Interpolation

1. What is interpolation?

Interpolation a process which, given function values (plus perhaps noise) for some set of given locations, can return estimates of the function values at any other location.

Here I will discuss the case that we want the interpolating function to pass exactly through the given points. Approximate interpolation is closely related, but is better thought of in the context of curve-fitting. I will also use a linear framework, by which I mean that the interpolation function can be written as a linear sum of the sample values:

$$\hat{f}(x) = \sum_i W(x, x_i) f(x_i) \tag{1}$$

(Not to be confused with linear interpolation!).

Thus we can think of the interpolation as determining not just values at a bunch of new points, but as a whole continuous function. This is useful for example if the interpolating function is integrable, meaning we can use the same interpolation to determine integrals.

2. Linear Interpolation

Linear interpolation is the simplest case. It has a number of drawbacks, but it illustrates several principles usefully.

The basic idea is that if you have some equally spaced set $\{x_i\}$ with known function values $\{f(x_i)\}$, and you want to interpolate $f(x)$ to some value x, you find which paid of x_i and x_{i+1} brackets x, and then:

$$\hat{f}(x) = f(x_i) + [f(x_{i+1}) - f(x_i)] \frac{x - x_i}{x_{i+1} - x_i} \tag{2}$$

This is pretty intuitive. It corresponds to:

$$W(x, x_i) = 1 - |x - x_i| \tag{3}$$

for $-1 < (x - x_i) < 1$, and zero otherwise. That is, it is the triangular function.

In addition, $\hat{f}(x)$ it can also be thought of as a sum of basis functions B_i with coefficients a_i:

$$\hat{f}(x) = \sum_i a_i B_i(x) \tag{4}$$

In this case $B_i(x) = W(x, x_i)$ but this does not hold in general!

3. Fitting Basis Coefficients

Let's use this approach to think a little more generally about fitting an interpolating function with a basis set. We have a set of points $f(x_j)$, and a set of equations to satisfy:

$$f(x_j) = \sum_i a_i B_i(x_j) \tag{5}$$

This can be written as a matrix equation:

$$\vec{f} = \mathbf{B} \cdot \vec{a} \tag{6}$$

which can be solved for $\vec{a}$, e.g.

$$\vec{a} = \mathbf{B}^{-1} \cdot \vec{f} \tag{7}$$

or alternatively written:

$$a_i = \sum_j B_{ij}^{-1} f(x_j) \tag{8}$$

This inverse exists as long as the basis vectors are independent and have support where your samples are.

Then the interpolating function is:

$$\begin{aligned} \hat{f}(x) &= \sum_i a_i B_i(x) \\ &= \sum_i B_i(x) \sum_j B_{ij}^{-1} f(x_j) \end{aligned} \tag{9}$$

That is to say,

$$W(x, x_i) = B_i(x) \sum_j B_{ij}^{-1} \tag{10}$$

In the case of linear interpolation, the matrix $B_i(x_j)$ is trivial, it is just the identity matrix, so $\vec{a} = \vec{f}$. That is why the weights $W(x, x_i)$ are equal to the basis functions $B_i(x)$. But this need not be the case.

4. Polynomial Basis Interpolation

Another possible interpolation basis is the space of polynomials.

$$\begin{aligned} B_0(x) &= 1 \\ B_1(x) &= x \\ B_2(x) &= x^2 \\ &\dots \end{aligned} \tag{11}$$

This interpolation basis leads to the same equation as above, but now the matrix $B_{ij} = B_i(x_j)$ is a dense matrix.

A brief note on numerical stability. Often your independent variable's natural units will not be of order unity. E.g. maybe you are doing something where a wavelength of light in the optical is reported in Angstroms, so $\lambda \sim 5000$. In this case, you do not want to use the raw independent variable for a high order fit. If I fit a 9th order polynomial in this case, then $\lambda^7 \sim 10^{33}$, which will overflow 32-bit bounds. It will also lead to a poorly conditioned K_{ij} matrix. You can avoid this specific case with 64-bit precision, but this will be slower and take more memory, and in any case the problem still can occur under the right circumstances.

Without any penalty, you can perform the same interpolation by first rescaling $x \to x'$ where x' is limited to values of order unity, so that powers of x' stay closer to unity. There might under certain circumstances be a greater chance of *underflow* in this case, but not necessarily, and in any case there is a greater dynamic range below zero than above, because of the existence of subnormal numbers.

As long as you are rescaling the x limits, a good choice is to rescale from -1 to 1. In that case you can also change your polynomial basis to Legendre polynomials, which are complete basis over that range. This choice will lead to better conditioned matrices.

Although you can do the matrix inversion for the polynomial basis numerically, there is in fact an analytic solution for the weights, which are as it turns out the Lagrange polynomials:

$$W(x, x_i) = \prod_{j \neq i} \frac{x - x_j}{x_i - x_j} \tag{12}$$

You can kind of guess these from the fact that they need to have the property that at each data point all of the polynomials except for one should have a node.

Note that at higher orders, the polynomial interpolator becomes pretty poorly behaved. But the polynomial basis will be an important one as we think about methods in integration and differentiation. The particular choice of Legendre polynomials also will form the basis later for the method of Gaussian quadrature.

5. Fourier Basis Interpolation

We can also consider the Fourier basis. This—or methods close to it—can be appropriate, especially if you are interpolating equally spaced points.

If you consider a continuous set of basis functions:

$$B_\omega = \frac{1}{\sqrt{2\pi}} \exp(i\omega x) \tag{13}$$

and for uniform sampling, with $|\omega|$ limited by the sampling rate, you can find:

$$W(x, x_i) = \frac{1}{2\pi} \int_{-\pi}^{\pi} \mathrm{d}\omega \exp\left[i\omega(x - x_i)\right] = \frac{\sin\left[\pi(x - x_i)\right]}{\pi(x - x_i)} = \mathrm{sinc}(x - x_i) \tag{14}$$

Under certain (rarely satisfied) conditions this set of weights is *perfect*. If the underlying function has no power in its Fourier spectrum beyond the Nyquist frequency, and your grid is very large (well . . . infinite) then you have fully determined the function with your sampling and the interpolation is exact. Note that there is also a version of this interpolation for a finite grid, using the discrete Fourier basis, which isn't quite as restrictive.

A more commonly used approach, called Lanczos interpolation or resampling, is to modify this set of interpolation weights slightly:

$$W(x, x_i) = \mathrm{sinc}(x - x_i)\,\mathrm{sinc}\left[(x - x_i)/a\right], \tag{15}$$

where typically $a = 2, 3$, or 4 (and is always an integer) and $W(x, x_i) = 0$ for $|x - x_i| > a$.