Introduction to Computation (CS2201)

Lecture 3

Kripabandhu Ghosh

CDS, IISER Kolkata

INTERPOLATION

Interpolation

Premise

- The analytical form of a function $f(x)$ is unknown
- The values of $f(x)$ at some points x_0, x_1, ..., x_n are known to be $f(x_0)$, $f(x_1)$, ..., $f(x_n)$

Assumption

f is continuously differentiable sufficient number of times

Problem

Calculate $f(x_i)$ where x_i is distinct from the above tabulated points (x_0, x_1, ..., x_n) but is in the vicinity of these points

Solution using Interpolation

Since the formula for $f(x)$ is not known, the approximate value of the same can be determined by approximating f by another function $\phi(x)$

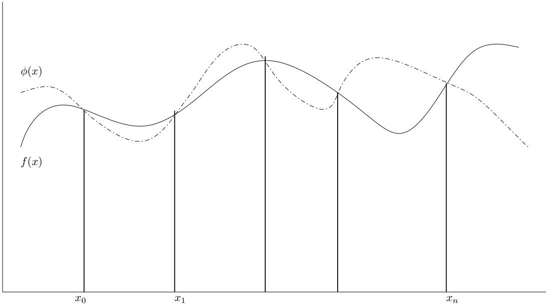
Interpolation function

Features

- $\phi(x)$ coincides with $f(x)$ at the tabular points $(x_0, x_1, ..., x_n)$
- That is, $f(x) \simeq$ such that

$$\phi(x_i) = f(x_i)$$

$i = 1, 2, ..., n$



Interpolation function (contd.)

Choice

- Linear interpolation: $\phi(x)$ is a linear function
- Polynomial interpolation: $\phi(x)$ is a polynomial function

Vote for Polynomial

- Weierstrass's polynomial approximation theorem states that if $f(x)$ is continuous in [a, b] then given $\epsilon > 0$, we can find a polynomial $P_\epsilon(x)$ such that $|f(x) - P_\epsilon(x)| < \epsilon$ for every $x \in$ [a, b]
- That is, a polynomial is supposed to be capable of approximating any $f(x)$

Newton's Forward Interpolation

Formula

$f(x) = y = y_0 + \binom{u}{1} \Delta y_0 + \binom{u}{2} \Delta^2 y_0 + \ldots + \binom{u}{n} \Delta^n y_0$

- $x_i = x_0 + i*h$ (equispaced points)
- $u = \frac{x - x_0}{h}$
- $\Delta y_i = y_{i+1}$ - y_i, $i = 0, 1, \ldots$
- $\Delta^k y_i = \Delta^{k-1} y_{i+1}$ - $\Delta^{k-1} y_i$, $i = 0, 1, \ldots; k = 2, 3, \ldots$

Application

Useful for computing $f(x)$ for values of x at the beginning of the table

x_i	y_i	Δy_i	$\Delta^2 y_i$	...	$\Delta^n y_i$
x_0	y_0	Δy_0	$\Delta^2 y_0$	...	$\Delta^n y_0$
x_1	y_1	Δy_1	$\Delta^2 y_1$		
x_2	y_2	Δy_2			
...	...				
x_n	y_n				

Newton's Forward Interpolation (example)

Actual function

f(x) = Sin(x)

x_i	y_i	Δy_i	$\Delta^2 y_i$	$\Delta^3 y_i$
45	0.7071	0.0589	-0.0057	-0.0007
50	0.766	0.0532	-0.0064	
55	0.8192	0.0468		
60	0.866			

Result

- Interpolated value at $x = 46$ is 0.719302
- $f(46) = 0.719340$

Newton's Backward Interpolation

Formula

$$f(x) = y = y_n + \binom{u}{1} \Delta y_{n-1} + \binom{u}{2} \Delta^2 y_{n-2} + \ldots + \binom{u+n-1}{n} \Delta^n y_0$$

- $x_i = x_0 + i*h$ (equispaced points)
- $u = \frac{x-x_n}{h}$
- $\Delta y_{n-i} = y_{n-i}$ - y_{n-i-1}, $i = 1, 2, ..$

Application

Useful for computing $f(x)$ for values of x at the **end** of the table

x_i	y_i	Δy_i	$\Delta^2 y_i$	...	$\Delta^n y_i$
x_0	y_0				
x_1	y_1	Δy_0			
...	...				
x_{n-1}	y_{n-1}	Δy_{n-2}			
x_n	y_n	Δy_{n-1}	$\Delta^2 y_{n-2}$	...	$\Delta^n y_0$

Newton's Backward Interpolation (example)

Actual function

f(x) = Sin(x)

x_i	y_i	Δy_i	$\Delta^2 y_i$	$\Delta^3 y_i$
45	0.7071			
50	0.766	0.0		
55	0.8192	0.0532	0	
60	0.866	0.0468	-0.0064	0

Result

- Interpolated value at $x = 56$ is 0.823952
- $f(56) = 0.829038$

Lagrange Interpolation

Formula

$$f(x) = y = \frac{(x-x_1)(x-x_2)\dots(x-x_n)}{(x_0-x_1)(x_0-x_2)\dots(x_0-x_n)}\ y_0 + \frac{(x-x_0)(x-x_2)\dots(x-x_n)}{(x_1-x_0)(x_1-x_2)\dots(x_1-x_n)}\ y_1 + \dots + \frac{(x-x_0)(x-x_1)\dots(x-x_{n-1})}{(x_n-x_0)(x_n-x_1)\dots(x_n-x_{n-1})}\ y_n$$

Application

Useful for computing $f(x)$ for values of x which are not equispaced.

Lagrange Interpolation (example)

Actual function

f(x) = Sin(x)

x_i	45	50	56	63
y_i	0.7071	0.7660	0.8290	0.89100

Result

- Interpolated value at $x = 52$ is 0.788009
- $f(52) = 0.788011$

Disadvantage

For every new point, the whole calculation needs to be repeated.

scipy interpolate function

Format

scipy.interpolate.interp1d(x, y, kind='linear', ...)

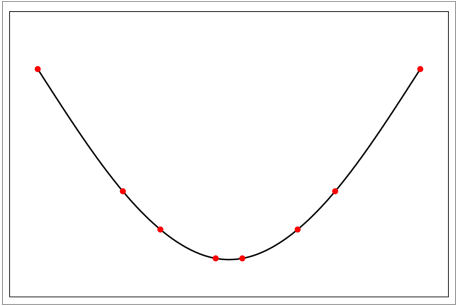
Features

- x: A 1-D array of real values.
- y: A N-D array of real values. The length of y along the interpolation axis must be equal to the length of x
- kind: Specifies the kind of interpolation as a string or as an integer specifying the order of the *spline interpolator* to use.
  - The string has to be one of *linear*, *nearest*, *nearest-up*, *zero*, *slinear*, *quadratic*, *cubic*, *previous*, or *next*
  - *zero*, *slinear*, *quadratic*, *cubic* refer to a spline interpolation of zeroth, first, second or third order
  - *previous*, or *next* simply return the previous or next value of the point;
  - Default is *linear*.

Spline Interpolation

Description

- Interpolant is a special type of piecewise polynomial called a spline
- Instead of fitting a single, high-degree polynomial to all of the values at once, spline interpolation fits low-degree polynomials to small subsets of the values. For example, fitting nine cubic polynomials between each of the pairs of ten points, instead of fitting a single degree-ten polynomial to all of them



THANK YOU !!!