Lagrange interpolation

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

Lagrange interpolation constructs \( P_n \) as:

\[
P_n(x) = y_0 l_0(x) + y_1 l_1(x) + \ldots + y_n l_n(x)
\]

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:

\[
l_i(x) = \frac{(x-x_0) \ldots (x-x_{i-1})(x-x_{i+1}) \ldots (x-x_n)}{(x_i-x_0) \ldots (x_i-x_{i-1})(x_i-x_{i+1}) \ldots (x_i-x_n)} = \frac{1}{\prod_{j \neq i} (x_i-x_j)} \]

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 \( P(x) \) as \( y_0(x) + \ldots + y_n(x) \) without solving any system. However, if we wanted to write \( P(x) \) in the form \( a_0 + a_1 x + \ldots + a_n x^n \), the cost for this would be very high! Even writing a single \( l_i(x) \) in this form would require \( \alpha N^2 \) 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 \( n^3 \) 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.
The construction of the $l_i$'s assumes we know all the $x_i$'s. However, building $P(x)$ 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

$$P_0(x) = y_0$$
Step 1: 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:

\[
\frac{P_1(x_0)}{y_0} = \frac{P_0(x_0) + M_1(x_0)}{y_0} \Rightarrow M_1(x_0) = 0
\]

Thus \( M_1(x) = c_1(x-x_0) \). We can determine \( c_1 \) using:

\[
P_1(x_1) = P_0(x_1) + c_1(x_1-x_0) \Rightarrow c_1 = \frac{P_1(x_1) - P_0(x_1)}{x_1-x_0} = \frac{y_1 - P_0(x_1)}{x_1-x_0}
\]

Step 2: Now construct \( P_2(x) \) which interpolates the three points \((x_0, y_0), (x_1, y_1), (x_2, y_2)\). Define it as:

\[
P_2(x) = P_1(x) + M_2(x) \quad (M_2: \text{degree} = 2)
\]
Once again we observe that
\[
\begin{align*}
\frac{P_2(x_0)}{y_0} &= \frac{P_1(x_0) + M_2(x_0)}{y_0} \\
\frac{P_2(x_1)}{y_0} &= \frac{P_1(x_1) + M_2(x_1)}{y_0}
\end{align*}
\]
\[\Rightarrow M_2(x_0) = M_2(x_1) = 0\]

Thus \( M_2(x) \) must have the form:
\[M_2(x) = c_2 (x-x_0)(x-x_1)\].

Substituting \( x-x_0 \) we get an expression for \( c_2 \)
\[
y_2 = P_2(x_2) = P_1(x_2) + c_2 (x_2-x_0)(x_2-x_1)
\]
\[\Rightarrow c_2 = \frac{y_2 - P_1(x_2)}{(x_2-x_0)(x_2-x_1)}\]

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

Again,
\[P_n(x) = P_{k-1}(x) + M_n(x)\] where \( M_n \) has degree \( k \).
Now, we have:

For any \( r \in \{0, 1, \ldots, k-1\} \),

\[
\frac{P_n(x_i)}{y_i} = \frac{P_{k-1}(x_i)}{y_i} + N_k(x_i) = M_k(x_i) = 0 \quad \forall i = 0, 1, \ldots, k-1
\]

Thus, the \( k \)-degree polynomial \( M_n \) must have the form

\[
M_k(x) = c_k(x - x_0)(x - x_k)
\]

Substituting \( x \rightarrow x_k \) we get

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

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

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

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

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

\[
P(x) = c_0N_0(x) + c_1N_1(x) + \ldots + c_nN_n(x).
\]
Where \( N_0(x) = 1 \)
\( N_1(x) = x - x_0 \)
\( N_2(x) = (x - x_0)(x - x_1) \)
\( \vdots \)
\( N_k(x) = (x - x_0)(x - x_1) \ldots (x - x_{k-1}) \)

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 accelerated, using a modification of Horner's scheme

\[
P(x) = c_0 + c_1 (x - x_0) + c_2 (x - x_0)(x - x_1) + \ldots \\
+ c_m (x - x_0)(x - x_1) \ldots (x - x_{m-1}) = \\
= c_0 + (x - x_0) \left[ c_1 + (x - x_1) \left[ c_2 + (x - x_2) \left[ c_3 + \ldots \right. \right. \right. \\
\left. \left. \left. + c_{m-1} + (x - x_{m-1}) \right] \right] \ldots \right] 
\]
e.g. for \( n=3 \)

\[
P(x) = c_0 + c_1 (x-x_0) + c_2 (x-x_0)(x-x_1) + c_3 (x-x_0)(x-x_1)(x-x_2)
\]

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

- 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 divided differences. 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:

\[
\left[ \frac{\Delta}{\Delta x} \right] f (x_i, x_{i+1}, \ldots, x_{i+j-1}, x_{i+j})
\]

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 highest power of \( x \) in a polynomial that interpolates through

\((x_i, y_i), (x_{i+1}, y_{i+1}), \ldots, (x_{i+j}, y_{i+j}), (x_{i+j}, y_{i+j})\)
Why is this useful?

Remember, the polynomial that interpolates
\((x_0, y_0), \ldots, (x_k, y_k)\) is

\[ P_n(x) = P_{n-1}(x) + c_n (x-x_0) \cdots (x-x_{n-1}) \]

\[ = x^{k-1} = c_n x^k + \text{lower powers}. \]

Thus \(c_n = \frac{f}{[x_0, x_1, x_2, \ldots, x_k]}\) !

or

\[ P(x) = \frac{f}{[x_0]} \]

\[ + \frac{f}{[x_0, x_1]} (x-x_0) \]

\[ + \frac{f}{[x_0, x_1, x_2]} (x-x_0)(x-x_1) \]

\[ \vdots \]

\[ + \frac{f}{[x_0, x_1, \ldots, x_n]} (x-x_0) \cdots (x-x_{n-1}). \]

So, if we can quickly evaluate the divided differences, we have determined \(P(x)\) !
Let's see a specific example

\[(x_0, y_0) = (-2, -27)\]
\[(x_1, y_1) = (0, 1)\]
\[(x_2, y_2) = (1, 0)\]

\[f[x_0] = y_0 = -27\]
\[f[x_1] = y_1 = 1\]
\[f[x_2] = y_2 = 0\]

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

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

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

Thus \[P(x) = f[x_0] + f[x_0, x_1](x-x_0) + f[x_0, x_1, x_2](x-x_0)(x-x_1)\]

\[= -27 + 13(x-2) - 4(x-2)x\]