LEC - 4 | 5

POLYNOMIAL INTERPOLATION - I

Polynomial Interpolation ?

Problem - Given a dataset $(x_0, f_0) \ldots (x_n, f_n)$ consisting of

distinct nodes : $[x_0, x_1, \ldots x_n]$
and values : $[f_0, f_1, \ldots f_n]$

construct a polynomial $p(x)$ of lowest degree such that $p(x_j) = f_j$ for $j = 0 : n$.

Vandermonde Interpolating Polynomial.

Thm - consider nodes $[x_0, \ldots x_n]$
values $[f_0, \ldots f_n]$
There exist a polynomial for degree atmost n such that $p(x_j) = f_j$
for $j = 0 : n$

Proof - Consider the polynomial $p(x) = a_0 x_j^0 + a_1 x_j^1 + \ldots + \ldots a_n x^n$

$P(x_j) = f_j$ for $j = 0 : n$

$$\begin{bmatrix} 1 & x_0 & \cdots & x_0^n \\ 1 & x_1 & & x_1^n \\ 1 & & & \vdots \\ 1 & x_n & & x_n^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} f_0 \\ f_1 \\ \vdots \\ f_n \end{bmatrix}$$

Vandermonde Matrix

$VA = Y$

~~det~~

$\boxed{\det(V) \neq 0 \quad \text{so unique solution.}}$

- The computation is numerically unstable.
- Solution of system require $O(n^3)$ operations.
- Any additional new data (x_{n+1}, f_{n+1}) requires recomputation
- Evaluation of $p(x)$ at given require $O(n^2)$ operations.

Interpolating polynomial in General basis

Let P_n denote the vector space of polynomials of degree at most n. Let $\phi_0(x), \phi_1(x), \ldots \phi_n(x)$ be a basis of P_n. Let $p(x) = a_0 \phi_0(x) + a_1 \phi(x) + \ldots + \ldots \phi_n(x) . a_n$

$$\begin{bmatrix} \phi_0(x_0) & \phi_1(x_0) & \cdots & \phi_n(x_0) \\ \phi_0(x_1) & \phi_1(x_1) & & \vdots \\ \phi_0(x_2) & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ \phi_0(x_n) & \phi_1(x_n) & & \phi_n(x_n) \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} f_0 \\ f_1 \\ \vdots \\ \vdots \\ f_n \end{bmatrix}$$

Coefficient matrix is non singular.
The linear system has unique solution.

LAGRANGE INTERPOLATING POLYNOMIAL

Example - Let say for three data points -

$$(x_0, f_0), \quad (x_1, f_1), \ldots (x_2, f_2)$$

define,

$$p(x) = \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} f_0 + \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} f_1$$

$$+ \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} f_2$$

Then
$$p(x_0) = f_0$$
$$p(x_1) = f_1$$
$$p(x_2) = f_2$$

NOW IN GENERAL -

for data set $(x_0, f_0), (x_1, f_1) \ldots (x_n, f_n)$

define $w(x) = (x-x_0)(x-x_1) \ldots (x-x_n) \in P_{n+1}$

$$y(x) = \prod_{i \neq j} \left(\frac{x - x_i}{-x_i + x_j} \right) = \prod_{i \neq j} \frac{(x - x_i)}{(x_j - x_i)}$$

$$w(x) = \prod_{m=0}^{n} (x - x_m)$$

classmate

Date ________

Page ________

$$w(x) = (x - x_j) \cdot \prod_{\substack{i=0 \\ i \neq j}}^{n} (x - x_i) \quad \text{--- } ①$$

$$w'(x_j) = \frac{d}{dx}\left[(x - x_j) \cdot \prod_{\substack{i=0 \\ i \neq j}}^{n} (x - x_i) \right]$$

$$= \prod_{\substack{i=0 \\ i \neq j}}^{n} (x_j - x_i) + \cancel{(x_j - x_j)}^{\,0} \cdot \frac{d}{dx}\left(\prod_{\substack{i=0 \\ i \neq j}}^{n} (x - x_i) \right.$$

$$w'(x_j) = \prod_{\substack{i=0 \\ i \neq j}}^{n} (x_j - x_i) \quad \text{--- } ②$$

from ① and ②

$$l_j(x) = \frac{w(x)}{(x - x_j) \cdot w'(x_j)} \quad , \text{ for } j = 0 \ldots n$$

$l_j(x) = \delta_{ij}$ where δ_{ij} is Dirac Delta Function.

Hence - $p(x) = f_0 l_0(x) + \ldots + f_n l_n(x)$

interpolates the dataset $(x_0, f_0) \ldots (x_n, f_n)$.

The basis $l_0(x), l_1(x) \ldots l_n(x)$ is called Lagrange basis of P_n and $p(x)$ is called Lagrange interpolating polynomial.

IMP $$\sum_{j=0}^{n} l_j(x) = 1$$

→ Computation of $p(x)$ at given x require $O(n^2)$ operation

why ?

Calculating single $l_j(x)$ takes multiplyin n terms so takes $O(n)$.

Total time = $O(n^2)$

→ Cannot accomodate new (x_{n+1}, f_{n+1}). Any additional data requires recomputation.

A

DOUBT -

Computation of $p(x)$ requires $O(n^2)$ operations?

BARYCENTRIC LAGRANGE INTERPOLATION

$$l_j = \frac{w(x)}{(x-x_j)\cdot w'(x_j)} = w(x)\cdot\frac{w_j}{(x-x_j)}$$

where $w_j = \dfrac{1}{w'(x_j)}$

$$p(x) = \sum_{j=0}^{n} f_j(x)\, l_j(x)$$

Barycentric Form 1

$$= w(x)\sum_{j=0}^{n}\frac{f_j(x)\cdot w_j}{x-x_j} \quad \text{— ①}$$

Also,

$$1 = \sum_{j=0}^{n} l_j(x)$$

$$1 = w(x)\sum_{j=0}^{n}\frac{w_j}{x-x_j} \quad \text{— ②}$$

① ÷ ②

$$\frac{p(x)}{1} = \cancel{\sum w_j f_j(x)} = \frac{\displaystyle\sum_{j=0}^{n}\frac{w_j\cdot \cancel{w(x)}\cdot f_j}{x-x_j}}{\displaystyle\sum_{j=0}^{n}\frac{w_j\cdot \cancel{f_j}\, \cancel{w(x)}}{x-x_j}}$$

$$P(x) = \frac{\sum_{j=0}^{n} \frac{w_j f_j}{x - x_j}}{\sum_{j=0}^{n} \frac{w_j}{x - x_j}}$$ { Barycentric Form 2 }

Advantages -

① Once I know w_j. Evaluating $p(x)$ at $x = x_k$ takes $O(n)$ time.

However calculating every w_j takes $O(n^2)$.

→ $O(n)$ for one j.

→ $O(n^2)$ for all j.

② To Incorporate a new data point (x_{n+1}, f_{n+1})

ⓐ updating on existing weights takes $O(2n+1)$ operations.

$$w_j^{old} = \frac{1}{\prod_{\substack{k=0 \\ k \neq j}}^{n} (x_j - x_k)}$$

$$w_j^{new} = \frac{1}{\prod_{\substack{k=0 \\ k \neq j}}^{n+1} (x_j - x_k)} = \frac{w_j^{old}}{(x_j - x_{n+1})}$$

(b) Computing the new weight ω_{n+1} takes $O(2n+3)$

$$\omega_{n+1} = \frac{1}{\prod_{k=0}^{n} (x_{n+1} - x_k)}$$

(3) Barycentric formulas has beautiful symmetry. The weights ω_j appear in denominator exactly as in numerator, except with factor f_j. This means any common factor in all the weights ω_j may be cancelled without affecting the value of $p(x)$.

LEC - 6

NEWTON INTERPOLATING POLYNOMIAL

Example - For data set (x_0, f_0), (x_1, f_1), (x_2, f_2)

$$p(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1)$$

$$p(x_0) = f_0$$
$$p(x_1) = f_1$$
$$p(x_2) = f_2$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & (x_1 - x_0) & 0 \\ 1 & (x_2 - x_0) & (x_2 - x_0)(x_2 - x_1) \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} f_0 \\ f_1 \\ f_2 \end{bmatrix}$$

General -

$$N_0(x) = 1$$
$$N_j(x) = (x - x_0)(x - x_1) \dots (x - x_{j-1})$$
for $j = 1$ to N

$N_j(x_i) = 0$ for $i = 0 : j-1$ and
$j = 1 : N$

$$\boxed{N_{j+1}(x) = N_j(x) \cdot (x - x_j)}$$

$N_0(x)$, $N_1(x)$ ..... $N_n(x)$ → Newton's Basis

$$p(x) = a_0 N_0(x) + a_1 N_1(x) + \dots + a_n N_n(x)$$

$p(x_j) = f_j$ for $j = 0:n$ yields-

$$\begin{bmatrix} N_0(x_0) & 0 & 0 & \cdots & 0 \\ N_0(x_1) & N_1(x_1) & 0 & \cdots & 0 \\ N_0(x_2) & N_1(x_2) & N_2(x_2) & \cdots & 0 \\ \vdots & \vdots & \vdots & & \\ N_0(x_n) & N_1(x_n) & N_2(x_n) & \cdots & N_n(x_n) \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix}$$

Remark-

→ Solution of lower Δ system requires $O(n^2)$ flops.

→ Can accomodate new data (x_{n+1}, f_{n+1}) with addition $O(n)$ operations

→ Evaluation of $p(x)$ at a given x require $O(n^2)$ operations.

→ Computation of $p(x)$ requires $O(n^2)$ operations.

Computation of $N_j(x_j)$ may be prone OVERFLOW / UNDERFLOW.

ASK

Ask remark 2

DIVIDED DIFFERENCE -

- For example - consider (x_0, f_0), (x_1, f_1), (x_2, f_2)

~~a~~ Newton's Polynomial is -

$$p(x) = a_0 + a_1(x - x_0) + a_2(x - x_1)(x - x_0)$$

$$f[x_j] := f_j \quad \text{for} \quad j = 0, 1, 2$$

$$p(x_0) = f_0, \quad p(x_1) = f_1, \quad p(x_2) = f_2$$

$$a_0 = f[x_0] = f_0$$

$$\cancel{a_1 = f[x_1]}$$

$$\cancel{a_2 = f[x_2]}$$

$$a_1 = \frac{f[x_1] - f[x_0]}{x_1 - x_0} = f[x_0, x_1]$$

$$a_2 = \left(\frac{f[x_2] - f[x_0]}{x_2 - x_0} - f[x_0, x_1] \right) \Big/ (x_2 - x_1)$$

$$= \frac{\left(\frac{f[x_2] - f[x_1]}{x_2 - x_0} + \frac{f[x_1] - f[x_0]}{x_2 - x_0} - f[x_0, x_1] \right)}{(x_2 - x_1)}$$

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

Date
Page

General case–

Divided difference can be generated using table of divided difference.

diagonal - left

x	f			
x_0	f_0			
x_1	f_1	$f[x_0, x_1]$		
x_2	f_2	$f[x_1, x_2]$	$f[x_0, x_1, x_2]$	
x_3	f_3	$f[x_2, x_3]$	$f[x_1, x_2, x_3]$	$f[x_0, x_1, x_2, x_3]$
⋮				

$$f[x_0, x_1, x_2] = \frac{(\ \ ) - (\ \ )}{x_2 - x_0}$$

go to diagonal

extreme – left of x

$n+1$ diagonals $\longrightarrow$ coefficients of Newton Interpolating polynomial

Adding new data (x_{n+1}, f_{n+1}). $\rightarrow$ Adding a new row at the bottom of the table. Additional $O(n)$ operations.

$\hookrightarrow$ Underflow / Overflow problem is solved.

Thm - Let $p(x) = a_0 N_0(x) + \cdots + a_n N_n(x)$

such that $p(x_j) = f_j$ for $j = 0:n$.

then,

$$a_j = f[x_0, x_1, \ldots x_j] \quad \text{for } j = 0:n$$

Proof - Proof using induction.

The result is true for $n=0$.

Assume that the result is true for Newton interpolating polynomials of degree $\leq N-1$.

Let $q(x) = \sum_{j=0}^{n-1} b_j N_j(x)$

↳ for data set $(x_1, f_1) \ldots\ldots (x_n, f_n)$

let $s(x) = \sum_{j=0}^{n-1} c_j N_j(x)$

↳ for dataset $(x_0, f_0) \ldots\ldots (x_{n-1}, f_{n-1})$

By induction hypothesis $b_{n-1} = f[x_1, x_2 \ldots x_n]$
$c_{n-1} = f[x_0, x_1 \ldots x_{n-1}]$

~~$r(x) =$~~

For the dataset $(x_0, f_0) \ldots\ldots (x_n, f_n)$
$r(x)$ be interpolating polynomial.

$$\boxed{r(x) = q(x) + \frac{x - x_n}{x_n - x_0}\left(q(x) - s(x)\right)}$$

→ $r(x)$ satisfies for datas - $(x_1, f_0) \ldots (x_{n-1}, f_{n-1})$

because $q(x) - s(x) = 0$

$r(x) = q(x)$.

→ For (x_0, f_0)

$$r(x) = q(x_0) \div 1 \, (q(x_0) - s(x_0))$$

$$= s(x_0) = f_0$$

→ For (x_n, f_n)

$$r(x_n) = q(x_n) + 0$$
$$= f_n$$

Hence $r(x)$ interpolates $(x_0, f_0) \ldots (x_n, f_n)$

Now the coefficient of x^n in $r(x)$

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

$$a_n = f[x_0, x_1, \ldots x_n]$$

RUNGE PHENOMENON

consider the runge function

$f : [-1, 1] \to R$ given by

$$f(x) = \frac{1}{1+25x^2}$$

Then for equally spaced nodes $(x_0, \ldots x_n)$ and values $f_j = f(x_j)$ for $j = 0 : n$

The interpolant $p_n(x)$ doesn't converge to $f(x)$. In fact $\max_{|x| \le 1} | f(x) - p_n(x) | \to \infty$ as $n \to \infty$

But for Chebyshev Nodes $x_j = \cos\left(\frac{(2j+1)\pi}{2n+2}\right)$.

for $j = 0 : n$ ~~art~~ $\max_{|x| \le 1} |f(x) - p_n(x)| \to 0$

as $n \to \infty$.

The runge phenomenon is eliminated by choosing Chebyshev nodes in interpolation points in $[-1, 1]$

Date ______
Page ______

CHEBYSHEV POLYNOMIAL

Let $\theta \in [0, \pi]$
$x \in [-1, 1]$

Define $T_n(x) = \cos(n \cos^{-1} x)$.

$T_0(x) = 1$
$T_1(x) = x$
$T_2(x) = 2x^2 - 1$

$T_n(x)$ is n degree polynomial and is called Chebyshev Polynomial.

$T_n(x)$ satisfies - $T_{n+1}(x) = 2x T_n(x) - T_{n-1}(x)$

Why? If $x = \cos\theta$

$$\cos((n+1)\theta) + \cos((n-1)\theta) = 2\cos\theta \cos n\theta$$

$$\begin{bmatrix} T_1(x) \\ T_2(x) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & 2x \end{bmatrix} \begin{bmatrix} 1 \\ x \end{bmatrix}$$

$$\begin{bmatrix} T_n(x) \\ T_{n+1}(x) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & 2x \end{bmatrix} \begin{bmatrix} T_{n-1}(x) \\ T_n(x) \end{bmatrix}$$

$$\begin{bmatrix} T_n(x) \\ T_{n+1}(x) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & 2x \end{bmatrix}^n \begin{bmatrix} 1 \\ x \end{bmatrix}$$

$$T_n(x) = \sum_{j=0}^{\lfloor n/2 \rfloor} \binom{n}{2j} x^{n-2j} (x^2-1)^j$$

Chebshev NODES-

$$T_n(x) = \cos(n \cos^{-1} x)$$

we have $|T_n(x)| \leq 1$ ~~for x~~

CHEBYSHEV NODES $\Rightarrow T_n(x_j) = 0$ for $x_j = \cos\left(\frac{2j+1}{2n} \cdot \pi\right)$

$j = 0, 1, \ldots, n-1$

GAUSS LOBATTO NODES $\Rightarrow T_n(x_j) = (-1)^j$ for $x_j = \cos\left(\frac{j\pi}{n}\right)$

$j = 0, 1, \ldots, n$

~~Chebyshev Nodes~~

Connection of Chebshev nodes-

We want build $p(x)$ of degree $\leq n$ through $n+1$ Chebshev nodes.

The zeros of $T_{n+1}(x)$ will give exactly $n+1$ points in $[-1,1]$ i.e - Chebshev nodes will be zeros of $T_{n+1}(x)$.

classmate

Date ______

Page ______

Barycentric Lagrange Interpolation with Chebyshev Nodes

$$p(x) = \frac{\sum_{j=0}^{n} w_j f_j / x - x_j}{\sum_{j=0}^{n} w_j / x - x_j}$$

where $w_j = \frac{1}{\prod_{i \neq j} (x_j - x_i)}$ for $j = 0 : n$

For chebyshev nodes in $[-1,1]$ -

$$x_j = \cos\left(\frac{j\pi}{n}\right) \quad j = 0, 1, \ldots, n$$

$$w_j = \begin{cases} \frac{1}{2}(-1)^j & j=0 \text{ or } j=n. \\ (-1)^j & \text{otherwise} \end{cases}$$

$$p(x) = \frac{\sum_{j=0}^{n}{}' (-1)^j f_j / (x - x_j)}{\sum_{j=0}^{n}{}' w_j(-1)^j / (x - x_j)}$$

$\sum'$ means that terms $j=0$ and $j=n$ are multiplied by $1/2$.

NOTE - Barycentric interpolation formula remains valid for Chebyshev nodes $[a,b]$

Approximation

Let $C[a,b] := \{f : [a,b] \to \mathbb{R} \mid f \text{ is cont}$

For $f \in C[a,b]$, define

$$\|f\|_\infty = \max\{|f(x)| \; x \in [a,b]\}$$

① $\|f\|_\infty = 0 \iff f = 0$

② $\|\alpha f\|_\infty = \cancel{\alpha} \; |\alpha| . \|f\|_\infty$

③ $\|f+g\|_\infty \leq \|f\|_\infty + \|g\|_\infty$

Weierstrass approx theorem:
Let $f \in C[a,b]$ and $\epsilon > 0$. Then there is a polynomial $p(x)$ such that $\|f - p\|_\infty < \epsilon$.

or

$$\max\{|f(x) - p(x)| : x \in [a,b]\} \leq \epsilon$$

Question - Does $p_n(x)$ approx $f(x)$ for large enough n? In other words, does $\|p_n - f\|_\infty \to$ as $n \to \infty$?

ANS - NO, for equispaced we saw $n \to \infty$ the error was ∞

Date
Page

Interpolation error -

Let $f \in C[a,b]$ and $[x_0, \dots x_n]$ be distinct nodes in $[a,b]$, $f_j = f(x_j)$ for $j = 0:n$.

Lagrange polynomial -

$$p(x) = f_0 l_0(x) + f_1 l_1(x) + \dots + f_n l_n(x)$$

Define :

{LEBESGUE FUNC} $\lambda_n(x) = |l_0(x)| + \dots + |l_n(x)|$

{LEBESGUE CONST} $\Lambda_n = \|\lambda_n\|_\infty$

Set :

$$E_n(f) = \min\{\|f - p\|_\infty : p \in P_n\}$$

Proof 1 - $|p_n(x)| \leq \Lambda_n \|f\|_\infty$

Proof -
$$\begin{aligned} |p_n(x)| &= \left|\sum_{j=0}^{n} f_j l_j(x)\right| \\ &\leq \sum_{j=0}^{n} |f_j||l_j(x)| \\ &\leq \|f\|_\infty \sum_{j=0}^{n} |l_j(x)| \\ &\leq \|f\|_\infty \lambda_n(x) \end{aligned}$$

$$\begin{aligned} \|p_n(x)\| &\leq \|f\|_\infty \|\lambda_n\|_\infty \\ &= \|f\|_\infty \Lambda_n \end{aligned}$$

$|p_n(x)| \leq \|f\|_\infty \Lambda_n$ {Point wise also holds true}

Proof 2 - $\|f - P_n\|_\infty \leq (1 + \Lambda_n) E_n(f)$

Proof -

→ Interpolation error is at most the best possible polynomial error multiplied by $1+\Lambda_n$. If Λ_n is large → worse. Small Λ_n → better.

→ Λ_n depends only on nodes not on f. So node choice matters.

FACT - for equispaced nodes, ~~the Runge function $f(x) =$~~ $\Lambda_n \sim \frac{2^n}{e n \log n}$ and for Chebyshev $\Lambda_n \leq \frac{2 \log}{\pi}$

ERROR TERM FOR SMOOTH FUNCTION -

Let $C^n[a,b]$ denote the set of n times continuously differentiable functions in $[a,b]$.

Theorem -

If $f \in C^n[a,b]$ and $p_n(x)$ be unique polynomial of degree at most n passing through $(x_0, f(x_0)), \ldots, (x_n, f(x_n))$

$$\boxed{f(x) - p_n(x) = \frac{f^{(n+1)}(\theta x)}{(n+1)!} \cdot w(x)}$$

for some $\theta_x \in [x_{min}, x_{max}]$ where x_{min} and x_{max} are the largest and the smallest nodes in $[x_0, x_1, \ldots x_n, x]$ and $w(x) = \prod_{i=0}^{n}(x - x_i)$

Proof -

so, $|f(x) - P_n(x)| \leq \frac{\|f^{(n+1)}\|_\infty}{(n+1)!} \cdot |w(x)|$

ERROR

Error is dependent on $w(x)$, so we want to minimize $|w(x)|$

Chebyshev's Thm -

Goal - Choose nodes $x_0, \ldots x_n$ in $[a,b]$ that minimizes $\max_{x \in [a,b]} \prod_{j=0}^{n} |x - x_j|$

$$\min_{x_0, \ldots x_n} \max_{x \in [a,b]} \prod_{j=0}^{n} |x - x_j|$$

ASK - If proof is there or not.

classmate

Date

Page

Thm- In $x \in [-1,1]$

$$\min_{x_0, x_1, \dots x_n} \max_{x \in [-1,1]} \prod_{j=0}^{n} |(x-x_j)| = 2^{-n}$$

and the minimum is attained when

$$\omega(x) = \prod_{j=0}^{n} (x-x_j) = 2^{-n} T_{n+1}(x)$$

Minimum is attained when $x_0, \dots x_n$ are Chebyshev nodes in $[-1,1]$.

Hence,

$$|f(x) - P_n(x)| \leq \frac{\|f^{(n+1)}\|_\infty}{(n+1)! \; 2^n}$$