Numerical Analysis I
Math 373

Surya Teja Gavva
Rutgers University

July 16 2018

Polynomial Interpolation

Problem 1: Can you approximate a complicated function with simpler functions like polynomials?
Problem 2: Can you find polynomials that interpolate some data?

Polynomial Approximation

Weierstrass Approximation:
Given any continuous function $f : [a, b] \to \mathbb{R}$ and $\varepsilon > 0$ there exists a polynomial $p(x)$ such that $|f(x) - p(x)| < \varepsilon$ for all $x \in [a, b]$.

Polynomial Approximation

Weierstrass Approximation:
Given any continuous function $f : [a, b] \to \mathbb{R}$ and $\varepsilon > 0$ there exists a polynomial $p(x)$ such that $|f(x) - p(x)| < \varepsilon$ for all $x \in [a, b]$.

Can you find some explicit approximations?

Taylor approximations

$$f(x) = f(x_0)+f'(x_0)(x-x_0)+\frac{f''(x_0)}{2!}(x-x_0)^2+\cdots+\frac{f^{(n)}(x_0)}{n!}(x-x_0)^n$$
$$+E(x,x_0,f)$$

Where

$$E(x,x_0,f) = \frac{f^{(n)}(\xi_x)}{n!}(x-x_0)^{n+1}, \xi_x \in (x_0,x)$$

This approximation is only good for x very close to x_0, We need to use more information about $f(x)$ than just the local information at x_0

Interpolation

Given a function $f(x)$, find a polynomial of least degree such that $p(x_i) = f(x_i)$ for $i = 0$ to n.

Lagrange Interpolation

Idea: Find polynomials $L_k(x)$ such that $L_k(x_k) = 1$ and $L_k(x_j) = 0$ for $j \neq k$ Then $\sum_{k=0}^{n} f(x_k)L_k(x)$ satisfies $p(x_i) = f(x_i)$–
When you plug in x_i only $L_i(x_i)$ term is non-zero and that term gives $f(x_i).1 = f(x_i)$

Lagrange Interpolation

$$L_k(x) = \prod_{j \neq k} \frac{(x - x_j)}{(x_k - x_j)}$$

$$= \frac{(x - x_0)(x - x_1)\cdots(x - x_{k-1})(x - x_{k+1})\cdots(x - x_n)}{(x_k - x_0)(x_k - x_1)\cdots(x_k - x_{k-1})(x_k - x_{k+1})\cdots(x_k - x_n)}$$

Note that $L_k(x_k) = 1, L_k(x_j) = 0$ if $j \neq k$

Lagrange Interpolation: Error

Error: We want to understand the error in the approximation ie., $|f(x) - P_n(x)|$ as n increases.
$P_n(x)$ depends on the choice of the interpolating points $\{x_i\}$, so the error depends on this choice and the function f
If f is differentiable $n+1$ times, we have the following estimate for the error.

$$|f(x) - P_n(x)| \leq |\frac{f^{(n+1)}(\eta_x)}{(n+1)!}(x - x_0)(x - x_1)\cdots(x - x_n)|$$

Lagrange Interpolation: Error

Note that the error depends on the the "regularity" of the function and the the interpolation points $\{x_i\}$
Question: Does this error go to zero as $n \to \infty$?
Answer: NO. See Runge Phenomenon

Runge Phenomenon

Consider the function $f(x) = \frac{1}{1+25x^2}$ on $[-1, 1]$
Let $x_i = -1 + \frac{2i}{n}$ be the equidistant points.
The Lagrange interpolation $P_n(x)$ oscillates a lot at the ends of the interval and takes very large values.
Hence the error in approximation diverges as n tends to infinity.

Runge Phenomenon

The interpolations didnt converge to the function in the above example. Can we change the point $\{x_i\}$ and get the convergence? Yes. Special choice of points can make the expression $|(x - x_0)(x - x_1) \cdots (x - x_n)|$ in the error not so big and can give convergence.

Chebyshev Points

In the previous example , equidistant points didn't work and the function diverged a lot at the endpoints. To rectify this we may need points with more density at the end points.
Chebyshev points are a really good choice. $x_k = \cos(\frac{k\pi}{n})$
They have the property that

$$\sup_{x\in[-1,1]} |(x-x_0)(x-x_1)\cdots(x-x_n)| = \frac{1}{2^n} \leq \sup_{x\in[-1,1]} |p(x)|$$

for any monic polynomial $p(x)$ of degree $n+1$
For instance, for equidistant points x_i,
$|(x-x_0)(x-x_1)\cdots(x-x_n)| \sim \frac{1}{1.355^n} > \frac{1}{2^n}$

Chebyshev Points

In fact, for the above Runge function $f(x) = \frac{1}{1+25x^2}$ on $[-1, 1]$, the Lagrange interpolations $P_n(x)$ at the Chebyshev nodes converge uniformly to the function $f(x)$

Chebyshev Points

Are Chebyshev nodes good for all continuous functions? Do the interpolations at Chebyshev points converge to the function?
Yes, if the function is differentiable.
But for any choice of interpolation points, there are continuous functions (may not be differentiable) for which $P_n(x)$ don't converge.

Lagrange Interpolation

How do we compute these polynomials $P_n(x)$
Can we compute $P_{n+1}(x)$ from $P_n(x)$
In the form that we have, computation of $P_{n+1}(x)$ requires computing from scratch again even if we have computed $P_n(x)$
What should we do?

Neville's method

We can use the following relations between interpolations at various subsets of points to compute values of larger degree interpolations from smaller degree ones.
Let $P_{k_1,k_2,\cdots,k_r}$ be the Lagrange interpolation of $f(x)$ at $x_{k_1}, x_{k_2}, \cdots x_{k_r}$

Neville's method

P_k is the degree zero/constant function $f(x_k)$ $P_{i,j}$ is the linear interpolation at x_i, x_j

$$P_{i,j}(x) = \frac{(x - x_i)f(x_j) - (x - x_j)f(x_i)}{x_j - x_i}$$

We have the following formula:

$$P_{0,1,2,..,n}(x) = \frac{(x - x_0)P_{1,2,..,n} - (x - x_n)P_{0,1,2,..,n-1}(x)}{x_n - x_0}$$

Neville's method

We compute values of higher degree interpolation by following iterative process:

$$
\begin{array}{lllll}
x_0 & P_0 & & & \\
 & & P_{0,1} & & \\
x_1 & P_1 & & P_{0,1,2} & \\
 & & P_{1,2} & & P_{0,1,2,3} \\
x_2 & P_2 & & P_{1,2,3} & \\
 & & P_{2,3} & & \\
x_3 & P_3 & & &
\end{array}
$$

Neville's method

Neville's method is good if you want to calculate individual values but to compute the whole polynomial P_n iteratively involves a lot of computation in this form. So we need some method to do iterative computation of P_n from lower degree interpolations.

Newton Divided Difference method

We have

$$P_n(x) = P_{n-1}(x) + a_n(x - x_0)(x - x_1)\cdots(x - x_{n-1})$$

Therefore

$$P_n(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \cdots$$

$$+a_n(x - x_0)(x - x_1)\cdots(x - x_{n-1})$$

How do we find these a_n?

Newton Divided Difference method

Put $x = x_0$, gives $a_0 = f(x_0)$
$x = x_1$ gives $f(x_1) = f(x_0) + a_1(x_1 - x_0)$, so $a_1 = \frac{f(x_1)-f(x_0)}{x_1-x_0}$
Let us denote $f(x_0)$ by $f[x_0]$ and $\frac{f(x_1)-f(x_0)}{x_1-x_0}$ by $f[x_0, x_1]$
$x = x_2$ gives $a_2 = \frac{f[x_1,x_2]-f[x_0,x_1]}{x_2-x_0}$
Denote this by $f[x_0, x_1, x_2]$

In general

$$a_n = f[x_0, x_1, .., x_n] = \frac{f[x_1, x_2, x_3, .., x_n] - f[x_0, x_1, x_2, .., x_{n-1}]}{x_n - x_0}$$

Newton Divided Difference method

So we have

$$P_n(x) = f[x_0] + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) + \cdots$$

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

where

$$f[x_0, x_1, .., x_n] = \frac{f[x_1, x_2, x_3, .., x_n] - f[x_0, x_1, x_2, .., x_{n-1}]}{x_n - x_0}$$

are defined recursively starting with $f[x_i] = f(x_i)$

Divided Difference table

We compute values of divided differences by following iterative process:

$$
\begin{array}{ccccc}
x_0 & f[x_0] & & & \\
 & & f[x_0, x_1] & & \\
x_1 & f[x_1] & & f[x_0, x_1, x_2] & \\
 & & f[x_1, x_2] & & f[x_0, x_1, x_2, x_3] \\
x_2 & f[x_2] & & f[x_1, x_2, x_3] & \\
 & & f[x_2, x_3] & & \\
x_3 & f[x_3] & & &
\end{array}
$$