Fitting Functions

Showing that two signals $x$ and $y$ are correlated at a statistically significant level is interesting, but it doesn’t give us a lot of detail. Often we want to find a functional form that links two sets of measurements. For example, in the case of height and shoe size, we might like to have a way to predict height $h$ given knowledge of shoe size $s$, that is height as a function of shoe size, $h = f(s)$. Based on our scatter plot of shoe size versus height, we might start by guessing a function of the form:

$$h_i = bs_i; \tag{1}$$

where $b$ is a constant, and indicatess a slope and subscripts $i$ indicate each numbered data pair.

How do we determine $b$? You might imagine simply averaging the equation so that $\bar{h} = b\bar{s}$. Don’t do it. This might work sometimes, but it won’t give you a very general solution, and your results could be strangely influenced by data pairs that were poorly correlated. Instead we want to minimize the difference between $h$ and $f(s)$ by computing a “cost function” $\epsilon$ that depends on the misfit $h_i - f(s_i)$. We could try to minimize $|h_i - f(s_i)|$, but then we’d have to keep track of absolute values, so it would be mathematically awkward. Instead, we’ll minimize the squared error, $(h_i - f(s_i))^2$. Mathematically we can express this as a minimization of:

$$\epsilon = \sum_{i=1}^{N} (h_i - f(s_i))^2 = \sum_{i=1}^{N} (h_i - bs_i)^2 = \sum_{i=1}^{N} (h_i^2 - 2bhs_i) = \sum_{i=1}^{N} (h_i^2 - 2bhs_i + b^2s_i^2 - 2bhs_i) = N. \tag{2}$$

We’d like our cost function $\epsilon$ to be as small as possible. Since we’re looking for the smallest possible squared error, this is called a least squares fit. We’ll use a little calculus to find the minimum of $\epsilon$:

$$\frac{\partial \epsilon}{\partial b} = 2 \sum_{i=1}^{N} (bs_i - h_is_i) = 0. \tag{3}$$

Solving this, indicates that $b = \frac{\bar{h}s}{\bar{s}^2}$, and in this case $b = 6.84$. In our example, this solution does not work very well, as shown in Figure 1. Clearly we’ve forgotten something.

Our model needs to be a little more complicated. Instead we could try fitting

$$h_i = a + bs_i. \tag{4}$$

Here the constant $a$ will take account of the fact that the best fit to the two data sets does not intersect the origin. (Zero height doesn’t correspond to zero shoe size.) How do I determine $a$ and $b$? We have $N$ equations with two unknowns, if $N > 2$ our problem is “overdetermined”. We have too much information,
and our information is probably contradictory, since this is real data and it might be noisy, either because of measurement errors or because the real world does not exactly fit our prediction that height and shoe size should be exactly related.

As before, we can find the solution by setting the derivatives of $\epsilon$ to zero.

$$
\epsilon = \sum_{i=1}^{N} (h_i - f(s_i))^2 = \sum_{i=1}^{N} (h_i - a - bs_i)^2 = \sum_{i=1}^{N} (h_i^2 + a^2 + b^2s_i^2 - 2ah_i - 2bh_is_i + 2abs_i).
$$

so

$$
\frac{\partial \epsilon}{\partial a} = 2 \sum_{i=1}^{N} (a - h_i + bs_i) = 0 \tag{5}
$$

$$
\frac{\partial \epsilon}{\partial b} = 2 \sum_{i=1}^{N} (bs_i^2 - h_is_i + as_i) = 0 \tag{6}
$$

and we can work through the equations to solve for $a$ and $b$. In this case, $a = 58.4$ and $b = 1.14$. The resulting fit is plotted in Figure 2. Solving for our coefficients by differentiating $\epsilon$ in terms of all the unknowns will get a little tiresome as our number of unknowns increases. Instead, we need to find a more general solution.

Let’s rewrite our equation as a matrix equation:

$$
[h_1 \\ h_2 \\ h_3 \\ \vdots \\ h_N] = A\begin{bmatrix} 1 & s_1 \\ 1 & s_2 \\ 1 & s_3 \\ \vdots \\ 1 & s_N \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} a + bs_1 \\ a + bs_2 \\ a + bs_3 \\ \vdots \\ a + bs_N \end{bmatrix}, \tag{8}
$$

where $A$ is an $n \times 2$ matrix and $x$ is a two element vector containing $a$ and $b$. Now we just want to find the best solution $x$ for $h = Ax$. 

Figure 2: Scatter plot of shoe size versus height in inches with (green line) best fit for $h = a + bs$. 
A Quick Review of Linear Algebra

As a first step, we should review a little linear algebra. Matlab is a powerful tool for dealing with matrix equations such as this, and it will do most of the hard work. So we just need to review the basics.

We can always represent our least squares fitting problems as matrix equations of the form $Ax = d$, where $A$ is a matrix, $x$ is a vector containing the unknown coefficients and $d$ is a vector containing the data to which we are fitting.

Let’s start by thinking about how to define a matrix. The matrix $A$ can be represented as an $n \times m$ element array. Here $n$ tells us the number of rows and $m$ the number of columns. $A$ does not need to be square; in otherwords, we do not require that $n = m$.

In Matlab, we can fill a matrix one element at a time:

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix};$$

does the same thing as:

$$A(1,1)=1; \ A(1,2)=2; \ A(2,1)=3; \ A(2,2)=4;$$

where the resulting matrix is:

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}.$$

Alternatively we can specify the matrix elements a column at a time:

$$n = \text{length}(s);$$

$$A = \begin{bmatrix} \text{ones}(n,1) \ s \end{bmatrix};$$

produces the matrix $A$ shown above for our problem of fitting shoe size to height. We could make this more complicated. For example:

$$n = \text{length}(s);$$

$$A = \begin{bmatrix} \text{ones}(n,1) \ s \ s.^2 \end{bmatrix};$$

would produce an $n \times 3$ matrix containing a column of ones, a column of shoe sizes, and a column of shoe size squared.

We can multiply a matrix $A$ by a matrix $B$ provided that the number of columns in $A$ is equal to the number of rows in $B$. Similarly, we can multiply a matrix $A$ by a vector $x$ provided that the number of columns in $A$ is equal to the number of elements in $x$. The product of a $n \times m$ matrix $A$ and a $m \times p$ matrix $B$ will be a $n \times p$ matrix. For matrices, order of multiplication matters, so in general $AB \neq BA$, even if both products are defined.

If $C = AB$, then how do we compute $c_{11}$, the element in the first row and first column of $C$? Matlab will do this automatically, so we don’t need to fret about details of the computation, but we do need to remember what it represents. At index $i, j$, $c_{ij} = \sum_{l=1}^{m} a_{il} b_{lj}$. That is, we take the first row of $A$ and the first column of $B$, which must be the same length. We match up the elements, multiply them together, and sum the result.

The transpose of a matrix $A^T$ swaps rows for columns, so if $A$ is $n \times m$, then $A^T$ is $m \times n$. Matlab identifies the transpose using a prime:

$$A_{\text{transpose}} = A';$$

The product $C = AA^T$ is $n \times n$ and the elements along the diagonal $c_{ii}$ are $\sum_{l=1}^{n} a_{il}^2$. Incidentally, if column $i$ of $A$ had zero mean, then this would represent the variance of the $i$th column of $A$.

We should also think about matrix inverses. Inverses are only well defined for square matrices. Multiplying a matrix by its inverse is analogous to multiplying a scalar by its reciprocal to find $c/c = 1$. $AA^{-1} = I$ where $I$ is the identity matrix. $I$ contains ones along the diagonal, and zeros everywhere else. And importantly multiplying by $I$ is analogous to multiplying by $1$: the product of $I$ and any arbitrary matrix of the same size is $CI = IC = C$. In Matlab we can compute the inverse of a matrix $A$ using:
\texttt{A\_inverse = inv(A);} \\

If you’re multiplying an inverse by another matrix, then you can also use forward or backward slashes. See the Matlab help file on “slash” for information. An identity matrix can be created in Matlab using \texttt{eye}: \\
\texttt{identity=eye(3,3);}