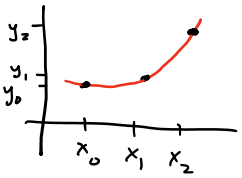
Newton's Divided Difference Method

Thursday, September 17, 2020 7:32 PM

Alternative algorithm to basic Lagrange interpolation or Barycentric version

A classic but it was especially important for doing calculations by hand, less important these days.

Same setup: nodes $\{x_0, x_1, \dots, x_n\}$ w/ values $\{y_0, y_1, \dots, y_n\}$



From last time: we can write the n degree interpolating polynomial as

$$p_n(x) = \sum_{k=0}^{n} y_k L_{n,k}(x), \qquad L_{n,k}(x) = \prod_{\substack{i=0 \\ i \neq k}}^{n} \frac{(x - x_i)}{(x_k - x_i)}$$

All we're going to do is write p_n differently,

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

"Newton's form"

It's not immediately obvious this is possible. The idea is that a_0 is determined by y_0 since $p_n(x_0) = a_0 + 0 + 0 + 0 + \dots + 0 = y_0$

Then, note $p_n(x_1) = a_0 + a_1(x_1 - x_0) + 0 + 0 + \dots + 0 = y_1$

(a_0: we already determined a_0)

so $a_1 = \dfrac{y_1 - a_0}{x_1 - x_0} = \dfrac{y_1 - y_0}{x_1 - x_0}$

⚠ computation order matters! Do a_0 first, then a_1

... and so on. We're going to formalize this now.

We'll need notation:

Square brackets tell us it's a divided difference

- the 0^{th} divided difference is $f[x_i] := y_i$ (and $y_i = f(x_i)$ often) — $n+1$ of these, $i = 0, 1, \dots, n$
- the 1^{st} divided difference is $f[x_i, x_{i+1}] = \dfrac{f[x_{i+1}] - f[x_i]}{x_{i+1} - x_i}$ — n of these, $i = 0, 1, \dots, n-1$
- the 2^{nd} divided difference is $f[x_i, x_{i+1}, x_{i+2}] = \dfrac{f[x_{i+1}, x_{i+2}] - f[x_i, x_{i+1}]}{x_{i+2} - x_i}$ — $n-1$ of these, $i = 0, \dots, n-2$
- the k^{th} divided difference is $f[x_i, x_{i+1}, \dots, x_{i+k}] = \dfrac{f[x_{i+1}, \dots, x_{i+k}] - f[x_i, \dots, x_{i+k-1}]}{x_{i+k} - x_i}$ — $n-k$ of these, $i = 0, \dots, n-k$

· stop at the n^{th} divided difference

1 of these, $i=0$

$$f[x_0, x_1, \dots, x_n] = \frac{f[x_1, x_2, \dots, x_n] - f[x_0, x_1, \dots, x_{n-1}]}{x_n - x_0}$$

Back to our polynomial

$$p_n(x) = a_0 + a_1(x-x_0) + a_2(x-x_0)(x-x_1) + \dots + a_n(x-x_0)(x-x_1)\cdots(x-x_n)$$

$a_0 = f[x_0]$, $a_1 = f[x_0, x_1]$ (we proved above), $a_2 = f[x_0, x_1, x_2]$, $a_n = f[x_0, x_1, \dots, x_n]$ (can prove but it's messy)

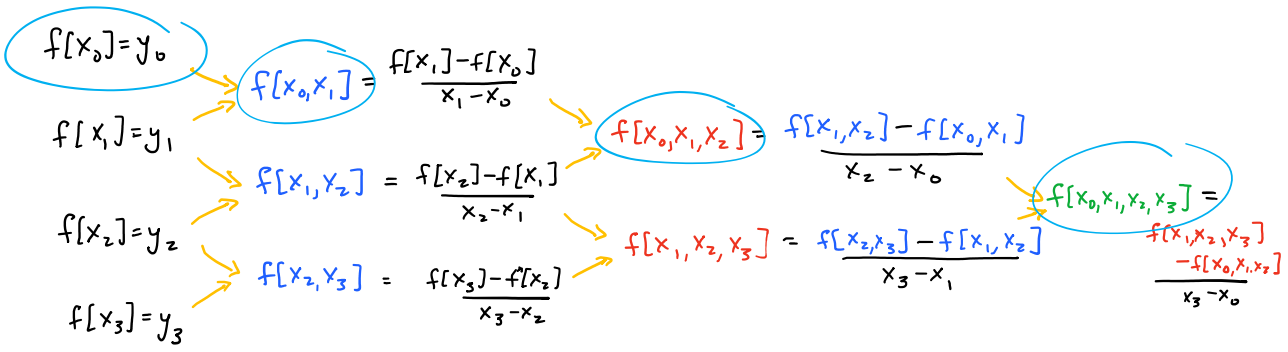
So

$$p_n(x) = f[x_0] + \sum_{k=1}^{n} f[x_0, x_1, \dots, x_k] \cdot (x-x_0)(x-x_1)\cdots(x-x_{k-1})$$

* note: we didn't need to assume $x_0 < x_1 < x_2 < \dots$ so you are free to re-order the points

How to compute divided differences

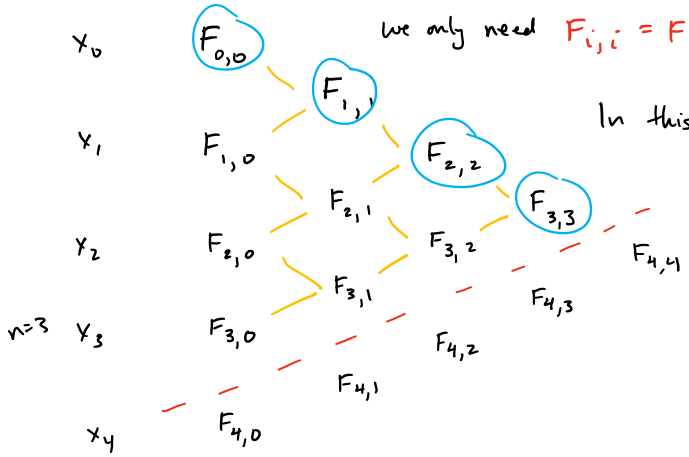
Ex. w/ $n=3$



We only need the terms circled, but to compute these, we have to generate all the previous terms

Using the shorthand notation $F_{i,j} = F[x_{i-j}, \dots, x_i]$ for $i = 0, 1, \dots, n$, $j = 0, 1, \dots, i$

we only need $F_{i,i} = F[x_0, x_1, \dots, x_i]$ for $i = 0, 1, \dots, n$



In this notation, $p_n(x) = F_{0,0} + \sum_{i=1}^{n} F_{i,i} \cdot \prod_{j=0}^{i-1} (x - x_j)$

Note: Suppose we decide to add a new interpolation point x_{n+1}. Then we can easily update the table of coefficients.

i.e., "Algo 3.2"

NEWTON DIVIDED DIFFERENCES

① For $i = 1, 2, \dots, n$

For $j = 1, 2, \dots, i$

$$F_{i,j} = \frac{F_{i,j-1} - F_{i-1,j-1}}{x_i - x_{i-j}}$$

② then to evaluate $p_n(x)$,

use $p_n(x) = F_{0,0} + \sum_{i=1}^{n} F_{i,i} \cdot \prod_{j=0}^{i-1} (x - x_j)$

Computational Complexity

① two "for" loops,
$O(1)$ computation at the innermost level,
repeat i times,
for $i = 1, 2, \dots, n$

So Flop count is $\sum_{i=1}^{n} i$ } you may recall from Calc. I.

("floating point operation")

$$\text{So} = \frac{n(n+1)}{2} = O(n^2)$$

The trick: $1 + 2 + 3 + 4 + 5 = 3 + 6 + 6$ ($\frac{n-1}{2}$ of these terms)

$n = 5$ (odd)

$6 = n+1$

$$= \frac{(n+1)}{2} + \frac{(n-1)}{2} \cdot (n+1)$$

$$= \boxed{\frac{n \cdot (n+1)}{2}}$$

② Evaluating at a point x

It looks like $\sum_{i=1}^{n} i$ again

but in fact it is $\sum_{i=1}^{n} 1 = O(n)$

because we can save values $\prod_{j=0}^{i-1} (x - x_j) = (x - x_{i-1}) \cdot \underbrace{\prod_{j=0}^{i-2} (x - x_j)}_{\text{already computed.}}$

$n = 6$ (even)

$$1 + 2 + 3 + 4 + 5 + 6 = \underbrace{7 + 7 + 7}_{n/2} = \frac{n(n+1)}{2}$$

Minor details (see book ch. 3.3)

① Since $f[x_0, x_1] = \frac{f(x_1) - f(x_0)}{x_1 - x_0}$, then if f is differentiable, the Mean Value Thm

$\Rightarrow \exists\, \xi \in (x_0, x_1)$ s.t. $f[x_0, x_1] = f'(\xi)$

and in fact a more general statement is true:

Thm 3.6 If $f \in C^n([a,b])$ and $\{x_0, \dots, x_n\} \subseteq [a,b]$ are distinct, then $\exists\, \xi \in (a,b)$

s.t. $f[x_0, x_1, \dots, x_n] = \frac{f^{(n)}(\xi)}{n!}$

② As for Lagrange and Barycentric formulas, if $\{x_0, x_1, \dots, x_n\}$ are uniformly spaced, i.e.,

$x_1 = x_0 + h$

$x_2 = x_0 + 2 \cdot h$ ... $x_n = x_0 + n \cdot h$

then things simplify.

To write $x - x_i$, let $x = x_0 + s \cdot h$ (s is probably not an integer)

then $x - x_i = (s - i) \cdot h$ and we can simplify w/ binomial coefficients

Combined with our forward difference operator Δ

where $\Delta y_0 = y_1 - y_0$, $\Delta^2 y_0 = \Delta y_1 - \Delta y_0$

then we get a particularly compact formula

$$p_n(x) = y_0 + \sum_{i=1}^{n} \binom{s}{i} \Delta^i y_0 \quad \text{where } s = \frac{x - x_0}{h}$$

[For uniform spacing]

(3) There's a backward differences formula

Not too important for this class, but let's define the notation in case we see it again

Recall, if (x_n) is a sequence, the

forward difference $\Delta x_n := x_{n+1} - x_n$

Now define the

backward difference $\nabla x_n := x_n - x_{n-1}$

and $\Delta^2 x_n = \Delta(\Delta x_n)$, $\nabla^2 x_n = \nabla(\nabla x_n)$

(4) How to use in practice

We might have a big, precalculated table of $F_{i,j}$

To evaluate at x, if x is close to x_0 or x_1, then even low degree interpolants will be accurate, and we can compute $p_k(x)$ for $k < n$ and save time.

If x is close to x_{n-1} or x_n, we can do the backwards version and again evaluate $p_k(x)$ in $O(k)$ time

If x is close to $x_{n/2}$, the above schemes to compute $p_k(x)$ are not efficient ... some other schemes are sometimes useful

ex: Stirling's method