Nonlinear function interpolation: Lagrange

C. Alex Simpkins

November 16, 2007

0.1 Nonlinear interpolation

0.1.1 Lagrange interpolation

In nonlinear interpolation, we can fit an (n-1)st order curve exactly through n data points (this is the lowest order curve, since an [n-1]st order polynomial will have exactly n parameters). This technique is especially useful in cases with very few data points. For large numbers of data points, above a few, it is more appropriate to use some form of cubic spline interpolation where a curve is fit through each pair of points. Why exactly that is a problem will be explored in detail later. For now let us begin by considering a polynomial

$$f(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + ... + a_1 x + a_0. \tag{1}$$

This is simply the equation for a polynomial. Now we want to constrain this polynomial equation to pass exactly through our n points

$$f(x_i) = y_i, i = 1, 2, ...n. \tag{2}$$

Our unknowns are the a constants. We have n points (x, y pairs), and n unknowns (the a's). We can solve (**??**) for the a's by substituting (**??**) into (**??**). This results in

$$\begin{aligned} a_{n-1}x_1^{n-1} + a_{n-2}x_1^{n-2} + ...a_1 x_1 + a_0 &= y_1 \\ a_{n-1}x_2^{n-1} + a_{n-2}x_2^{n-2} + ...a_1 x_2 + a_0 &= y_2 \\ ... &= ... \\ ... &= ... \\ a_{n-1}x_n^{n-1} + a_{n-2}x_n^{n-2} + ...a_1 x_n + a_0 &= y_n \end{aligned} \tag{3}$$

Or, writing this more compactly, we can write simply

$$a_{n-1}x_i^{n-1} + a_{n-2}x_i^{n-2} + ...a_1 x_i + a_0 = y_i, i = 1, 2, ...n. \tag{4}$$

Now we have n algebraic equations in n unknowns, which is a perfectly constrained problem. We could simply solve this problem using Gaussian elimination or another algorithm, but the problem becomes ill-conditioned for about $n > 5$. This makes it difficult to solve these equations for the a's accurately.

Another approach to solving for the a's is to solve them in such a way as they are linear combinations of the y's.

$$f(x) = \sum_{k=1}^{n} y_k L_k(x) \tag{5}$$

Where the $L_k(x)$'s are polynomials of degree $n-1$. Recall that an equation for a polynomial P is simply

$$P(x) = \sum_{k=0}^{n-1} a_k x^k \tag{6}$$

and we want the polynomial to equal the y data points at those values of x. Thus, we can make

$$L_j(x_i) = \delta_{ij}, i = 1, 2, ...n, j = 1, 2, ...n \tag{7}$$

where δ is defined as

$$\delta_{ij} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases} \tag{8}$$

From algebra we recall that *any* polynomial of degree n can be factored into a constant multiple of n factors $(x - x_p)$, where x_p are the zeros of the polynomial. Since $L_j(x_i)$ is a polynomial of degree $n-1$ with known zeros $(i \neq j)$, it has the form

$$L_j(x) = C_j(x - x_1)(x - x_2)...(x - x_n) \tag{9}$$

We want to find C_j where $i = j$, so that $L_j(x_j) = 1$

$$1 = C_j(x_j - x_1)(x_j - x_2)...(x_j - x_n) \tag{10}$$

Which we solve by dividing both sides by $(x_j - x_1)(x_j - x_2)...(x_j - x_n)$

$$C_j = \frac{1}{(x_j - x_1)(x_j - x_2)...(x_j - x_n)} \tag{11}$$

Now we substitute (**??**) into (**??**) to arrive at

$$L_j = \frac{(x - x_1)(x - x_2)...(x - x_n)}{(x_j - x_1)(x_j - x_2)...(x_j - x_n)} \tag{12}$$

Thus we can write out the solution in two parts

$$L_i = \prod_{j=0, j\neq i}^{n-1} \frac{(x - x_j)}{x_i - x_j} \tag{13}$$

$$y(x) = \sum_{i=0}^{n-1} L_i(x) f(x_i) \tag{14}$$

This reduces to $y = f(x_i)$ at all i vertex points. One very important note to make is that the $i \neq j$ in the $L(x)$ equation. If it did, one would have a $1/0$ situation, which may cause numerical instability, and certainly would not pass through the data points exactly.