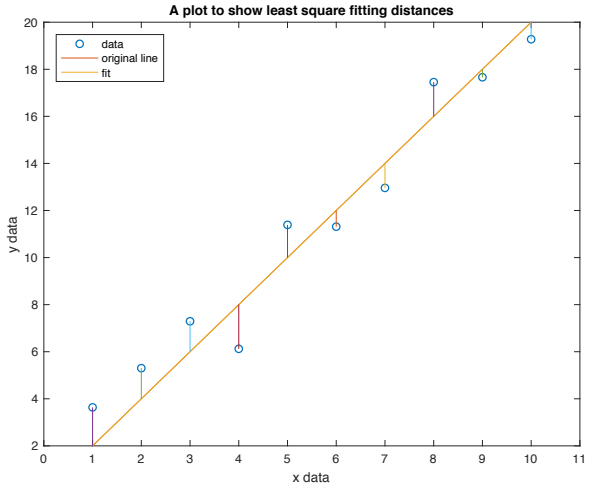
3/20 in Class – Least Square Fitting of Linear Data

This is as handed out in class 3/20. However, it seems too difficult as written, so I'm going to basically rewrite it for 3/22.

In class today, I showed that

$$\chi^2 = \sum_{k=1}^{N} (mx_k + b - y_k)^2$$

is the sum of the squares of the distances (in the y direction) from the fit to the data. Also sometimes called the sum of the squares of the errors.



1. I also showed an example of code I had written to test a least square calculation. Start at the white board in groups of two or three writing your structure plans (with algebra) for the following chunks of code: And while you're here, do (b) also.

   (a) Write code to generate a set of data that is offset from a known line by some known amount. Let's all start with the same line: $y = 2x + 0$ Write out the offset data to a file called `knownOffset.txt` You can do the same thing I did – offset every other data point by ± 0.5 units in y, or you can do your own. But you need to know the offset.

   (b) Calculate by hand what your sum of the squares (χ^2) will be.

   (c) Write code to read in that data file and then calculate the sum of the squares of the distances between the data and the original line. Call it `sumSquares.m`. Check to see that your code gives what you expect from your calculation by hand for the last part.

2. Now to find the line of best fit to some data, one way is the method of least squares: that is, finding the line that minimizes of the sum of the squares you just calculated. I showed that finding the minimum of χ^2 (by taking the first derivative with respect to m and b and setting them each equal to zero, then solving two equations for two unknowns) yields:

$$m = \frac{c_{xy} - c_x c_y}{c_{xx} - c_x c_x} \qquad b = \frac{c_{xx} c_y - c_x c_{xy}}{c_{xx} - c_x c_x}$$

where

$$c_x = \frac{1}{N}\sum_{k=1}^{N} x_k \qquad c_{xx} = \frac{1}{N}\sum_{k=1}^{N} x_k^2 \qquad c_y = \frac{1}{N}\sum_{k=1}^{N} y_k \qquad c_{xy} = \frac{1}{N}\sum_{k=1}^{N} x_k y_k$$

Go to the white board again with your group and write a structure plan for how to do that. Go into enough detail: will you use for loops? Can you vectorize it?

Then write code to do this. Call it `leastSquaresLine.m`

3. Run your code from Problem 2 on the data you generated in Problem 1 (hopefully in a file called `knownOffset.txt`.)

   Does your least fit algorithm get you back your original line? ($y = 2x$, so, $m = 2$ and $b = 0$)

   It should be close, but not exact. Why?

   Take the values you found for the minimum of m and b and plug them into your script from Problem 1. (`sumSquares.m`) and see if the new fit values make the χ^2 smaller. Does it? It should–why?

4. If you buy the Optimization Toolbox from MATLAB, there's a function called `lsqcurvefit` that does this. I did not require that for this class (and in fact neither I nor the lab computers have it), so let's do it another way, and it's also good practice writing functions.

   The command `fminsearch` finds the root(s) that minimize the value of any function.

   So all we have to do is write our own function that calculates χ^2 so that we can call `fminsearch` on the function. Do that. Call the function `chiSquared`. Hints: You may have to load the data here. I found it easier to pass the m and b to the function as one array variable, as in $p = [m, b]$.

   Then write a script that calls `fminsearch` on your `chiSquared` and get the values of the m and b that will minimize the sum of the squares. Use your same data set; load it in from `knownOffset.txt`

   Call this script `useFmin.m`. Include the function at the end of the file if you can. (Most of us can.)

5. At the end of class, email me whatever scripts (.m files, not .txt) you have finished from this worksheet.