Interpolation & Extrapolation

(A) Lagrange Interpolation (polynomials)

\[(x_i, f(x_i))\]

0 - data pts

Form equation or

expansive numerical

extrapolations

\[\star\]

Need in-between

values!

\[\rightarrow\] by constructing

\[P_n(x)\] through

data pts.

\[\rightarrow\] unique straight line

\[\rightarrow\] quadratic

\[\rightarrow\] cubic

Weierstrass Approximation Theorem

\[\rightarrow f(x) \text{ in } [a,b] \text{ can always be approximated by } P_n(x)^n\]

(Continuous)

Note: Interpolation ≠ curve fitting

Inter \[\rightarrow\] good estimate of local \( (P(x_i) = f(x_i)) \)

curve fit \[\rightarrow\] global behavior \( (P(x_i) \neq f(x_i)) \)

\(- \sum \) is min over \( r \)
Polynomial Construction

Given: N pairs of points \( (1, \ldots, N) \) : \( [x_i, f(x_i)] \)

Goal: Construct a polynomial \( p_{N-1}(x) \) of order \( (N-1) \)

such that

\[ p_{N-1}(x_i) = f(x_i) \]

A simple example: 2 points - \( (x_1, f(x_1)) \)

\( (x_2, f(x_2)) \)

From Taylor's series expansion, we have

\[
\begin{align*}
  f(x) &= f(x) + (x_i - x)f'(x) + O(x^2) \\
  f(x_2) &= f(x) + (x_2 - x)f'(x) + O(x^2)
\end{align*}
\]

Expanding \( f(x_i) \) and \( f(x_2) \) around \( x \).

Now, we want to define a 1st-order polynomial

\( p_1(x) \) — linear in \( x \) [a straight line]

to approximate \( f(x) \) by requiring that

\[ f(x_i) = p_1(x_i) \]
Motivated by Taylor's expansion, try:

\[ f(x) = P(x) + (x_1 - x) P'(x) \]

And \[ f(x_2) = P(x) + (x_2 - x) P'(x) \]

**Note:** by construction,

\[ f(x_1) = P(x_1) \quad \& \quad f(x_2) = P(x_2) \]

Now, we can solve for \( P(x) \) by multiplying \( (x_2 - x) \) to the top eq. and \( (x_1 - x) \) to the bottom eq.

\[ f(x_1)(x_2 - x) = (x_2 - x) P(x) + (x_2 - x)(x_1 - x) P'(x) \]

\[ f(x_2)(x_1 - x) = (x_1 - x) P(x) + (x_1 - x)(x_2 - x) P'(x) \]

\[ \frac{f(x_1)(x_2 - x) - f(x_2)(x_1 - x)}{(x_2 - x)} = P(x) \left[ (x_2 - x) - (x_1 - x) \right] \]

\[ P(x) = \frac{x - x_2}{x_1 - x_2} f(x_1) + \frac{x - x_1}{x_2 - x_1} f(x_2) \]

- a first order polynomial such that \( P(x_1) = f(x_1) \) and \( P(x_2) = f(x_2) \)
- a linearly weighted sum of \( f(x_1) \) and \( f(x_2) \)
For a general case with \( P_{n-1}(x) \) and \( N \) data points:

Lagrange formula is a general way to construct \( P_{n-1}(x) \) from \( N \) data points \([(x_1, f(x_1)), \ldots, (x_N, f(x_N))]\)

\[
P_{n-1}(x) = \sum_{k=1}^{N} L_{nk}(x) f(x_k)
\]

where

\[
L_{nk}(x) = \frac{(x-x_1) \ldots (x-x_{k-1})(x-x_k) \ldots (x-x_N)}{(x_k-x_1) \ldots (x_k-x_{k-1})(x_k-x_{k+1}) \ldots (x_k-x_N)}
\]

Example: we have already seen the 2 points example:

Here is one for three points: \( \{ (x_1, f_1), (x_2, f_2), (x_3, f_3) \} \)

\[
P_2(x) = \frac{(x-x_2)(x-x_3)}{(x_1-x_2)(x_1-x_3)} f_1 + \frac{(x-x_1)(x-x_3)}{(x_2-x_1)(x_2-x_3)} f_2 + \frac{(x-x_1)(x-x_2)}{(x_3-x_1)(x_3-x_2)} f_3
\]
Property of $L_{nk}(x)$:

- $L_{nk}(x)$ are polynomials of $(N-1)$ degree

\[
\begin{align*}
\sum_{k=1}^{N} L_{nk}(x_k) &= 1 \\
L_{nk}(x_i) &= 0 \text{ if } i \neq k.
\end{align*}
\]

It acts like a Kronecker delta function!

Property of $P_{n-1}(x)$:

1. By construction, we have:

\[
P_{n-1}(x_i) = f_i \quad (\text{as required!})
\]

2. $L_{nk}(x)$ are polynomials of $N$-degree,

so that $P_{n-1}(x)$ is a polynomial of degree $(N-1)$!

Main Theorem of Polynomial Interpolation:

Let \{(x_1, f_1), \ldots, (x_n, f_n)\} be $n+1$ points in the plane with distinct $x_i$. Then, there exists one and only one polynomial $P_{n-1}(x)$ of degree $(N-1)$ or less that satisfies $P_{n-1}(x_i) = f_i$ for all $i = 1, \ldots, n$. 

Proof (on the uniqueness part):

Existence is given by explicit construction.

Let $P_n(x)$ and $Q_{N-1}(x)$ be two different polynomials of degree $(N-1)$ or less that interpolate the given $N$ data points.

This means that

$$P_n(x_i) = Q_{N-1}(x_i) = f_i \text{ for all } i = 1, \ldots, N.$$  

Now, consider the different function

$$H_{N-1}(x) = P_n(x) - Q_{N-1}(x)$$

$H_{N-1}(x)$ is at most of degree $(N-1)$ AND

$$H_{N-1}(x_i) = P_n(x_i) - Q_{N-1}(x_i) = 0 \text{ (at the data points)!}$$

So that, $H_{N-1}(x)$ has $N$ distinct zeros!

Then, from the Fundamental Theorem of Algebra (a degree $p$ polynomial can have at most $p$ zeros unless it is an identically zero polynomial.)

$\Rightarrow H_{N-1}(x) = 0!$ and $P_n(x)$ is unique!
Neville's algorithm

- Recursive way
- Iterative way
- Computationally efficient way to construct one continuous \( P_n(x) \) thru \((n+1)\) pts

**4 pts example**

\[
\left[ f_1, f_2, f_3, f_4 \right] \quad \left[ (x_1, x_2, x_3, x_4) \right]
\]

(Necessary eveny spaced)

1. Define 4 0th order polynomials (constant fins)

   \( P_1, P_2, P_3, P_4 \) to approx f new \( x_1, x_2, x_3, x_4 \)

   such that \( P_i = f_i \)

2. For each pairs of pts \( [x_i, x_{i+1}] \) (3),

   Define 2 1st order polynomials (straight lines)

   \( P_{12}, P_{23}, P_{34} \) to approx f using \( P_i \):

   **Note**: For each pairs of pts \( [x_i, x_{i+1}] \), \( \{P_{12}, P_{23}, P_{34}\} \)

   are unique

\[\text{constructed from} \ P_0\]
Similarly for high order polynomials with more pts.

\[ \begin{align*}
- & 2 \left\{ \begin{array}{c}
X_1, X_2, X_3 \rightarrow P_{123} \quad \text{ Quadratic} \\
X_2, X_3, X_4 \rightarrow P_{234} \quad \text{ Quadratic}
\end{array} \right.
\end{align*} \]

\[ \uparrow \quad \text{constructed from } P_{ij} \]

Finally, the best we can do with 4 data pts is a cubic!

\[ 1 - X_1, X_2, X_3, X_4 \rightarrow P_{1234} \quad \text{ Cubic} \]

\[ \uparrow \quad \text{constructed from } P_{ijk} \]

\[ \quad \frac{\text{(this is the final goal)}}{\text{We can put them in a table}} \]

| \hline
<table>
<thead>
<tr>
<th>0th order poly</th>
<th>1st order poly</th>
<th>2nd order poly</th>
<th>3rd order poly</th>
</tr>
</thead>
<tbody>
<tr>
<td>( X_1 ): ( P_1 )</td>
<td>( P_{12} )</td>
<td>( P_{123} )</td>
<td>( P_{1234} )</td>
</tr>
<tr>
<td>( X_2 ): ( P_2 )</td>
<td>( P_{12} )</td>
<td>( P_{123} )</td>
<td>( P_{1234} )</td>
</tr>
<tr>
<td>( X_3 ): ( P_3 )</td>
<td>( P_{23} )</td>
<td>( P_{234} )</td>
<td>( P_{1234} )</td>
</tr>
<tr>
<td>( X_4 ): ( P_4 )</td>
<td>( P_{34} )</td>
<td>( P_{34} )</td>
<td>( P_{1234} )</td>
</tr>
</tbody>
</table>

Neville's algorithm is an efficient method to fill them in from left to right!
- \( P_i : \) Construction of other orders is simple:
  \[ P_i = f_i \quad \{P_1, P_2, P_3, P_4\} \]

- \( P_{i, i+1} : \) Linear interpolation between \( P_i, P_{i+1} \)
  \[ P_{i, i+1}(x) = \frac{(x - x_{i+1}) P_i - (x - x_i) P_{i+1}}{(x_i - x_{i+1})} \]

- \( P_{i, i+1, i+2} : \) Linear interpolation between \( P_{i, i+1}, P_{i+1, i+2} \)
  \[ P_{i, i+1, i+2}(x) = \frac{(x - x_{i+2}) P_{i, i+1} - (x - x_{i+1}) P_{i+1, i+2}}{(x_{i+1} - x_{i+2})} \]

- \( P_{i, i+1, i+2, i+3} : \) Linear interpolation between \( P_{i, i+1, i+2}, P_{i+1, i+2, i+3} \)
  \[ P_{i, i+1, i+2, i+3}(x) = \frac{(x - x_{i+3}) P_{i, i+1, i+2} - (x - x_{i+2}) P_{i+1, i+2, i+3}}{(x_{i+2} - x_{i+3})} \]

- \( P_{i, i+1, i+2, i+3, i+4} : \) Linear interpolation between \( P_{i, i+1, i+2, i+3}, P_{i+1, i+2, i+3, i+4} \)
  \[ P_{i, i+1, i+2, i+3, i+4}(x) = \frac{(x - x_{i+4}) P_{i, i+1, i+2, i+3} - (x - x_{i+3}) P_{i+1, i+2, i+3, i+4}}{(x_{i+3} - x_{i+4})} \]
To show

\[ p_{123}(x) = \frac{(x-x_3)p_2 - (x-x_1)p_2}{x_1-x_3} \]

does give \[ p_{123}(x) \rightarrow \text{Quadratic through points} \]

\text{Lagrange formula}

\[
\frac{(x-x_5)}{(x_1-x_3)} \left[ \frac{(x-x_2)f_1}{(x_1-x_2)} - \frac{(x-x_1)f_2}{(x_1-x_2)} \right] \\
- \frac{(x-x_1)}{(x_1-x_3)} \left[ \frac{(x-x_2)f_2}{(x_2-x_3)} - \frac{(x-x_2)f_3}{(x_2-x_3)} \right] \\
= \frac{(x-x_3)(x-x_2)}{(x_1-x_3)(x_1-x_2)} f_1 \\
- \frac{(x-x_5)(x-x_1)(x_2-x_3)}{(x_1-x_3)(x_1-x_2)(x_2-x_3)} f_2 - \frac{(x-x_1)(x-x_3)(x_1-x_2)}{(x_1-x_3)(x_2-x_3)(x_1-x_2)} f_2 \\
+ \frac{(x-x_1)(x-x_2)}{(x_1-x_3)(x_2-x_3)} f_3 \\
- \left( \frac{(x-x_3)(x-x_1)f_2}{(x_4-x_2)(x_1-x_2)(x_2-x_3)} \left[ x_2 - x_5 + x_1 - x_2 \right] \right) \\
= \frac{(x-x_1)(x-x_3)}{(x_2-x_1)(x_2-x_3)} f_2 \checkmark
Advantage:

This can be put into an iterative form:

\[\begin{align*}
\{ \text{two parents} & : \frac{P_i(i+1)\ldots(i+m)}{m} \quad (P_{23}) \\
\{ \text{one daughter} & : \frac{P_i(i+1)\ldots(i+m)}{(m+1)} \quad (P_{123}) \\
\end{align*}\]

\[P_i(i+1)\ldots(i+m) = \frac{(X-x_i) - (X-x_{i+m})}{m} \]
Usage of this algorithm

\* input: \( x \)  \hspace{1cm} output: \( P_{1234}(x) \)

\[ \begin{align*}
X_1: & \hspace{1cm} Y_1 = P_1 \\
X \rightarrow X_2: & \hspace{1cm} Y_2 = P_2 \\
X_2: & \hspace{1cm} Y_2 = P_2 \\
X_3: & \hspace{1cm} Y_3 = P_3 \\
X_4: & \hspace{1cm} Y_4 = P_4
\end{align*} \]

\[ \begin{align*}
P_{12} & \rightarrow P_{123} \\
P_{23} & \rightarrow P_{234} \\
P_{34} & \rightarrow \text{fill in tree}
\end{align*} \]

0th order  \( \gamma \cong P_2 = Y_2 \)  \( P_1, P_3, P_4 \)

1st order  \( \gamma \cong P_{23} = \frac{(x_1 - x_3)P_2 - (x - x_2)P_3}{(x_2 - x_3)} \)  \( P_{12}, P_{34} \)

2nd order  \( \gamma \cong P_{123} = \frac{(x - x_3)P_{12} - (x - x_1)P_{23}}{(x_1 - x_3)} \)  \( P_{234} \)

3rd order  \( \gamma \cong P_{1234} = \frac{(x - x_4)P_{123} - (x - x_1)P_{234}}{(x_1 - x_4)} \)  \( \_ \)

\* it is a way to calculate (estimate) the value \( f(x) \approx P_{1234}(x) \).

\* it is not

\* it does not give the exact of the estimating function!

\* need to go through the tree procedure for each \( x \).