

§10.1 Least Squares

Least squares is a method of fitting observed data to a theoretical model function. In the general setting we are given a set of data

$$\begin{array}{c|c|c|c|c} x & x_0 & x_1 & \dots & x_n \\ \hline y & y_0 & y_1 & \dots & y_n \end{array}$$

and some class of functions, \mathcal{F} . The goal then is to find the “best” $f \in \mathcal{F}$ to fit to the data. Usually the class of functions \mathcal{F} will be determined by some small number of parameters; the number of parameters will be smaller (usually much smaller) than the number of data points. The theory here will be concerned with defining “best,” and examining methods for finding the “best” function to fit the data.

Linear Least Squares

The linear case is the easiest. In this case \mathcal{F} is the class of all linear functions. That is

$$\mathcal{F} = \{f(x) = ax + b \mid a, b \in \mathbb{R}\}.$$

We can now think of our job as drawing the best line through the data.

In the unlikely case that the data points $\{(x_i, y_i)\}_{i=0}^n$ are collinear, there is a clear best line through them. However, this is unlikely.

Suppose we have some candidate function $f(x) = ax + b$. Let the absolute error at the k^{th} data point be

$$|ax_k + b - y_k|$$

The total absolute error is the sum of all the errors:

$$\sum_{k=0}^n |ax_k + b - y_k|$$

One definition of the best line through the data points is that it is the one that minimizes this total absolute error. This is called the ℓ_1 approximation; this approximation cannot be solved with calculus and relies on other techniques.

Instead we look at another norm on the error. That is we seek to find the function $f(x) = ax + b$, where a, b minimize the ℓ_2 error function:

$$\phi(a, b) = \sum_{k=0}^n |ax_k + b - y_k|^2$$

We minimize this function with calculus. We set

$$0 = \frac{\partial \phi}{\partial a} = \sum_{k=0}^n 2x_k (ax_k + b - y_k)$$

$$0 = \frac{\partial \phi}{\partial b} = \sum_{k=0}^n 2(ax_k + b - y_k)$$

These are called the *normal equations*. Believe it or not these are two equations in two unknowns. They can be reduced to

$$\sum x_k^2 a + \sum x_k b = \sum x_k y_k$$

$$\sum x_k a + (n+1)b = \sum y_k$$

The solution is found by naïve Gaussian Elimination, and is ugly. Let

$$d_{11} = \sum x_k^2$$

$$d_{12} = d_{21} = \sum x_k$$

$$d_{22} = n+1$$

$$e_1 = \sum x_k y_k$$

$$e_2 = \sum y_k$$

We want to solve

$$d_{11}a + d_{12}b = e_1$$

$$d_{21}a + d_{22}b = e_2$$

Gaussian Elimination produces

$$a = \frac{d_{22}e_1 - d_{12}e_2}{d_{22}d_{11} - d_{12}d_{21}}$$

$$b = \frac{d_{11}e_2 - d_{21}e_1}{d_{22}d_{11} - d_{12}d_{21}}$$

The answer is not so enlightening as the means of finding the solution.

We should, for a moment, consider whether this is indeed the solution. Our calculations have only shown an extrema at this choice of (a, b) ; could it not be a maxima?

General Least Squares

We now suppose that \mathcal{F} is an $m+1$ dimensional class. That is, there is some linearly independent set of *basis functions* $\{g_j(x)\}_{j=0}^m$ such that

$$\mathcal{F} = \left\{ f(x) = \sum_j c_j g_j(x) \mid c_j \in \mathbb{R}, j = 0, 1, \dots, m \right\}$$

To find the best member of \mathcal{F} for a given set of data, we find the ℓ_2 approximation by minimizing the function

$$\phi(c_0, c_1, \dots, c_m) = \sum_{k=0}^n \left[\left(\sum_j c_j g_j(x_k) \right) - y_k \right]^2$$

Again we set partials to zero and solve

$$0 = \frac{\partial \phi}{\partial c_i} = \sum_{k=0}^n 2 \left[\left(\sum_j c_j g_j(x_k) \right) - y_k \right] g_i(x_k)$$

This can be rearranged to get

$$\sum_{j=0}^m \left[\sum_k g_j(x_k) g_i(x_k) \right] c_j = \sum_{k=0}^n y_k g_i(x_k)$$

If we now let

$$d_{ij} = \sum_{k=0}^n g_j(x_k) g_i(x_k), \quad e_i = \sum_{k=0}^n y_k g_i(x_k),$$

Then we have reduced the problem to the linear system (again, called the normal equations):

$$\begin{bmatrix} d_{00} & d_{01} & d_{02} & \cdots & d_{0m} \\ d_{10} & d_{11} & d_{12} & \cdots & d_{1m} \\ d_{20} & d_{21} & d_{22} & \cdots & d_{2m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ d_{m0} & d_{m1} & d_{m2} & \cdots & d_{mm} \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_m \end{bmatrix} = \begin{bmatrix} e_0 \\ e_1 \\ e_2 \\ \vdots \\ e_m \end{bmatrix} \quad (1)$$

The choice of the basis functions can affect how easy it is to solve this system. We explore this in §10.2. Note that we are talking about the basis $\{g_j\}_{j=0}^m$, and not exactly about the class of functions \mathcal{F} . A given class of functions will have more than one choice of basis functions.

For example, consider what would happen if the system of normal equations were diagonal. In this case, solving the system would be rather trivial.

Example 1. We consider the awfully chosen basis functions:

$$\begin{aligned} g_0(x) &= \left(\frac{\epsilon}{2} - 1 \right) x^2 - \frac{\epsilon}{2} x + 1 \\ g_1(x) &= x^3 + \left(\frac{\epsilon}{2} - 1 \right) (x^2 + x) + 1 \end{aligned}$$

where ϵ is small, around machine precision.

Suppose the data are given at the nodes $x_0 = 0, x_1 = 1, x_2 = -1$. We want to set up the normal equations, so we compute some of the d_{ij} . First we have to evaluate the basis functions at the nodes x_i . But this example was rigged to give:

$$\begin{aligned} g_0(x_0) &= 1, g_0(x_1) = 0, g_0(x_2) = \epsilon \\ g_1(x_0) &= 1, g_1(x_1) = \epsilon, g_1(x_2) = 0 \end{aligned}$$

After much work we find we want to solve

$$\begin{bmatrix} 1 + \epsilon^2 & 1 \\ 1 & 1 + \epsilon^2 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \end{bmatrix} = \begin{bmatrix} y_0 + \epsilon y_2 \\ y_0 + \epsilon y_1 \end{bmatrix}$$

However, the computer would only find this if it had infinite precision. Since it does not, and since ϵ is rather small, the computer thinks $\epsilon^2 = 0$, and so tries to solve the system

$$\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \end{bmatrix} = \begin{bmatrix} y_0 + \epsilon y_2 \\ y_0 + \epsilon y_1 \end{bmatrix}$$

When $y_1 \neq y_2$, this has no solution. Bummer.

This kind of thing is common in the method of least squares: the coefficients of the normal equations include terms like

$$g_i(x_k)g_j(x_k).$$

When the g_i are small at the nodes x_k , these coefficients can get really small, since we are squaring.

Now we draw a rough sketch of the basis functions. We find they do a pretty poor job of discriminating around all the nodes.

Example 2. We consider the case where $m = 0$, and $g_0(x) = \log x$. This is homework question 7.