Lecture 19

Curve fitting I

1 Introduction

Suppose we are presented with eight points of measured data \((x_i, y_j)\). As shown in Fig. 1 on the left, we could represent the “underlying function” of which these data are samples by interpolating between the data points using one of the methods we have studied previously.

However, maybe the data are samples of the response of a process that we know, in theory, is supposed to have the form \(y = f(x) = ax + b\) where \(a, b\) are constants. Maybe we also know that \(y\) is a very weak signal and the sensor used to measure it is “noisy,” that is, it adds its own (random) signal in with the “true” \(y\) data. Given this it makes no sense to interpolate the data because in part we’ll be interpolating noise, and we know that the “real” signal should have the form \(y = ax + b\). In a situation like this we prefer to fit a line to the data rather than perform an interpolation (Fig. 1 at right). If done correctly this can provide a degree of immunity against the effects of measurement errors and noise. More generally we want to develop curve fitting techniques that allow theoretical curves, or models, with unknown parameters (such as \(a\) and \(b\) in the line case) to be “fit” to \(n\) data points.

2 Fitting a constant to measured data

The simplest “curve fitting” problem is estimating a parameter from multiple measurements. Suppose \(m\) is the mass of an object. We want to measure this using a scale. Unfortunately the scales in our laboratory are not well calibrated. However we have nine scales. We expect that if...
we take measurements with all of them and average the results we should get a better estimate of the true mass that by relying on the measurement from a single scale. Our results might look something like shown in Fig. 2. Let the measurement of the $i^{th}$ scale be $m_i$ then the average measurement is given by

\[ \bar{m} = \frac{1}{n} \sum_{i=1}^{n} m_i \]  

(1)

where $n$ is the number of measurements. This is what we should use for our “best estimate” of the true mass. Averaging is a very basic form of curve fitting.

### 3 Least-squares line fit

Going back to the situation illustrated in Fig. 1, how do we figure out the “best fit” line? There doesn't seem to be a straightforward way to “average” the data like we did in Fig. 2. Instead, let's suppose we have $n$ data points $(x_i, y_i)$. We are interested in a linear model of the form $y = ax + b$, and our task is calculate the “best” values for $a$ and $b$. If all our data actually fell on a line then the best $a$ and $b$ values would result in $y_i - (ax_i + b) = 0$ for $i = 1, 2, ..., n$. More generally let's define the residual (“error of the fit”) for the $i^{th}$ data point as

\[ r_i = y_i - (ax_i + b) \]  

(2)

A perfect fit would give $r_i = 0$ for all $i$. The residual can be positive or negative, but what we are most concerned with is its magnitude. Let's define the mean squared error (MSE) as

\[ MSE = \frac{1}{n} \sum_{i=1}^{n} r_i^2 = \frac{1}{n} \sum_{i=1}^{n} (y_i - (ax_i + b))^2 \]  

(3)
We now seek the values of $a$ and $b$ that minimize the $MSE$. These will satisfy

$$\frac{\partial \text{MSE}}{\partial a} = 0 \quad \text{and} \quad \frac{\partial \text{MSE}}{\partial b} = 0$$

(4)

The $b$ derivative is

$$\frac{\partial \text{MSE}}{\partial b} = -\frac{2}{n} \sum_{i=1}^{n} (y_i - (ax_i + b)) = 0$$

(5)

Multiplying through by $-1/2$ and rearranging we find

$$\frac{1}{n} \sum_{i=1}^{n} y_i - \frac{a}{n} \sum_{i=1}^{n} x_i - \frac{b}{n} \sum_{i=1}^{n} b = 0$$

(6)

Now define the average $x$ and $y$ values as

$$\langle y \rangle = \frac{1}{n} \sum_{i=1}^{n} y_i, \quad \langle x \rangle = \frac{1}{n} \sum_{i=1}^{n} x_i$$

(7)

Equation (6) then reads

$$\langle y \rangle - a \langle x \rangle - b = 0$$

(8)

or

$$a \langle x \rangle + b = \langle y \rangle$$

(9)

This tells us that the point $\left( \langle x \rangle, \langle y \rangle \right)$ (the “centroid” of the data) falls on the line.

The $a$ derivative of the $MSE$ is

$$\frac{\partial \text{MSE}}{\partial a} = -\frac{2}{n} \sum_{i=1}^{n} (y_i - (ax_i + b)) x_i = 0$$

(10)

Multiplying through by $-1/2$ and rearranging we find

$$\frac{1}{n} \sum_{i=1}^{n} x_i y_i - \frac{a}{n} \sum_{i=1}^{n} x_i^2 - \frac{b}{n} \sum_{i=1}^{n} x_i = 0$$

(11)

or

$$\langle x y \rangle - a \langle x^2 \rangle - b \langle x \rangle = 0$$

(12)

with the additional definitions

$$\langle x y \rangle = \frac{1}{n} \sum_{i=1}^{n} x_i y_i, \quad \langle x^2 \rangle = \frac{1}{n} \sum_{i=1}^{n} x_i^2$$

(13)

A final rearrangement gives us

$$a \langle x^2 \rangle + b \langle x \rangle = \langle x y \rangle$$

(14)

We now have two equations in the two unknowns $a, b$

$$a \langle x \rangle + b = \langle y \rangle$$

(15)
Solving the first equation for \( b \)

\[
b = \langle y \rangle - a \langle x \rangle
\]  
(16)

and substituting this into the second equation we obtain

\[
a \langle x^2 \rangle + (\langle y \rangle - a \langle x \rangle) \langle x \rangle = \langle xy \rangle
\]  
(17)

Solving this for \( a \) we have

\[
a = \frac{\langle xy \rangle - \langle x \rangle \langle y \rangle}{\langle x^2 \rangle - \langle x \rangle^2}
\]  
(18)

Equations (18) and (16) provide the “best-fit” values of \( a \) and \( b \). Because we obtained these parameters by minimizing the sum of squared residuals, this is called a least-squares line fit.

**Example.** The code below generates six points on the line \( y = 1 - x \) and adds normally-distributed noise of standard deviation 0.1 to the \( y \) values. Then (18) and (16) are used to calculate the best-fit values of \( a \) and \( b \). The data and fit line are plotted in Fig. 3. The “true” values are \( a = -1, b = 1 \). The fit values are \( a = -0.91, b = 1.02 \).

```matlab
--> x = [0:0.2:1]';
--> y = 1-x+rand(x,'normal')*0.1;
--> a = (mean(x.*y)-mean(x)*mean(y))/(mean(x.^2)-mean(x).^2)
a =
   -0.9103471
--> b = mean(y)-a*mean(x)
b =
   1.0191425
```

![Fig. 3: Least-squares line fit to noisy data.](image)
4 Linear least-squares

The least-squares idea can be applied to a linear combination of any \(m\) functions \(f_1(x), f_2(x), \ldots, f_m(x)\). Our model has the form

\[
y = \sum_{j=1}^{m} c_j f_j(x)
\]  

(19)

For example, if \(m=2\) and \(f_1(x)=1, f_2(x)=x\) then our model is

\[
y = c_1 + c_2 x
\]

(20)

which is just the linear case we've already dealt with. If we add \(f_3(x)=x^2\) then the model is

\[
y = c_1 + c_2 x + c_3 x^2
\]

(21)

which is an arbitrary quadratic. Or we could have a model such as

\[
y = c_1 \cos(5x) + c_2 \sin(5x) + c_3 \cos(10x) + c_4 \sin(10x)
\]

(22)

In any case we'll continue to define the residuals as the difference between the observed and the modeled \(y\) values

\[
r_i = y_i - \sum_{j=1}^{m} c_j f_j(x_i)
\]

(23)

and the mean-squared error as

\[
MSE = \frac{1}{n} \sum_{i=1}^{n} r_i^2 = \frac{1}{n} \sum_{i=1}^{n} \left( y_i - \sum_{j=1}^{m} c_j f_j(x_i) \right)^2
\]

(24)

Let's expand this as

\[
\frac{1}{n} \sum_{i=1}^{n} \left( y_i - \sum_{j=1}^{m} c_j f_j(x_i) \right)^2 = \frac{1}{n} \sum_{i=1}^{n} \left( y_i^2 - 2 y_i \sum_{j=1}^{m} c_j f_j(x_i) + \left[ \sum_{j=1}^{m} c_j f_j(x_i) \right]^2 \right)
\]

(25)

Call

\[
\frac{1}{n} \sum_{i=1}^{n} y_i^2 = \langle y^2 \rangle
\]

and

\[
-\frac{2}{n} \sum_{i=1}^{n} y_i \sum_{j=1}^{m} c_j f_j(x_i) = -\sum_{j=1}^{m} b_j c_j
\]

(26)

with

\[
b_j = \frac{2}{n} \sum_{i=1}^{n} y_i f_j(x_i)
\]

(27)

The last term in (25) can be written
\[
\left[ \sum_{j=1}^{m} c_j f_j(x_i) \right]^2 = \sum_{j=1}^{m} c_j f_j(x_i) \sum_{k=1}^{m} c_k f_k(x_i)
\]

(28)

Therefore

\[
\frac{1}{n} \sum_{i=1}^{n} \left[ \sum_{j=1}^{m} c_j f_j(x_i) \right]^2 = \frac{1}{n} \sum_{i=1}^{n} \left( \sum_{j=1}^{m} c_j f_j(x_i) \sum_{k=1}^{m} c_k f_k(x_i) \right) = \frac{1}{2} \sum_{j=1}^{m} \sum_{k=1}^{m} a_{jk} c_j c_k
\]

(29)

with

\[
a_{jk} = a_{kj} = \frac{2}{n} \sum_{i=1}^{n} f_j(x_i) f_k(x_i)
\]

(30)

Finally we can write

\[
MSE = \langle y^2 \rangle - \sum_{i=1}^{m} b_i c_i + \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} a_{ij} c_i c_j
\]

(31)

This shows that the \(MSE\) is a quadratic function of the unknown coefficients. In the lecture “Optimization in n dimensions” we calculated the solution to a system of this form, except that the second term (with the \(b\) coefficients) had a plus rather than minus sign. Defining the \(m\times1\) column vectors \(b\) and \(c\) and the \(m\timesm\) matrix \(A\) as

\[
c=[c_j], \quad b=[b_j], \quad A=[a_{ij}]
\]

(32)

the condition for a minimum is (with the minus sign for the \(b\) coefficients)

\[-b + A c = 0
\]

(33)

and

\[c = A^{-1} b\]

(34)

Another way to arrive at this result is to define the \(n\times1\) column vector

\[y=[y_i]\]

(35)

and the \(n\timesm\) matrix

\[F=[f_{ij}]\quad \text{with} \quad f_{ij}=f_j(x_i)\]

(36)

Then our model is

\[y = F c\]

(37)

This is \(n\) equations in \(m<n\) unknowns and in general will not have a solution. Multiplying both sides on the left by \(F^T\) results in the system

\[F^T F c = F^T y\]

(38)

Since \(F^T F\) is \(m\timesm\) and \(F^T y\) is \(m\times1\) this is a system of \(m\) equations in \(m\) unknowns that, in general, will have a unique solution

\[c = (F^T F)^{-1} F^T y\]

(39)

The elements of \(F^T F\) are
\[ [F^T F]_{jk} = \sum_{i=1}^{n} f_{ij} f_{ik} = \frac{n}{2} a_{jk} \]  

(40)

while the elements of \( F^T y \) are

\[ [F^T y]_j = \sum_{i=1}^{n} f_{ij} y_i = \frac{n}{2} b_j \]  

(41)

Therefore \( F^T F c = F^T y \), when multiplied through by \( 2/n \), is equivalent to

\[ A c = b \]  

(42)

The linear system (38) is called the normal equation, and we have the following algorithm

**Linear least squares fit**

*Given n samples \((x_i, y_i)\)*

*and a model \( y = \sum_{j=1}^{m} c_j f_j(x) \)*

*Form the \( n \times m \) matrix \( F \) with elements \( f_{ij} = f_j(x_i) \)*

*Form the \( n \times 1 \) column vector \( y \) with elements \( y_i \)*

*Solve the normal equation \( F^T F c = F^T y \) for \( c \)*

*The modeled \( y \) values are \( \hat{y} = F c \)*

The \( n \times m \) matrix \( F \) is not square if \( n > m \), so we cannot solve the linear system

\[ y = F c \]  

(43)

by writing

\[ c = F^{-1} y \]  

(44)

because \( F \) does not have an inverse. However, as we’ve seen, we can compute

\[ c = \left( F^T F \right)^{-1} F^T y \]  

(45)

and this \( c \) will come as close as possible (in a least-squares sense) to solving (43). This leads us to define the pseudoinverse of \( F \) as the \( m \times n \) matrix

\[ F^+ = \left( F^T F \right)^{-1} F^T \]  

(46)

Our least-squares solution can now be written

\[ c = F^+ y \]  

(47)

In Scilab/Matlab the pseudo inverse is computed by the command `pinv(F)`. However, if we simply apply the backslash operator as we would for a square system

\[ c = F \backslash y \]

Scilab/Matlab returns the least-squares solution. We do not have to explicitly form the normal
equation or the pseudoinverse.

**Example.** Noise was added to Eleven samples of \( y = x^2 - x \), \( x = 0, 0.1, 0.2, \ldots, 1 \). A least-squares fit of the model \( c_1 + c_2 x + c_3 x^2 \) gave

\[
c_1 = 0.044, c_1 = -1.110, c_2 = 1.039
\]

Code is shown below and results are plotted in Fig. 4.

```matlab
--> x = [0:0.1:1]';
--> y0 = x.^2-x;
--> y = y0 + rand(y0,'normal')*0.03; // add noise
--> F = [ones(x), x, x.^2];
--> c = F\y
   c  =
     0.0436524
     -1.1104735
       1.0390231
--> yf = F*c
```

5 **Goodness of fit**

Once we've fit a model to data we may wonder if the fit is “good” or not. It would be helpful to have a measure of *goodness of fit*. Doing this rigorously requires details from probability theory. We will present the following results without derivation.

Assume our \( y \) values are of the form

\[
y_i = s_i + \eta_i
\]

where \( s_i \) is the *signal* that we are trying to model and \( \eta_i \) is *noise*. If our model were to perfectly
fit the signal, then the residuals

\[ r_i = y_i - \sum_{j=1}^{m} c_j f_j(x_i) \]  \hspace{1cm} (48)

would simply be noise \( r_i = \eta_i \). We can quantify the goodness of fit by comparing the statistics of our residuals to the (assumed known) statistics of the noise. Specially, for large \( n-m \), and normally distributed noise, a good fit will result in the number

\[ \sigma = \sqrt{\frac{1}{n-m} \sum_{i=1}^{n} r_i^2} \]  \hspace{1cm} (49)

being equal, on average, to the standard deviation of the noise, where \( n \) is the number of data and \( m \) is the number of model coefficients. If it is significantly larger than this it indicates that the model is not accounting for all of the signal, where a fractional change of about \( \sqrt{\frac{2}{n-m}} \) is statistically significant. For example, \( \sqrt{\frac{2}{50}} = 0.2 \) means that a change of around 20% is statistically significant. If the noise standard deviation is 0.1, a \( \sigma \) larger than about 0.1(1.2)=0.12 implies the signal is not being fully modeled. The following example illustrates the use of this goodness-of-fit measure.

**Example.** The following code was used to generate 50 samples of the function \( f(x) = x + x^2 \) over the interval \( 0 \leq x \leq 1 \) with normally distributed noise of standard deviation 0.05 added to each sample.

```
n = 50;
rand('seed',2);
x = [linspace(0,1,n)]';
y = x+x.^2+rand(x,'normal')*0.05;
```

These data were then fit by the four models \( y = c_1 \), \( y = c_1 + c_2 x \), \( y = c_1 + c_2 x + c_3 x^2 \) and \( y = c_1 + c_2 x + c_3 x^2 + c_4 x^3 \). The resulting \( \sigma \) values were \( \sigma_0 = 0.6018 \), \( \sigma_1 = 0.0864 \), \( \sigma_2 = 0.0506 \) and \( \sigma_3 = 0.0504 \). Since \( \sqrt{\frac{2}{50}} = 0.2 \) a change of about 20% is statistically significant. The fits improved significantly until the last model. The data therefore support the model \( y = c_1 + c_2 x + c_3 x^2 \) but not the cubic model. The fits are shown in Fig. 5.
Fig. 5 Data set fit by polynomials. Top-left: $y = c_1$, $\sigma = 0.6018$. Top-right: $y = c_1 + c_2 x$, $\sigma = 0.0864$. Bottom-left: $y = c_1 + c_2 x + c_3 x^2$, $\sigma = 0.0506$. Bottom-right: $y = c_1 + c_2 x + c_3 x^2 + c_4 x^3$, $\sigma = 0.0504$. 