Computational Methods
CMSC/AMSC/MAPL 460

Polynomial Interpolation

Ramani Duraiswami,
Dept. of Computer Science
Interpolation

• Given a function at \( N \) points, find its value at other point(s)
  – “within” the points
    • Interpolation
  – “outside” the points
    • Extrapolation

• How do we extend the value from known points to unknown points?
  – Have to have prior knowledge about the function
  – Can do what is convenient

• Occam’s razor
  – Parsimony: “One should not increase, beyond what is necessary, the number of entities required to explain anything”
  – *Key in all scientific modeling*
  – Will return to this issue
Function representation: Polynomials

• In interpolation sampled values of a function are available
• We need to extend these values to points where they are not available
• Can assume that the function is expanded over a basis of functions which span the functional space
  – Polynomials are a basis
  – Fourier series are another
  – Many others
• Most common basis are power series/polynomials
  – Here $x_*$ is a point about which we expand in series
  – $a_m$ are coefficients
• Polynomial interpolation often preferred
  – Part of theory of Taylor series, solution of differential equations via power series, in computing integrals
  – A theorem guaranteeing that a polynomial can represent a function is available
Properties of Power Series

1) For any power series there exists \( r_* \), such that the series converges absolutely at \(|y - x_*| < r_*\), and diverges at \(|y-x_*| > r_*\).
   - The number \( r_* \) is called the *convergence radius* of the series,
   - \( 0 \leq r_* \leq \infty \).
   - For any number \( q \), such that \( 0 < q < r_* \), the power series uniformly converges at \(|y - x_*| < q\).

2) Convergent power series can be summed, multiplied by a scalar, or multiplied according to the Cauchy rule.

\[
\sum_{m=0}^{\infty} a_m (y - x_*)^m + \sum_{m=0}^{\infty} b_m (y - x_*)^m = \sum_{m=0}^{\infty} (a_m + b_m)(y - x_*)^m, \\
\alpha \sum_{m=0}^{\infty} a_m (y - x_*)^m = \sum_{m=0}^{\infty} \alpha a_m (y - x_*)^m, \\
\left[ \sum_{m=0}^{\infty} a_m (y - x_*)^m \right] \left[ \sum_{m=0}^{\infty} b_m (y - x_*)^m \right] = \sum_{n=0}^{\infty} \left[ \sum_{m=0}^{n} a_m b_{n-m} \right](y - x_*)^n.
\]
Taylor Series

Let \( f(y) \) be a real function, \( f(y) \in C^n [x_*, x_* + r] \)
- \( n^{th} \) derivative \( f^n \) exists for \( x_* \leq y < x_* + r \)

\[
f(y) = f(x_*) + f'(x_*)(y - x_*) + \frac{1}{2!}f''(x_*)(y - x_*)^2 + \ldots + \frac{1}{(n-1)!}f^{(n-1)}(x_*)(y - x_*)^{n-1} + \text{Residual}_n(y).
\]

- Residual determines accuracy
- Two evaluations of remainder
  - Cauchy evaluation
  - Lagrange evaluation

\[
|\text{Residual}_n(y)| \leq \frac{|y - x_*|^2}{n!} \sup_{x_* \leq y < x_* + r} |f^{(n)}(y)|.
\]

\[
\text{Residual}_n(y) = \int_{x_*}^{y} dx \int_{x_*}^{x} dx \ldots \int_{x_*}^{x} f^{(n)}(x)dx = \frac{1}{n!}f^{(n)}(X)(y - x_*)^n,
\]

\( X \in (x_*, x_* + r_*). \)
Interpolation

- Weierstrass theorem

**Theorem**: (Weierstrass) For all $f \in \mathcal{C}[a, b]$, for all $\epsilon > 0$, there exists a degree $n$ and a polynomial $p_n$ such that $\|f - p\|_\infty < \epsilon$.

- Provides a guarantee that polynomials can interpolate any function
- On the other hand does not tell us how to choose the polynomial
- Also does not guarantee that the polynomial will actually “interpolate” the function … only that it will be within $\epsilon$.
- Does not tell what the degree of the polynomial is
Polynomial Facts

• A polynomial of degree $k$ has at most $k$ distinct zeroes, unless it is identically zero.

• Sum of two polynomials of degree $k$ is another polynomial of degree at most $k$.

• Polynomials can be expressed in many ways.

For example $(x - 2)(x - 5) = x^2 - 2x - 5(x - 2) = x^2 - 7x + 10$.

We have used three different sets of basis functions in this example:

1. $(x - 2)(x - 5)$, $x - 1$, and 1.
2. $x^2$, $x$, and $x - 2$.
3. $x^2$, $x$, and 1.
   - Degree of basis functions is 2, 1 and 0 ...
   - Basis 3 is the power basis or monomial basis
   - Any basis can be used ... often “orthogonal polynomials” are used
Uniqueness of interpolant

- We know that the polynomial exists
- Suppose that there are two different polynomials that can interpolate the data
- Let them be $p_{n-1}$ and $q_{n-1}$.
- So we have $p_{n-1}(x_i) = y_i, \ i=1, \ ...n$
  \[ q_{n-1}(x_i) = y_i, \ i=1, \ ...n \]
- So $p_{n-1}(x_i) - q_{n-1}(x_i) = 0, \ i=1, \ ...n$
- $p_{n-1} - q_{n-1}$ is the difference of two polynomials of degree $n-1$.
- It has $n$ zeroes.
- Recall polynomial of degree $k$ has at most $k$ zeroes, or is the zero polynomial.
- Here we have more zeroes than degree … so it is the zero polynomial.
- So interpolant is unique
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 \( n-1 \) can interpolate
  \[
P(x) = c_1 x^{n-1} + c_2 x^{n-2} + \cdots + c_{n-1} x + c_n
  \]
- 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**
  ```matlab
  n = length(x);
  V(:,1) = ones(n,1);
  for j=2:n,
      V(:,j) = x.*V(:,j-1);
  end
  ```

- 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

\[
p_n(x) = \sum_{i=0}^{n} L_i(x) f(x_i)
\]
\[
L_i(x) = \prod_{k=0, k \neq i}^{n} \frac{(x - x_k)}{(x_i - x_k)}
\]
\[
L_i(x_j) = \delta_{ij} = \begin{cases} 
1 & i = j \\
0 & i \neq j 
\end{cases}
\]
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:

\[
p_n(x) = b_0 + b_1(x-x_0) + b_2(x-x_0)(x-x_1) + \ldots + b_n(x-x_0)(x-x_1) \cdots (x-x_{n-1})
\]

\[
b_0 = f(x_0)
\]

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

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

\[\vdots\]

\[
b_n = f[x_n, x_{n-1}, \ldots, x_0] = \frac{f[x_n, \ldots, x_1] - f[x_{n-1}, \ldots, x_0]}{x_n - x_0}
\]

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

- 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

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

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

- Define the error term as:

\[ \varepsilon_n(x) = f(x) - p_n(x) \]

- If \( f(x) \) is an \( n^{\text{th}} \) order polynomial \( p_n(x) \) is of course exact.
- Otherwise, since there is a perfect match at \( x_0, x_1, \ldots, x_n \)
- This function has at least \( n+1 \) roots at the interpolation points.

\[ \therefore \varepsilon_n(x) = (x - x_0)(x - x_1)\cdots(x - x_n)h(x) \]
Interpolation Errors

\[ \epsilon_n(x) = f(x) - p_n(x) = \frac{1}{(n+1)!} f^{(n+1)}(\xi) \prod_{i=0}^{n} (x - x_i) \]

\[ x \in [a, b], \xi \in (a, b) \]

- Looks a bit like Taylor series remainder
- Recall, first \( n+1 \) terms of the Taylor Series is also an \( n^{th} \) degree polynomial.
Interpolation Errors

• Suppose we want to measure error at a point $x$
• To make polynomial go through $x$, add to existing polynomial divided difference term.
• This is the error we make using existing polynomial

\[ x \not\in \{x_0, x_1, \ldots x_n\} \]

\[ \varepsilon_n(x) = f(x) - p_n(x) = f[x_0, x_1, \ldots x_n, x] \prod_{i=0}^{n} (x - x_i) \]

• Comparing with Taylor series

\[ f[x_0, x_1, \ldots x_n] = \frac{1}{n!} f^{(n)}(\xi) \]
Efficient polynomial evaluation

• Given a polynomial in power form how many operations does it take to evaluate it?