AMTH142 Lecture 16

Least Squares

April 20, 2007

All the Scilab functions defined in this Lecture can be found in the file `l16.sci` in the directory for this lecture.

Contents

16.1 Least Squares

In the previous lecture we looked at interpolation problems where we were given some data (x_i, y_i), $i = 1, \ldots, n$ and we wanted to find a function $y = f(x)$ which interpolated the data so that $y_i = f(x_i)$, $i = 1, \ldots, n$. This approach is only useful when the data are relatively free from error, otherwise the interpolating function will exhibit the same kind of fluctuations that are present in the data.

When we are dealing with data containing random errors the most common approach to fitting a function to data is the method of least squares.

16.1.1 The General Problem

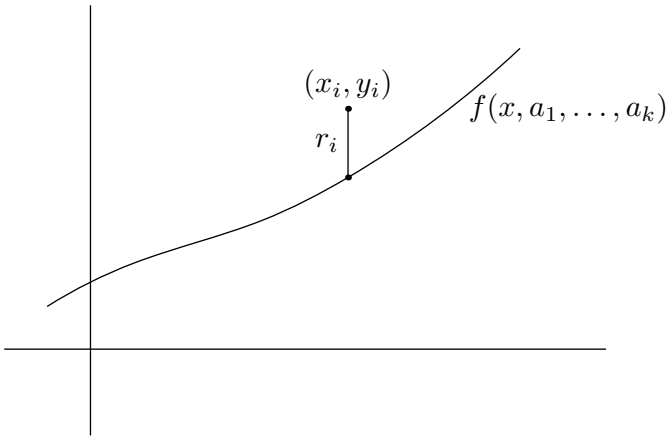
The general least squares problem can be formulated as follows: given data (x_i, y_i), $i = 1, \ldots, n$ and a function

$$y = f(x, a_1, a_2, \ldots, a_k)$$

depending on the parameters $a_1, \ldots, a_k$, let

$$r_i = y_i - f(x_i, a_1, \ldots, a_k)$$

denote the distance between the graph of $f(x, a_1, a_2, \ldots, a_k)$ and the data point (x_i, y_i).



We want to find values for the parameters $a_1, \ldots, a_k$ which minimizes the sum of squares of the r_i

$$\text{Minimize} \quad S^2 = \sum_{i=1}^{n} r_i^2 = \sum_{i=1}^{n} (y_i - f(x_i, a_1, \ldots, a_k))^2$$

Typically the number of parameters, k, is much smaller than the number of data points, n.

Least squares problems are optimization problems. They are classified as linear or nonlinear, depending on whether the function $f(x, a_1, \ldots, a_k)$ depends linearly or non-linearly on the parameters $a_1, \ldots, a_k$. Nonlinear least squares problems are often difficult to solve and the theory behind them is rather involved.

16.2 Linear Least Squares

The most common data fitting problem is fitting a straight line

$$y = ax + b$$

to data. This is an example of a linear least squares problem with two parameters, in this case a and b, to be determined. More generally, fitting a polynomial of degree k

$$y = a_0 + a_1 x + a_2 x^2 + \ldots a_k x^k$$

to data is another example of linear least squares. Although y is a nonlinear function of x it is a *linear* function of the parameters $a_0, \ldots, a_k$.

Linear least squares have a number of similarities to the general interpolation problem discussed in the previous lecture. Like interpolation it can be formulated as a problem in linear algebra.

16.2.1 The Computational Problem

Saying that the function $f(x, a_1, a_2, \ldots, a_k)$ depends linearly on the parameters a_j means that it can be written as a linear combination of some basis functions $f_1(x), \ldots, f_k(x)$:

$$f(x, a_1, a_2, \ldots, a_k) = a_1 f_1(x) + a_2 f_2(x) + \cdots + a_k f_k(x)$$

The least squares approach leads us to minimizing the sum of squares

$$S^2 = \sum_{i=1}^{n} [y_i - (a_1 f_1(x_i) + a_2 f_2(x_i) + \cdots + a_k f_k(x_i))]^2$$

Note that the sum is over the data points, and once the data and basis functions are given, the sum of squares S^2 is a function of the parameters $a_1, \ldots, a_k$ only.

The individual terms inside the square brackets

$$r_i = y_i - (a_1 f_1(x_i) + a_2 f_2(x_i) + \cdots + a_k f_k(x_i))$$

can be written in vector/matrix form

$$\begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} - \begin{bmatrix} f_1(x_1) & f_2(x_1) & \ldots & f_k(x_1) \\ f_1(x_2) & f_2(x_2) & \ldots & f_k(x_2) \\ \vdots & \vdots & \vdots & \vdots \\ f_1(x_n) & f_2(x_n) & \ldots & f_k(x_n) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_k \end{bmatrix}$$

The matrix in this equation is the same as the *basis matrix* used in interpolation. Like interpolation our aim is to find the coefficients $a_1, \ldots, a_k$, this time to minimize the sum of squares of the r_i. Note that when we have the same number of parameters a_i as data points, the problem reduces to an interpolation problem. In this case we could find values for the parameters so that all the r_i are zero.

16.2.2 Over-Determined Linear Systems

The formulation of the linear least squares problem given above is closely related to solving linear equations when there are more equations than unknowns. Consider a system of n linear equations in k unknowns:

$$\begin{array}{ccccccccc} b_{11}a_1 & + & b_{12}a_2 & + & \ldots & + & b_{1k}a_k & = & y_1 \\ b_{21}a_1 & + & b_{22}a_2 & + & \ldots & + & b_{2k}a_k & = & y_2 \\ \vdots & & \vdots & & & & \vdots & & \vdots \\ b_{n1}a_1 & + & b_{n2}a_2 & + & \ldots & + & b_{nk}a_k & = & y_n \end{array}$$

In matrix form,

$$\mathbf{Ba} = \mathbf{y}$$

where $\mathbf{B}$ is a given $n \times k$ matrix, $\mathbf{y}$ is a given n vector, and $\mathbf{a}$ is the k vector to be solved for.

When $n > k$ then the system of equations will not, in general, have a solution. For any vector $\mathbf{a}$ the residual is

$$\mathbf{r} = \mathbf{Ba} - \mathbf{y}$$

When we have the same number of equations as unknowns, the residual is zero for the solution vector $\mathbf{a}$. When we have more equations than unknowns, there is no vector $\mathbf{a}$ for which the residual is zero.

However what we can do in this case is to try to find a vector $\mathbf{a}$ for which the residual is as small as possible. If we measure the size of the residual by its norm

$$||\mathbf{r}|| = \left[\sum_{i=1}^{m} r_i^2\right]^{\frac{1}{2}}$$

then minimizing the norm is equivalent to minimizing

$$S^2 = \sum r_i^2$$

.

The linear least squares problem has exactly the same mathematical structure. Given (x_i, y_i), $i = 1, \ldots, n$ and basis functions $f_1(x), \ldots, f_k(x)$

form the basis matrix

$$\mathbf{B} = \begin{bmatrix} f_1(x_1) & f_2(x_1) & \dots & f_k(x_1) \\ f_1(x_2) & f_2(x_2) & \dots & f_k(x_2) \\ \vdots & \vdots & \vdots & \vdots \\ f_1(x_n) & f_2(x_n) & \dots & f_k(x_n) \end{bmatrix}$$

then we want to find the coefficients a_i of the basis functions minimize the norm of the residual

$$\mathbf{r} = \mathbf{Ba} - \mathbf{y}.$$

16.2.3 Scilab

In Scilab least squares problems, which as we have seen are equivalent to over-determined linear systems, are solved with the backslash operator \ the same way as linear equations are solved.

Example

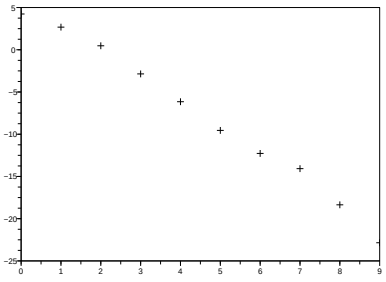
We will look at example of fitting a straight line to data. Our basis functions are $f_1(x) = 1$ and $f_2(x) = x$ and the basis matrix is

$$\begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}$$

For our example we will generate some x and y data where

$$y = -3x + 5 + \text{random perturbation}$$

```
-->x1 = (0:9)';

-->y1 = -3*x1 + 5 + rand(x1, 'normal');

-->plot2d(x1, y1, style = -1)
```



To fit a straight line to this data we first create the the basis matrix:

```
-->aa1 = [ones(x1) x1]
 aa1  =

!   1.    0. !
!   1.    1. !
!   1.    2. !
!   1.    3. !
!   1.    4. !
!   1.    5. !
!   1.    6. !
!   1.    7. !
!   1.    8. !
!   1.    9. !
```

Solving with the backslash operator

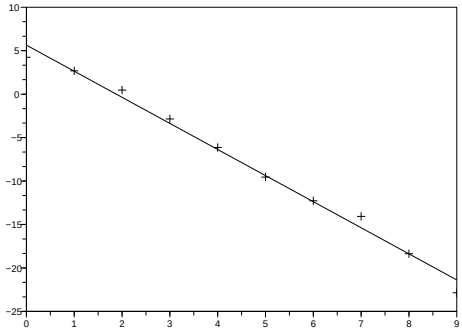
```
-->a = aa1\y1
 a  =

!   5.6400247 !
! - 3.0027612 !
```

The components of the solution, `a`, are the coefficients of the basis functions $f_1(x) = 1$ and $f_2(x) = x$, so our least squares straight line is

$$y = -3.003x + 5.640.$$

We can now evaluate and plot the straight line

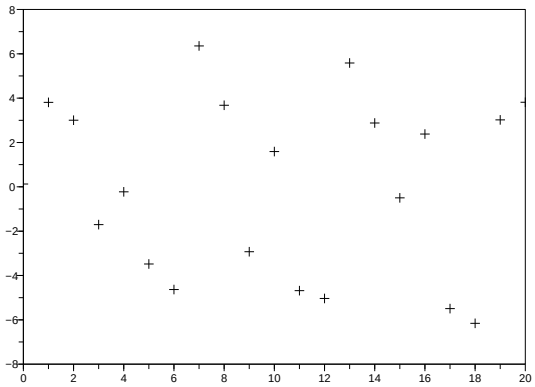
```
-->xx = 0:0.01:9;

-->yy = a(1) + a(2)*xx;

-->plot2d(xx,yy)
```



Example

Here is another example. Again we will generate some data, but this time using trigonometric functions:

```
-->x2 = (0:20)';

-->y2 = 4*sin(x2)+3*sin(2*x2)+2*sin(4*x2)+0.5*rand(x2,'normal');

-->plot2d(x2, y2, style = -1)
```

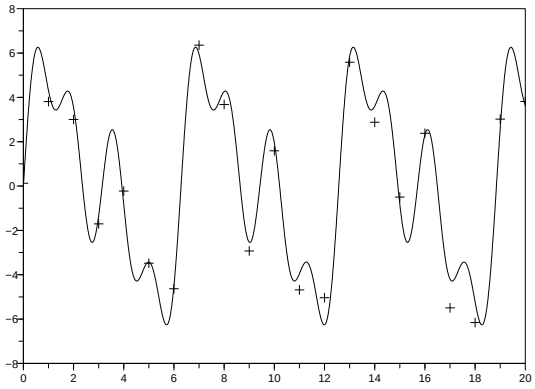


In this problem our basis functions will be

$$f_1(x) = \sin(x) \quad f_2(x) = \sin(2x) \quad f_3(x) = \sin(4x)$$

Our computations follow the same pattern as before:

```
-->aa2 = [sin(x2) sin(2*x2) sin(4*x2)];

-->a = aa2\y2
 a  =

!   3.9648784 !
!   2.8008647 !
!   2.0763987 !

-->xx = 0:0.01:20;

-->yy = a(1)*sin(xx) + a(2)*sin(2*xx) + a(3)*sin(4*xx);

-->plot2d(xx,yy)
```



16.2.4 Linear Functions

As we have seen, fitting a linear function to data is a simple application of linear least squares. It is handy to have a Scilab function to do this for us.

Given data in vectors `x` and `y` we first construct the basis matrix, in this case the 2 column matrix

```
aa = [ones(x) x]
```

(we have assumed `x` is a column vector), and then solve the least-squares problem to get the coefficients

```
a = aa\y
```

Here is our function:

```
function a = linfit(x, y)
  aa = [ones(x) x]
  a  = aa\y
endfunction
```

Example

We will use the same data `x1` and `y1` as before

```
-->a = linfit(x1,y1)
 a  =

!   5.6400247 !
! - 3.0027612 !
```

(and, of course, get the same result.)

16.2.5 Polynomials

This is another application of linear least squares. To fit a polynomial of degree k

$$y = a_0 + a_1 x + \ldots + a_k x^k$$

to a data set we use the basis matrix

```
aa = [ones(x) x x.^2 .... x.^k]
```

Again we will write a Scilab function to do the job for us. Not surprisingly it is similar to the function for polynomial interpolation in Lecture 15.

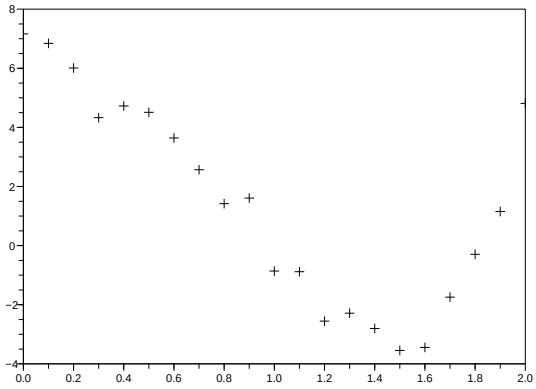
```
function a = polyfit(x, y, k)
  n = length(x)
  aa = zeros(n, k+1)
  aa(:,1) = ones(x);
  for i = 1:k        // loop to construct the basis matrix
    aa(:,i+1) = x.^i
  end
  a = aa\y
endfunction
```

The function `polyeval` from Lecture 15 can be used to evaluate the polynomial.

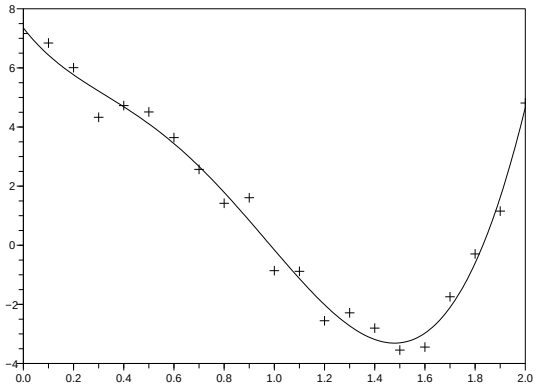
Example

Again we will generate some noisy data:

```
-->x3 = (0:0.1:2)';

-->y3 = x3.^5 - 3*x3.^3 - 5*x3 + 7 + 0.5*rand(x3,'normal');

-->plot2d(x3,y3,style = -1)
```

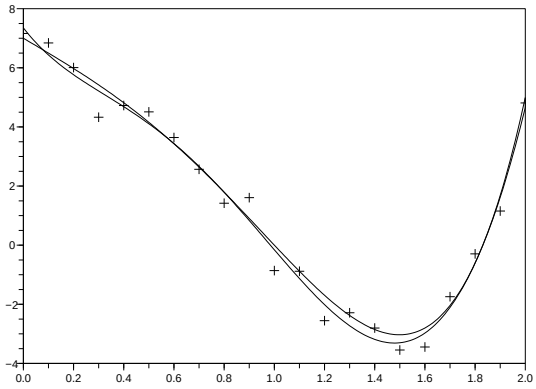


```
-->p = polyfit(x3,y3,5)
 p  =

!   7.3497182 !
! - 10.903575 !
!   20.310461 !
! - 29.978715 !
!   15.022939 !
! - 1.9582651 !

-->xx = (0:0.01:2)';

-->yy = polyeval(p,xx);

-->plot2d(xx,yy)
```



We can compare to the original unperturbed polynomial:

```
-->pp = [7 -5 0 -3 0 1]'
 pp  =

!   7. !
! - 5. !
!   0. !
! - 3. !
!   0. !
!   1. !

-->yy1 = polyeval(pp,xx);

-->plot2d(xx,yy1)
```

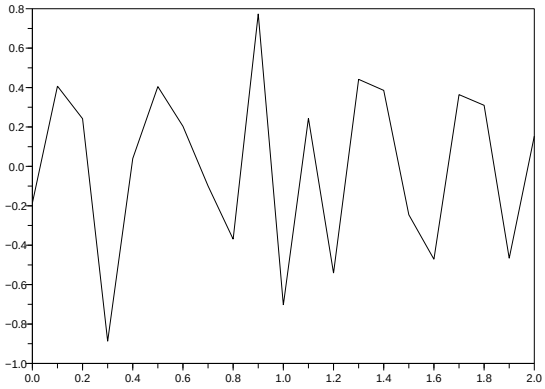


Although some of the coefficients of the two polynomials are quite different, the graphs are quite similar. Put another way, we have two quite different polynomials which both give a reasonable fit to the data.

16.2.6 Analyzing Results

The **residuals**, that is the differences between the data and the fitted function, are good way to determine to determine how well a function fits a set of data. For our example these are:

```
-->res3 = y3 - polyeval(p,x3);

-->plot2d(x3,res3)
```



The graph of the residuals should look random with no discernible pattern as in this example.