Least Squares Tutorial

CS 310 — Fall 1999

1 Doing Least Squares in Matlab

In this lab we are going to look at taking a function of the form

$$c_1 + c_2 \sin(x) + c_3 \cos(x) + \cdots + c_{2m} \sin(mx) + c_{2m+1} \cos(mx)$$

and fitting it to a given set of data for some value of m. We'll first do an example by hand; then you'll copy a program that will do the work for you, and we will look at the behavior of the least squares fit.

Let's suppose we have the following 8 data points:

(0, 6)	(1, 4)	(2, 3)	(3, 5)
(4, 3)	(5, 4)	(6, -1)	(7, 2)

The first thing we need to do is enter this data into Matlab:

```
x = 0:7
y = [6 4 3 5 3 4 -1 2]
```

We're going to fit this data to a function of the form:

$$y = c_1 + c_2 \sin(x) + c_3 \cos(x)$$

which corresponds to using $m = 1$ in the general form. For each of our (x, y) pairs, we have an equation with the unknowns c_1, c_2 and c_3. For example, for the point (1,4) we have this equation:

$$4 = c_1 + c_2 \sin(1) + c_3 \cos(1)$$

or

$$4 = c_1 + 0.8415\, c_2 + 0.5403\, c_3$$

We have 8 points, so we have 8 equations. Since we have only 3 unknowns (c_1, c_2, and c_3), this system of equations probably does not have a solution. So, we will try to find the values of c_1, c_2 and c_3 that approximate the solution to the system in the least-squares sense.

The first thing we have to do is enter the equations into Matlab. For each equation, the coefficient of c_1 is always going to be 1, the coefficient of c_2 is the sine of the x value for the point, and the coefficient of c_3 is the cosine of x. The right-hand side is the y data value. So, we can create the matrix of coefficients and the right-hand side vector using these Matlab commands:

```
A = [ones(8,1), sin(x'), cos(x')]
b = y'
```

$$A = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 0.8415 & 0.5403 \\ 1 & 0.9093 & -0.4161 \\ 1 & 0.1411 & -0.9900 \\ 1 & -0.7568 & -0.6536 \\ 1 & -0.9589 & 0.2837 \\ 1 & -0.2794 & 0.9602 \\ 1 & 0.6570 & 0.7539 \end{bmatrix}$$

Look carefully at the matrix `A`. Where is the equation corresponding to the point $(1, 4)$ that we looked at before?

Notice that we took the transpose of `x` and `y` so that the row vectors would be switched to column vectors.

We're solving this equation:

$$Ac = b$$

where c contains the coefficients c_1, c_2, and c_3.

Now we'll solve this system for c_1, c_2, and c_3 in the least-squares sense. To do this, we will first get the QR-decomposition of A using the Matlab function `qr`. Then we will have

$$QRc = b$$

Recall that Q is an orthogonal matrix and R is "upper triangular".

Now we'll multiply both sides by Q^{-1}, which is the transpose of Q since Q is an orthogonal matrix (as discussed in lecture). This will give us

$$Rc = Q'b$$

Since R is upper-triangular, this system of equations can be solved using back substitution.

Doing this in Matlab:

```
[Q,R] = qr(A)
b = Q'*b
```

You might want to check that $A = QR$, by typing `Q*R` and seeing if the result is `A`.

The system of equations that we now have is $Rc = Q'b$ and looks like:

$$\begin{bmatrix} -2.8284 & -0.1958 & -0.5226 \\ 0 & -1.8758 & -0.1514 \\ 0 & 0 & 2.0365 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} -9.1924 \\ -0.2596 \\ -1.5515 \\ 0.7896 \\ -1.0399 \\ 0.1416 \\ -4.8327 \\ -1.9864 \end{bmatrix}$$

Of these 8 equations, 5 of them obviously cannot be satisfied, such as $0 = 0.7896$, the fourth equation. The magnitude of the right-hand sides of these equations tells us something about how poor our solution is.

The first three equations can be solved using back-substitution. Before we can do that, we first need to remove the bottom 5 rows of each side.

```
R1 = R(1:3,:)
b1 = b(1:3)
coef = R1\b1
```

The vector `coef` contains the values of the coefficients c_1, c_2 and c_3.

Now, let's plot the data together with our estimated solution:

```
xx = 0:.1:7;
yy = coef(1) + coef(2)*sin(xx) + coef(3)*cos(xx);
plot(x,y,'o',xx,yy);
```

As you can see, the curve does not go through any of the points. But, it does go as close as possible. If we were to use more coefficients, we could get a better fit.

To get a numerical measure of how well our function fits the data, we can compute the square root of the sum of the squares of the differences between the y-coordinate of each data point and the y-value predicted by the function. Another method of estimating the error is to look at the bottom 5 equations that were not satisfied and compute the square root of the sum of the squares of the right-hand sides. We can do each of these with Matlab as follows:

```
r1 = norm(y' - A*coef)
r2 = norm(b(4:8))
```

r1 and r2 are the same. It turns out that the two methods for estimating the error are equivalent. Note that r1 and r2 the norms (or lengths) of the residual vector.

If we use Matlab, like we are doing today, we can save ourselves some work. It turns out that if you ask Matlab to solve a linear system using Gaussian elimination (by typing A\b), and A has more rows than columns, Matlab will solve the system in the least-squares sense. You still need to set up A and b as done here, but you can avoid the QR decomposition. So, that's a real advantage we get using Matlab.

☞ Using the same data points, fit a function of the form

$$y = c_1 + c_2 \sin(x) + c_3 \cos(x) + c_4 \sin(2x) + c_5 \cos(2x)$$

and plot your function together with the data points. Does the fit look any better? What happened to the residual vector?

2 More Fitting Trig Functions to Data

Copy the file trigfit.m from the course Web page. Use your favorite editor to look at trigfit.m

trigfit.m generates a set of data and then fits it, for some value of m, with a trigonometric sum of the form discussed in the previous section:

$$c_1 + c_2 \sin(x) + c_3 \cos(x) + \cdots + c_{2m} \sin(mx) + c_{2m+1} \cos(mx)$$

Look in trigfit.m to see how the curve fitting is being done. The method is the same as used in Section 1, but it is a little more general.

The purpose of this exercise is to show that for some data sets and some approximating functions there are reasonable approximations, and for others there are poor approximations. You should experiment with different values of the parameter m, which is called modes in trigfit.m. Also, feel free to modify trigfit.m by changing the size of the data set, which is Ndata. You can also change the function used to generate the data.

For example, with function 1, for $m = 3$ the fit is rather good. With $m = 7$, you should see that the residual is smaller, but the function is less satisfying. This shows that including higher frequencies may not be a good idea.

Note that since we use a random vector to generate the data, no two successive runs will use the same data. If you want to use the same data over again, invoke the help facility to check on setting the seed for the function rand.

☞ Use function 5, $y = \sin(2\sin(x) + 0.2)$, with an increasing number of modes, starting with $m = 1$, to see how the data fit changes. Display the vector `coef` to find the coefficients for the higher modes. Can you explain what is happening?

If you want to re-run `trigfit` and save your old graphs, type `figure` before re-running `trigfit`. Matlab will open up a new plot window, and the next plot will appear in that window; this way, you won't overwrite your old plots.

☞ Do the same with function 8.

☞ Do the same with function 2. In this case, the fit does not improve with the addition of more modes. What happens to the values of `coef` at the higher modes in this case?

☞ As mentioned above, for function 3, the fit for $m = 3$ looks good, but for $m = 7$ the fit looks much worse. Yet, the residual is smaller when $m = 7$. The residual is smaller still for even larger values of m (try $m = 15$!) but the fit looks horrible. Explain what is happening.

☞ *(optional)* If you try to use too large a value for m, the program aborts with an error. Figure out why.