Computational Methods
CMSC/AMSC/MAPL 460

Polynomial Interpolation

Ramani Duraiswami,
Dept. of Computer Science
Interpolating polynomials in power form

• Given $n$ values of the function $y_i$ at points $x_i$
• Fit a polynomial $P(x)$ that interpolates data at these points
• If we have $n$ points to interpolate, then a polynomial of degree $n-1$ is
  \[
P(x) = c_1 x^{n-1} + c_2 x^{n-2} + \cdots + c_{n-1} x + c_n
  \]
• We can write the condition that it interpolate as a linear system
  \[
  \begin{pmatrix}
    x_1^{n-1} & x_1^{n-2} & \cdots & x_1 & 1 \\
    x_2^{n-1} & x_2^{n-2} & \cdots & x_2 & 1 \\
    \vdots & \vdots & \ddots & \vdots & \vdots \\
    x_n^{n-1} & x_n^{n-2} & \cdots & x_n & 1
  \end{pmatrix}
  \begin{pmatrix}
    c_1 \\
    c_2 \\
    \vdots \\
    c_n
  \end{pmatrix}
  =
  \begin{pmatrix}
    y_1 \\
    y_2 \\
    \vdots \\
    y_n
  \end{pmatrix}
  \]
Vandermonde matrices

**Example:** A **Vandermonde** matrix $A$ is defined by a vector of elements $x_1, \ldots, x_n$. Its first column is all ones. Each later column is the preceding one times this vector.

- **Matlab code**
  \[
  \begin{align*}
  n &= \text{length}(x); \\
  V(:,1) &= \text{ones}(n,1); \\
  \text{for } j &= 2:n, \\
  V(:,j) &= x.*V(:,j-1); \\
  \text{end}
  \end{align*}
  \]

- **In matlab Vandermonde matrix** is defined in flipped order using the function `vander`

- **Example** ...

- Also there is a function to fit polynomials to data, `polyfit`

- Vandermonde matrices are nonsingular if the points are distinct

- However they can be very poorly conditioned
Computing interpolants by hand

- Suppose we have data at \( n \) points and we have fit a polynomial.
- How can we do this fit efficiently?
- Fit for one point is \( y = y_1 \).
- Fit for two points can be written as \( y = a(x-x_1) + b(x-x_2) \).
- Fit for three points can be written as
  \[ y = a(x-x_1)(x-x_2) + b(x-x_1)(x-x_3) + c(x-x_2)(x-x_3) \]
- And so on …
- Advantage: each coefficient can be calculated independent of others.
  - Why?
  - What is the form of the coefficient computed?
Lagrange and Newton forms for interpolations

- Lagrange and Newton modified these forms further to conveniently compute polynomials
Lagrange Polynomials

• Summation of terms, such that:
  – Equal to \( f(x) \) at a data point.
  – Equal to zero at all other data points.
  – Each term is a \( n^{th} \)-degree polynomial

\[
p_n(x) = \sum_{i=0}^{n} L_i(x) f(x_i)
\]

\[
L_i(x) = \prod_{k=0, k \neq i}^{n} \frac{(x - x_k)}{(x_i - x_k)}
\]

\[
L_i(x_j) = \delta_{ij} = \begin{cases} 
1 & i = j \\
0 & i \neq j 
\end{cases}
\]
Lagrange form

- $x_0 = 1, f(x_0) = -5$; $x_1 = 2, f(x_1) = -3$; $x_2 = 3, f(x_2) = 2$
- Define
- $L_0 = \frac{(x-2)(x-3)}{(1-2)(1-3)} = \frac{(x-2)(x-3)}{2}$
- $L_1 = \frac{(x-1)(x-3)}{(2-1)(2-3)} = -(x-1)(x-3)$
- $L_2 = \frac{(x-1)(x-2)}{(3-1)(3-2)} = \frac{(x-1)(x-2)}{2}$
- Polynomial is $-5L_0 - 3L_1 + 2L_2$
- $-\frac{5}{2} (x-2)(x-3) + 3 (x-1)(x-3) + (x-1)(x-2)$
Newton Interpolation

- Consider our data set of $n+1$ points $y_i = f(x_i)$ at $x_0, x_1, \ldots, x_i, \ldots, x_n$: $x_n > x_0$
- Since $p_n(x)$ is the unique polynomial $p_n(x)$ of order $n$, write it:

\[
p_n(x) = b_0 + b_1(x - x_0) + b_2(x - x_0)(x - x_1) + \ldots \]

\[
b_0 = f(x_0)
\]

\[
b_1 = f[x_1, x_0] = \frac{f(x_1) - f(x_0)}{x_1 - x_0}
\]

\[
b_2 = f[x_2, x_1, x_0] = \frac{f[x_2, x_1] - f[x_1, x_0]}{x_2 - x_0}
\]

\[\vdots\]

\[
b_n = f[x_n, x_{n-1}, \ldots, x_0] = \frac{f[x_n, \ldots] - f[x_{n-1}, \ldots]}{x_n - x_0}
\]

- $f[x_i, x_j]$ is a first divided difference
- $f[x_2, x_1, x_0]$ is a second divided difference, etc.
- Efficient way of adding points to the interpolation!
- Used to fit data to a table
Newton Interpolation

- Example
  \[x_0 = 1, f(x_0) = -5; \quad x_1 = 2, f(x_1) = -3;\]
  \[x_2 = 3, f(x_2) = 2; \quad x_3 = 4, f(x_3) = 4.\]

- Build divided difference table
  - \[f[x_0] = -5\]
  - \[f[x_1] = -3, f[x_0, x_1] = 2\]
  - \[f[x_2] = 2, f[x_1, x_2] = 5, f[x_0, x_1, x_2] = 3/2\]
  - \[f[x_3] = 4, f[x_2, x_3] = 2, f[x_1, x_2, x_3] = -3/2\]
  - \[f[x_0, x_1, x_2, x_3] = (3/2 + 3/2) / (1 - 4) = -1\]

- To compute Newton form we need \(f[x_0], f[x_0, x_1], f[x_0, x_1, x_2], f[x_0, x_1, x_2, x_3]\)
Newton form

- Interpolation

\[ P(x) = f[x_0] + f[x_0, x_1] (x-x_0) + f[x_0, x_1, x_2] (x-x_0)(x-x_1) + f[x_0, x_1, x_2, x_3](x-x_0)(x-x_1)(x-x_2) \]

\[ P(x) = -5 + 2(x-1) + \frac{3}{2}(x-1)(x-2) - (x-1)(x-2)(x-3) \]