\ \sum_{i=1}^{M} y_i x_i^2 \\ \vdots \\ \sum_{i=1}^{M} y_i x_i^n \end{pmatrix} \tag{35}$$

and finally

$$A = \begin{bmatrix} \sum_{i=1}^{M} x_i^0 & \sum_{i=1}^{M} x_i^1 & \sum_{i=1}^{M} x_i^2 & \cdots & \sum_{i=1}^{M} x_i^n \\ \sum_{i=1}^{M} x_i^1 & \sum_{i=1}^{M} x_i^2 & \sum_{i=1}^{M} x_i^3 & \cdots & \sum_{i=1}^{M} x_i^{n+1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^{M} x_i^n & \sum_{i=1}^{M} x_i^{n+1} & \sum_{i=1}^{M} x_i^{n+2} & \cdots & \sum_{i=1}^{M} x_i^{2n} \end{bmatrix} \tag{36}$$

Then we can write this:

$$\begin{bmatrix} \sum_{i=1}^{M} x_i^0 & \sum_{i=1}^{M} x_i^1 & \sum_{i=1}^{M} x_i^2 & \cdots & \sum_{i=1}^{M} x_i^n \\ \sum_{i=1}^{M} x_i^1 & \sum_{i=1}^{M} x_i^2 & \sum_{i=1}^{M} x_i^3 & \cdots & \sum_{i=1}^{M} x_i^{n+1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^{M} x_i^n & \sum_{i=1}^{M} x_i^{n+1} & \sum_{i=1}^{M} x_i^{n+2} & \cdots & \sum_{i=1}^{M} x_i^{2n} \end{bmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{M} y_i \\ \sum_{i=1}^{M} y_i x_i \\ \sum_{i=1}^{M} y_i x_i^2 \\ \vdots \\ \sum_{i=1}^{M} y_i x_i^n \end{pmatrix} \tag{37}$$

Or more simply...

$$Ax = b \tag{38}$$

x is found by

$$x = A^{-1}b \tag{39}$$

Which can be solved, as before by Gaussian elimination, or another (more efficient) algorithm.

2.2 Matlab implementation

The exact same approach can be used as before to implement this in matlab using the left matrix divide.

Let's consider a simple case where you have five points and you want to fit a quadratic polynomial using least squares regression (quadratic is of degree 2). The points are (1,2) (3,4) (2, 3.5), (4, 7.5) and (5, 11.5). Then we have $x = [1, 3, 2, 4, 5], y = [2, 4, 3.5, 7.5, 11.5]$. If we substitute the points into the equation for a quadratic polynomial we have

$$a_0 + 1a_1 + (1)^2 a_2 = 2 \tag{40}$$

$$a_0 + 3a_1 + (3)^2 a_2 = 4 \tag{41}$$

$$a_0 + 2a_1 + (2)^2 a_2 = 3.5 \tag{42}$$

$$a_0 + 4a_1 + (4)^2 a_2 = 7.5 \tag{43}$$

Which is an over-constrained problem since there is more information than there are unknowns. Now we can recognize the canonical linear algebra problem and define $A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 3 & 9 \\ 1 & 2 & 4 \\ 1 & 4 & 16 \\ 1 & 5 & 25 \end{bmatrix}$, $x = \begin{Bmatrix} a_0 \\ a_1 \\ a_2 \end{Bmatrix}$, and $b = \begin{Bmatrix} 2 \\ 4 \\ 3.5 \\ 7.5 \\ 11.5 \end{Bmatrix}$. Then after typing these into matlab, we can solve for x by writing $x = A \setminus b$ at the command prompt. Matlab promptly solves for x, which we assign to a by $a = x$, ie (a_0, a_1, and a_2). If you have more points just add more rows to A and b. Now to plot your fit over your data, type:

$x = [1, 3, 2, 4, 5]; y = [2, 4, 3.5, 3.5, 11.5];$

plot(x,y,'*r');

Then create another set of points by creating the x values, and plugging them into our equation from the least squares fit,

$xf = 0.5 : .1 : 6;$

$yf = a(1) + a(2) * xf + a(3) * xf.^2;$

hold on; plot(xf, yf,'r').

Your results should look like the following:

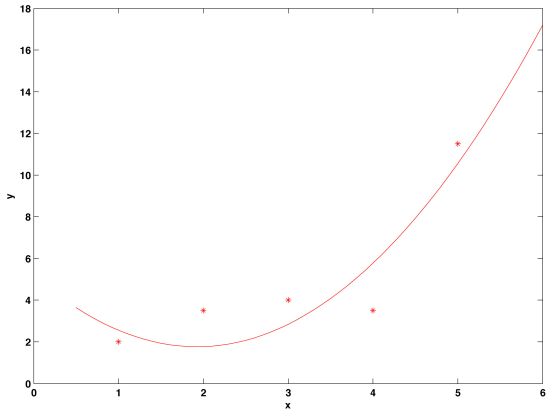

Figure 4: A basic quadratic least squares data fit to four points

References

[1] David C. Lay. *Linear Algebra and Its Applications*. Addison Wesley, 2nd edition edition, 1996.

[2] F. Talke and N. Delson. Mae 150 course reader. Soft Reserves, UCSD, 9500 Gilman Drive, October 2001.