Lagrange interpolation

We seek an n-degree polynomial $P_n(x)$ which interpolates the data points $(x_0,y_0), (x_1,y_1), \dots (x_n,y_n)$.

Lagrange interpolation constructs P_n as:

$$P_n(x) = y_0 l_0(x) + y_1 l_1(x) + \dots + y_n l_n(x)$$

Each of the $l_i(x)$'s is an n-degree polynomial, which equals zero at every x_j $(j \neq i)$, while $l_i(x_i) = 1$.

We saw that this can be constructed as:

$$l_i(x) = \frac{(x-x_1)\cdots(x-x_{i-1})(x-x_{i+1})\cdots(x-x_n)}{(x_i-x_1)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_n)} = \frac{\prod_{j \neq i}(x-x_j)}{\prod_{j \neq i}(x_i-x_j)}$$

from this definition it is obvious that

$$l_i(x_j) = \begin{cases} 1, & \text{if } i=j \\ 0, & \text{if } i \neq j \end{cases}$$

Let us evaluate this approach, as we did with the Vandermonde-system method:

- Cost of determining $P(x)$: VERY EASY. Essentially we can write a formula for $P(x) = y_0 l_0(x) + \cdots + y_n l_n(x)$ without solving any system.
  However if we wanted to write $P(x)$ in the form $a_0 + a_1 x + \cdots + a_n x^n$ the cost for this would be very high! Even writing a single $l_i(x)$ ~~was~~ in this form would require $\approx N^2$ operations (if we are careful how we do it), leading to a $O(N^3)$ cost for the entire $P(x)$.
- Cost of evaluating $P(x)$ for an arbitrary x: SIGNIFICANT
  If we don't want to precompute the a_i's, evaluating each $l_i(x)$ requires n subtractions & n multiplications. In total, we need about n^2 operations to compute $P(x)$. This is not as bad as the n^3 operations to find the a_i's, but still quite high.
- Availability of derivatives: NOT READILY AVAILABLE
  Differentiating each l_i (using product rule) yields n terms, each with $n-1$ factors $\Rightarrow$ expensive.

- Incremental constuction: NOT SUPPORTED

  The constuction of the l_i's assumes we know all the x_i's. However bulding $P(x)$ from scratch if we are given an extra data point is not all that expensive...

Still, Lagrange interpolation is a good quality method, if we can accept its limitations.

Newton interpolation (§4.4) is another alternative, which enables both efficient evaluation and allows for incremental construction. Additionally, it allows the a_i's to be evaluated efficiently, and from those we can easily obtain derivatives, too.

Here is the basic idea:

We want to interpolate $(x_0, y_0), \ldots, (x_n, y_n)$.

Step 0: Define a 0-degree polynomial $P_0(x)$ that just interpolates (x_0, y_0). Obviously, we can achieve that by simply selecting

$$P_0(x) = y_0$$

<u>Step 1</u> Define a 1st degree polynomial $P_1(x)$ that now interpolates both (x_0, y_0) and (x_1, y_1). We also want to take advantage of the previously defined $P_0(x)$, by constructing P_1 as:

$$P_1(x) = P_0(x) + M_1(x)$$

$M_1(x)$ is a 1st degree polynomial and it needs to satisfy:

$$\underbrace{P_1(x_0)}_{=y_0} = \underbrace{P_0(x_0)}_{=y_0} + M_1(x_0) \Rightarrow M_1(x_0) = 0$$

Thus $M_1(x) = q(x - x_0)$. We can determine q using:

$$P_1(x_1) = P_0(x_1) + q(x_1 - x_0) \Rightarrow q = \frac{P_1(x_1) - P_0(x_1)}{x_1 - x_0} = \frac{y_1 - P_0(x_1)}{x_1 - x_0}$$

<u>Step 2</u>: Now constuct $P_2(x)$ which interpolates the three points $(x_0, y_0), (x_1, y_1), (x_2, y_2)$. Define it as:

$$P_2(x) = P_1(x) + M_2(x) \qquad (M_2 : \text{degree} = 2).$$

Once again we observe that

$$\left.\begin{aligned} \underbrace{P_2(x_0)}_{=y_0} &= \underbrace{P_1(x_0)}_{=y_0} + M_2(x_0) \\ \underbrace{P_2(x_1)}_{=y_0} &= \underbrace{P_1(x_1)}_{=y_0} + M_2(x_1) \end{aligned}\right\} \Longrightarrow M_2(x_0) = M_2(x_1) = 0$$

Thus $M_2(x)$ must have the form:

$$M_2(x) = c_2(x-x_0)(x-x_1).$$

Substituting $x \leftarrow x_2$ we get an expression for c_2

$$y_2 = P_2(x_2) = P_1(x_2) + c_2(x_2-x_0)(x_2-x_1)$$

$$\Rightarrow c_2 = \frac{y_2 - P_1(x_2)}{(x_2-x_0)(x_2-x_1)}$$

...

Step k: In the previous step, we constructed a $(k-1)$ degree polynomial that interpolates $(x_0, y_0) \ldots (x_{k-1}, y_{k-1})$. We will use this $P_{k-1}(x)$ and now define an n-degree polynomial $P_k(x)$ such that all of $(x_0, y_0) \ldots (x_k, y_k)$ are now interpolated.

Again $P_k(x) = P_{k-1}(x) + M_k(x)$ where M_k has degree $=k$

2|10|2011 L

Now, we have:

For any $i \in \{0, 1, \dots, k-1\}$

$$\underbrace{P_k(x_i)}_{=y_i} = \underbrace{P_{k-1}(x_i)}_{=y_i} + M_k(x_i) \Rightarrow M_k(x_i) = 0 \quad \forall i = 0, 1, \dots, k-1$$

Thus, the k-degree polynomial M_k must have the form

$$M_k(x) = c_k (x - x_0) \cdots (x - x_{k-1})$$

Substituting $x \leftarrow x_k$ we get

$$y_k = P_k(x_k) = P_{k-1}(x_k) + c_k (x_k - x_0) \cdots (x_k - x_{k-1})$$

$$\Rightarrow \quad c_k = \frac{y_k - P_{k-1}(x_k)}{\prod_{j=0}^{k-1} (x_k - x_j)}$$

Every polynomial $M_i(x)$ in this process is written as:

$$M_i(x) = c_i \cdot n_i(x) \quad \text{where } n_i(x) = \prod_{j=0}^{i-1} (x - x_j)$$

After n steps, the interpolating polynomial $P_n(x)$ is then written as:

$$P(x) = c_0 n_0(x) + c_1 n_1(x) + \cdots + c_n n_n(x).$$

where $n_0(x) = 1$

$n_1(x) = x - x_0$

$n_2(x) = (x-x_0)(x-x_1)$

$\vdots$

$n_K(x) = (x-x_0)(x-x_1)\cdots(x-x_{K-1})$

These are the Newton Polynomials (compare with the Lagrange polynomials $l_i(x)$). Note the x_i's are called the centers

Let us evaluate Newton interpolation, as we did with other methods:

- Cost of evaluating $P(x)$ for an arbitrary x : EASY
  This can be accellerated, using a modification of Horner's scheme

$$P(x) = c_0 + c_1(x-x_0) + c_2(x-x_0)(x-x_1) + \cdots$$
$$+ c_n(x-x_0)(x-x_1)\cdots(x-x_{n-1}) =$$

$$= c_0 + (x-x_0)\Big[c_1 + (x-x_1)\big[c_2 + (x-x_2)\big[c_3 + \cdots$$
$$+ c_{n-1} + (x-x_{n-1})c_n\big]\big]\cdots\Big]$$

e.g. for $n=3$

$$P(x) = c_0 + c_1(x-x_0) + c_2(x-x_0)(x-x_1) + c_3(x-x_0)(x-x_1)(x-x_2)$$

$$= c_0 + (x-x_0)\Big[c_1 + (x-x_1)\big[c_2 + (x-x_2)\,c_3\big]\Big]$$

- Cost of determining $P(x)$ (i.e, the coefficients $\{c_i\}$).

We saw one way of computing them, when describing the overall method. There is, however, another efficient and systematic way to compute them, called <u>divided differences</u>. A divided difference is a function defined over a set of sequentially indexed centers, e.g.

$$x_i, x_{i+1}, \ldots, x_{i+j-1}, x_{i+j}$$

The divided difference of these values is denoted by:

$$f[x_i, x_{i+1}, \ldots, x_{i+j-1}, x_{i+j}]$$

The value of this symbol is defined recursively, as follows:

For divided differences with 1 argument

$$f[x_i] := f(x_i) = y_i$$

With two arguments:

$$f[x_i, x_{i+1}] = \frac{f[x_{i+1}] - f[x_i]}{x_{i+1} - x_i}$$

With three:

$$f[x_i, x_{i+1}, x_{i+2}] = \frac{f[x_{i+1}, x_{i+2}] - f[x_i, x_{i+1}]}{x_{i+2} - x_i}$$

With $j+1$ arguments:

$$f[x_i, x_{i+1}, \ldots, x_{i+j-1}, x_{i+j}] = \frac{f[x_{i+1}, \ldots, x_{i+j}] - f[x_i, \ldots, x_{i+j-1}]}{x_{i+j} - x_i}$$

The fact that makes divided differences so useful, is that $f[x_i, x_{i+1}, \ldots, x_{i+j-1}, x_{i+j}]$ can be shown to be the coefficient of the <u>highest power of x</u> in a polynomial that interpolates through

$$(x_i, y_i), (x_{i+1}, y_{i+1}), \ldots, (x_{i+j-1}, y_{i+j-1}), (x_{i+j}, y_{i+j})$$

Why is this useful ?

Remember, the polynomial that interpolates

$(x_0, y_0), \ldots\ldots, (x_k, y_k)$ is

$$P_k(x) = \underbrace{P_{k-1}(x)}_{\substack{\text{highes power} \\ = x^{k-1}}} + \underbrace{c_k (x-x_0) \cdots (x-x_{k-1})}_{= c_k x^k + \text{lower powers.}}$$

Thus $c_k = f[x_0, x_1, x_2, \ldots, x_k]$!

or

$$\begin{aligned} P(x) = \ & f[x_0] \\ + \ & f[x_0, x_1] (x-x_0) \\ + \ & f[x_0, x_1, x_2] (x-x_0)(x-x_1) \\ & \vdots \\ + \ & f[x_0, x_1, \ldots, x_n] (x-x_0) \cdots (x-x_{n-1}). \end{aligned}$$

So, if we can quickly evaluate the divided differences, we have determined $P(x)$!

Let is see a specific example 2|10|2011 L

$(x_0, y_0) = (-2, -27)$

$(x_1, y_1) = (0, -1)$

$(x_2, y_2) = (1, 0)$

$f[x_0] = y_0 = -27$

$f[x_1] = y_1 = -1$

$f[x_2] = y_2 = 0$

$$f[x_0, x_1] = \frac{f[x_1] - f[x_0]}{x_1 - x_0} = \frac{-1 - (-27)}{0 - (-2)} = 13$$

$$f[x_1, x_2] = \frac{f[x_2] - f[x_1]}{x_2 - x_1} = \frac{0 - (-1)}{1 - 0} = 1$$

$$f[x_0, x_1, x_2] = \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0} = \frac{1 - 13}{1 - (-2)} = -4$$

thus $P(x) = f[x_0] + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2)(x - x_0)(x - x_1)$

$= -27 + 13(x+2) - 4(x+2) \cdot x$