MATH 341, Fall 2014: Least Squares

A fundamental problem in data analysis is fitting a line (or more generally, a linear or affine function) to a set of data points. Consider, for example, a set of k points $(x_1, y_1), (x_2, y_2), \ldots, (x_k, y_k)$ in the plane. Generally speaking, if $k > 2$ these points will not all lie on the same line, but which line best fits the data? By "best fits", we mean that the line comes as close as possible, in some sense, to the data points. The answer depends on how we quantify closeness of a line to a set of points. The most commonly used measure of the quality of the fit is as follows. If the equation of the line is $y = ax + b$, define the error in approximating the data by the line to be

$$E = \sum_{i=1}^{k} (ax_i + b - y_i)^2.$$

The main reason this measure is used is that the problem of finding constants a and b that minimize this error has a simple solution. The solution to this type of minimization problem is called a "least squares" solution.

Notice that E is a quadratic function of a and b and that the coefficients of a and b are positive, so E has a global minimum at a point where its partial derivatives are zero. (Keep in mind that in this minimization problem, the data points (x_i, y_i) are given and a and b are the unknowns.) Now

$$\frac{\partial E}{\partial a} = \sum_{i=1}^{k} 2x_i(ax_i + b - y_i) = 2\left(a\sum_{i=1}^{k} x_i^2 + b\sum_{i=1}^{k} x_i - \sum_{i=1}^{k} x_i y_i\right)$$

and

$$\frac{\partial E}{\partial b} = \sum_{i=1}^{k} 2(ax_i + b - y_i) = 2\left(a\sum_{i=1}^{k} x_i + bk - \sum_{i=1}^{k} y_i\right).$$

Thus both partial derivatives are zero if and only if

$$\begin{bmatrix} \sum_{i=1}^{k} x_i^2 & \sum_{i=1}^{k} x_i \\ \sum_{i=1}^{k} x_i & k \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{k} x_i y_i \\ \sum_{i=1}^{k} y_i \end{bmatrix}.$$

This is where linear algebra comes in. One can show that the determinant of the matrix above is positive (unless all the x_i are equal to each other), so a and b are uniquely determined by the data points. In fact, we could invert the matrix and write down a formula for a and b, but instead let's consider how to set up this problem as a linear algebra problem from the beginning and thus how to generalize this approach to higher-dimensional problems like fitting a plane to data points (x_i, y_i, z_i), etc.

First, let $\vec{\mathbf{c}} = \begin{bmatrix} a \\ b \end{bmatrix}$ and define the matrix and vector

$$X = \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_k & 1 \end{bmatrix}, \qquad \vec{\mathbf{y}} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_k \end{bmatrix}.$$

Then the coefficients of a line that went through all the data points would be a solution to the equation $X\vec{\mathbf{c}} = \vec{\mathbf{y}}$. Generally speaking, there is no solution to this system (k equations in 2 unknowns), so instead we want a vector $\vec{\mathbf{c}}$ that minimizes the error $E = |X\vec{\mathbf{c}} - \vec{\mathbf{y}}|^2$. (We could just as well define the error to be $|X\vec{\mathbf{c}} - \vec{\mathbf{y}}|$, since this quantity is minimized when E is.)

The linear system of equations we derived above to solve for $\vec{\mathbf{c}}$ can be written in terms of X and $\vec{\mathbf{y}}$ as

$$X^\top X\vec{\mathbf{c}} = X^\top\vec{\mathbf{y}}.$$

Therefore,

$$\vec{\mathbf{c}} = \left(X^\top X\right)^{-1} X^\top\vec{\mathbf{y}}.$$

Though we derived this system in a particular case, this is in fact the general solution to the least squares problem of minimizing $|X\vec{\mathbf{c}} - \vec{\mathbf{y}}|$ for a given matrix X and vector $\vec{\mathbf{y}}$. In order for there to be a unique solution (that is, for $\left(X^\top X\right)^{-1}$ to exist), X must have at least as many rows as columns (otherwise the system $X\vec{\mathbf{c}} = \vec{\mathbf{y}}$ has fewer equations than unknowns), and if X is an invertible square matrix then the expression above for $\vec{\mathbf{c}}$ reduces to $X^{-1}\vec{\mathbf{y}}$, so the interesting case here is when X has more rows then columns.

In MATLAB, you could of course find $\vec{\mathbf{c}}$ by typing `c = inv(X´*X)*X´*y`, but there is a simpler and better way: just type `c = X\y`. Indeed, if you typed `help slash` before, you saw that for non-square matrices the backslash operator solves the system in the least squares sense. Thus in general, to fit data with a linear or affine function, once you express the desired relationship between the data points in matrix form, the least squares solution is easy to obtain with MATLAB.