Chapter 4. Interpolation

Suppose we have a collection of values of a function

x	x_0	x_1	x_2	x_3	...
y	y_0	y_1	y_2	y_3	...

and we want to find a general formula for $f(x)$ so that $f(x_i) = y_i$ (and hopefully if the data came from a nice function that we have $f(x)$ close to the original function away from the x_i).

Why? We don't know the function.

The function is very slow to evaluate.

The y values are contaminated by measurement or computation errors.

We say the x_i are <u>nodes</u> and that a polynomial p <u>interpolates</u> the table if $p(x_i) = y_i$ for all i.

Example. If there are two nodes, a line segment interpolates the data.

Lagrange polynomials.

Suppose we had a collection of $\ell_i(x)$ so that

$$\ell_i(x_j) = \delta_{ij} = \begin{cases} 1 \text{ if } i=j \\ 0 \text{ if } i \neq j. \end{cases}$$

Then it would be easy to construct an interpolating polynomial by letting

$$p(x) = \sum y_i \ell_i(x).$$

It's not so hard to see that

$$\tilde{\ell}_i(x) = (x-x_0)(x-x_1)\cdots\widehat{(x-x_i)}\cdots(x-x_n)$$

(the hat means $(x-x_i)$ is missing) is zero at all x_j with $j \neq i$. To make $\ell_i(x_i)=1$, we just divide by $\tilde{\ell}_i(x_i)$ to get

$$\ell_i(x) = \frac{(x-x_0)}{(x_i-x_0)}\frac{(x-x_1)}{(x_i-x_1)}\cdots\frac{\widehat{(x-x_i)}}{\widehat{(x_i-x_i)}}\cdots\frac{(x-x_n)}{(x_i-x_n)}$$

Example. Find the polynomial interpolating

x	1	2	3
y	2	5	1

Example. <Interpolating polynomial.nb>

This proves that interpolating polynomials exist, of course.

Newton Polynomials.

We now give an alternate derivation for the interpolating polynomial.

Suppose $p_n(x)$ interpolates $x_0, \ldots, x_n$, and we want to extend it to interpolate x_{n+1}. We can write

$$p_{n+1}(x) = p_n(x) + c(x-x_0)\cdots(x-x_n)$$

where we must solve for c (and we need the $(x-x_i)$ stuff to make sure that we still interpolate $x_0, \ldots, x_n$).

We can solve for c by taking

$$y_{n+1} = p_n(x_{n+1}) + c(x_{n+1}-x_0)\cdots(x_{n+1}-x_n)$$

or

$$c = \frac{y_{n+1} - p_n(x_{n+1})}{(x_{n+1}-x_0)\cdots(x_{n+1}-x_n)}.$$

We can write Newton polynomials in <u>nested form</u> by factoring out common $(x-x_i)$ terms. Observe the example

x	0	1	-1	2	-2
y	-5	-3	-15	-39	-9

(Do the example).

We can then write

$$p_n(x) = \sum a_i \prod_{j=0}^{i-1} (x-x_j)$$

where we will then solve for the a_i.

The polynomials

$$\pi_i(x) = \prod_{j=0}^{i-1} (x-x_j)$$

are called Newton polynomials.

Notice that in nested form we can evaluate the Newton polynomial very quickly. We have

$$p(x) = a_0 + (x-x_0)(a_1 + (x-x_1)(a_2 + \dots (x-x_{n-1})a_n).$$

so we work from inside to outside

$$v_0 = a_n$$
$$v_1 = v_0(t - x_{n-1}) + a_{n-1}$$
$$v_2 = v_1(t - x_{n-2}) + a_{n-2}$$
$$\vdots$$
$$v_n = v_n(t - x_0) + a_0$$

and $v_n = p(t)$. This is easy to code, so it's desirable.

We still need to find an efficient way to solve for the a_i.

Here's the trick: we evaluate the Newton form at each of the x_i to get a set of equations involving the a_i.

$$p(x_0) = a_0$$

$$p(x_1) = a_0 + (x_1 - x_0)a_1$$

$$p(x_2) = a_0 + (x_1 - x_0)a_1 + (x_2 - x_1)(x_2 - x_0)a_2$$

$$\vdots$$

$$p(x_n) = a_0 + \cdots (x_n - x_0)\cdots(x_n - x_{n-1})a_n.$$

Notice that this system has a unique solution, and that each a_k depends only on the values of $p(x_0), \ldots, p(x_k)$. In fact, if we think of these p values as agreeing with some $f(x)$, we write

$$a_k = f[x_0, \ldots, x_k].$$

where

$f[x_0, \ldots, x_k] :=$ the divided difference of order k for f

To work out the formula for the divided differences, we start solving the system:

$$a_0 = f(x_0)$$

$$a_1 = \frac{f(x_1) - a_0}{x_1 - x_0} = \frac{f(x_1) - f(x_0)}{x_1 - x_0}$$

$$a_2 = \left(f(x_2) - a_0 - (x_2 - x_0)a_1\right) \Big/ (x_2 - x_1)(x_2 - x_0)$$

$$= \frac{\dfrac{f(x_2)}{x_2 - x_1} - \dfrac{f(x_0)}{x_2 - x_1} - \dfrac{f(x_1) - f(x_0)}{x_2 - x_1}\dfrac{(x_2 - x_0)}{(x_1 - x_0)}}{x_2 - x_0}$$

$$= \frac{\dfrac{f(x_2) - f(x_1)}{x_2 - x_1} + \dfrac{f(x_1) - f(x_0)}{x_2 - x_1} - \dfrac{f(x_1) - f(x_0)}{x_1 - x_0}\dfrac{(x_2 - x_0)}{(x_2 - x_1)}}{x_2 - x_0}$$

We now want to analyze

$$f(x_1)-f(x_0)\left[\frac{1}{x_2-x_1}-\frac{x_2-x_0}{(x_1-x_0)(x_2-x_1)}\right]$$

$$= f(x_1)-f(x_0)\left[\frac{(x_1-x_0)-(x_2-x_0)}{(x_2-x_1)(x_1-x_0)}\right]$$

$$= f(x_1)-f(x_0)\left[\frac{(x_1-x_2)}{(x_2-x_1)(x_1-x_0)}\right]$$

$$= -\frac{f(x_1)-f(x_0)}{x_1-x_0}$$

We finally get

$$a_2 = \frac{\dfrac{f(x_2)-f(x_1)}{x_2-x_1}-\dfrac{f(x_1)-f(x_0)}{x_1-x_0}}{x_2-x_0}.$$

Example. Compute the a_i for

x	1	-4	0
$f(x)$	3	13	23

$(a_0 = 3,\ a_1 = -2,\ a_2 = 7)$

We can then write

$$p_n(x) = \sum_{i=0}^{n} \left(f[x_0, \ldots, x_i] \prod_{j=0}^{i-1} (x - x_j) \right)$$

if we're willing to let $\prod_{j=0}^{-1} (x - x_j) = 1$.

Of course, we can solve this for the k-th divided difference in terms of the previous ones

$$p_n(x_k) = f[x_0, \ldots, x_k] \prod_{j=0}^{k-1} (x_k - x_j) + \sum_{i=0}^{k-1} f[x_0, \ldots, x_i] \prod_{j=0}^{i-1} (x_k - x_j).$$

or

$$f[x_0, \ldots, x_k] = \frac{f(x_k) - \sum_{i=0}^{k-1} f[x_0, \ldots, x_i] \prod_{j=0}^{i-1} (x_k - x_j)}{\prod_{j=0}^{k-1} (x_k - x_j)}.$$

Of course, we could use this algorithm to compute the $f[\,]$'s. But we can find a better one with a little bit of algebra.

Theorem. $f[x_0,\dots,x_k] = \dfrac{f[x_1,\dots,x_k]-f[x_0,\dots,x_{k-1}]}{x_k - x_0}$.

Let p_k be the interpolating polynomial for $x_0,\dots,x_k$, p_{k-1} be the interpolating poly. for $x_0,\dots,x_{k-1}$ and q be the interpolating poly. for $x_1,\dots,x_k$. We claim

$$p_k(x) = q(x) + \frac{x-x_k}{x_k-x_0}\left[q(x)-p_{k-1}(x)\right].$$

To see this, observe that each side is a polynomial of degree k. Further, for x_i with $1 \le i \le k-1$, we get

$$f(x_i) \stackrel{?}{=} f(x_i) + \frac{x_i-x_k}{x_k-x_0}\left[f(x_i)-f(x_i)\right] \checkmark$$

Evaluating at x_0, we get

$$p_k(x_0) = q(x_0) + \frac{x_0 - x_k}{x_k - x_0}[q(x_0) - p_{k-1}(x_0)]$$

$$= +p_{k-1}(x_0)$$

$$f(x_0) = f(x_0). \checkmark$$

and evaluating at x_k we get

$$p_k(x_k) = q(x_k) + \frac{x_k - x_k}{x_k - x_0}[q(x_k) - p_{k-1}(x_k)].$$

$$f(x_k) = f(x_k).$$

Since these degree K polys agree at K+1 points they are the same poly.

Now take the coefficient of x^k on both sides. Since there's only one term in Newton form with an x^k in p_k, this is

$$f[x_0,\dots,x_k] = \underbrace{0}_{q \text{ is degree } k-1} + \underbrace{\frac{f[x_1,\dots,x_k] - f[x_0,\dots,x_{k-1}]}{x_k - x_0}}_{\text{coeff of } x^{k-1} \text{ in } q}$$

coeff of x^{k-1} in p_{k-1}

Symmetry in entries.

Algorithm

$f[x_0]$

$f[x_1]$ $\quad f[x_0,x_1]$

$f[x_2]$ $\quad f[x_1,x_2]$ $\quad f[x_0,x_1,x_2]$

Vandermonde matrix.

Chebyshev

$T_0 = 1,\ T_1 = x$

$T_i = 2x\,T_{i-1} - T_{i-2}$

Having introduced the polynomials

$$1, x, x^2, \ldots\ldots$$

and

$$1, x, 2x^2-1, 4x^3-3x, 8x^4-8x^2+1, 16x^5-20x^3+5x$$

as possible basis functions for interpolating data, we suppose that in general we take

$$p_0(x), p_1(x), p_2(x), \ldots\ldots$$

How should we pick among such bases?

As a linear algebra problem, we have

$$\begin{bmatrix} p_0(x_0) & p_1(x_0) & p_2(x_0) & \cdots & p_k(x_0) \\ p_0(x_1) & p_1(x_1) & p_2(x_1) & \cdots & p_k(x_1) \\ \vdots & & & & \\ p_0(x_k) & p_1(x_k) & p_2(x_k) & \cdots & p_k(x_k) \end{bmatrix} \begin{bmatrix} c_0 \\ \\ \\ c_k \end{bmatrix} = \begin{bmatrix} y_0 \\ \\ \\ y_k \end{bmatrix}$$

The k column vectors of this matrix are just the vector formed by p_i evaluated at the nodes $x_0, \ldots, x_k$ and the inner product of two such vectors is

$$\sum_{\ell=0}^{k} p_i(x_\ell)\, p_j(x_\ell) \approx \ell \int_{x_0}^{x_\ell} p_i(x)\, p_j(x)\, dx$$

We see that the matrix will approach orthogonality if the integrals

$$\langle p_i, p_j \rangle = \int_{x_0}^{x_e} p_i\, p_j\, dx$$

vanish. (This is called the L^2 inner product on functions, and we are asking that the basis functions be L^2-orthogonal.)

The T_i are L^2-orthogonal.

We note there are many other orthogonal basis functions.

Remark. Given some values of $f(x)$, we can as easily interpolate $f^{-1}(x)$ as $f(x)$ by switching the x_i and y_i. This is usually used to guess a root of a function from values.

Example.

x	1	2	3
y	0.1	0.4	-0.2

We invert to get

-0.2	3	$\frac{-2}{0.3} = -\frac{20}{3}$	$\frac{10}{0.6} = \frac{100}{6}$
0.1	1		
0.4	2	$\frac{1}{0.3} = \frac{10}{3}$	

So $a_0 = 3$, $a_1 = -\frac{20}{3}$, $a_2 = \frac{100}{6}$ and

we have

$$p(x) = 3 + (x+0.2)\left(-\frac{20}{3} + (x-0.1)\left(\frac{100}{6}\right)\right)$$

$$= 3 - \frac{4}{3} + \frac{2}{10}\cdot\left(-\frac{1}{10}\right)\left(\frac{100}{6}\right)$$

$$= 3 - \frac{4}{3} - \frac{1}{3} = 1\,1/3. \quad ?$$

This example doesn't seem very good!

< inverse_interpolation.nb >