Lecture 20: Polynomial Approximations

(10-Spt-2012)

1

POLYNOMIAL APPROXIMATIONS, INTERPOLATIONS (Contd.)

LECTURE 20
10-SPT-2012

In the last class we discussed that if a set of discrete data points are given, then we can approximate a function to the given set of discrete data.

The approximate functions can be:

→ Polynomial Approximations
→ Trigonometric Approximations
→ Exponential Approximations, etc.

Q: Why do we require such approximations?

The actual function $f(x)$ may be complicated.

So if we want to do

→ Interpolation
→ Extrapolation
→ Differentiation
→ Integration etc

of the given data, it may be tedious or non-possible through exact function $f(x)$. Therefore in place of $f(x)$ we require an easier approximate functions to evaluate.

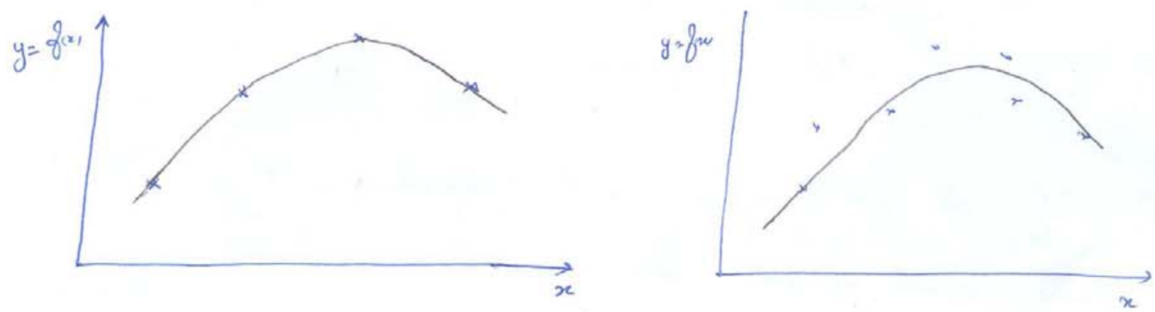
APPROXIMATIONS USING POLYNOMIALS

→ In this chapter we will be dealing with approximate functions that are polynomials.

Your polynomial approximation can be

→ Exactly fitting the data points

→ Approximately fitting the data points.



Therefore $f(x) \approx P_n(x)$

* As you are aware:

→ For a first degree polynomial $P_1(x) = a_0 + a_1 x$ you require minimum two ordered pairs (x_0, y_0) and (x_1, y_1).

→ Similarly, for second degree polynomial you require (x_0, y_0), (x_1, y_1), and (x_2, y_2).

→ That is for a n^{th} degree polynomial approximation

$$P_n(x) = a_0 + a_1 x + a_2 x^2 + \dots + a_n x^n$$

we require a minimum of $(n+1)^{th}$ number of data points.

→ For $(n+1)$ data points on the x-y plane, one can fit polynomials ranging from $P_1(x)$ to $P_n(x)$.

→ The $P_n(x)$ polynomial will be <u>unique</u>.

If you look into Taylor's series based on known point x_0.

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{1}{2!}f''(x_0)(x - x_0)^2 + \cdots$$

If you are approximating $f(x) \approx P_n(x)$, then

$$f(x) = P_n(x) + \underbrace{\frac{1}{(n+1)!}f^{(n+1)}(\xi)(x - x_0)^{n+1}}_{\text{Error Term}} \qquad x_0 \le \xi \le x$$

Once you approximate by polynomials, then you can do

→ Differentiation $\quad \frac{dP_n(x)}{dx} = P_n'(x) = a_1 + 2a_2 x + \cdots + n a_n x^{n-1}$

$= P_{n-1}(x)$

Ily $P_n''(x) = P_{n-2}(x)$

or

→ Integration $\quad I = \int P_n(x)\, dx = P_{n+1}(x)$

Now look into the following polynomial

$$P_4(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4$$

→ This involves

$x * x * x * x * a_4$ → 4 multiplication

$x * x * x * a_3$ → 3 multiplications

$x * x * a_2$ → 2 "

$x * a_1$ → 1

No: of addition = 4

i.e Total of 14 operations are performed.

⇒ Use Nested Algorithm procedure :

$$P_4(x) = a_0 + x(a_1 + x(a_2 + x(a_3 + a_4 x)))$$

Number of operations →

$x * a_4$ → 1 multiplication

$a_3 + a_4 x$ → 1 addition

$x * (a_3 + a_4 x)$ → 1 multiplication

$a_2 + x * (a_3 + a_4 x)$ → 1 addition

$x * (a_2 + x * (a_3 + a_4 x))$ → 1 multiplication

$a_1 + x * (\quad)$ → 1 addition

$x * (a_1 + \cdots)$ → 1 multiplication

$a_0 + x * (a_1 \cdots)$ → 1 addition

→ Total 8 operations.

It is better to use nested algorithm for computer methods.

Nested algorithm is :

$$P_n(x) = a_0 + x(a_1 + x(a_2 + \cdots + \underbrace{x(a_{n-1} + \underbrace{a_n x}_{b_n}))\cdots)}_{b_{n-1}}$$

$= \cancel{b_0 \pm}$

$$b_n = a_n$$
$$b_i = a_i + x\, b_{i+1} \quad ; \quad i = n-1, n-2, \ldots, 1, 0$$

$\Rightarrow$ Using nested algorithm you can do synthetic division

Say if you want to factor $(x-\alpha)$ from $P_n(x)$

$$P_n(x) = (x-\alpha)\, Q_{n-1}(x) + R$$

$$\therefore P_n(\alpha) = R$$

$$P_n'(x) = Q_{n-1}(x) + (x-\alpha)\, Q_{n-1}'(x)$$

Again $P_n'(\alpha) = Q_{n-1}(\alpha)$

That is the first derivative of the n^{th} degree polynomial evaluated from $(n-1)^{st}$ degree polynomial $Q_{n-1}(x)$.

$$Q_{n-1}(x) = b_1 + b_2 x + b_3 x^2 + \cdots + b_n x^{n-1}$$

$$\left.\begin{aligned} b_n &= a_n \\ b_{n-1} &= a_{n-1} + x\, b_n \\ &\vdots \\ b_1 &= a_1 + x b_2 \\ b_0 &= a_0 + x b_1 = R \end{aligned}\right\} \rightarrow \text{Horner's algorithm.}$$

$i = n-1, n-2, \ldots, 1, 0$

Direct - Fit Polynomials

In the given $(n+1)$ data sets $(x_0, f_0), (x_1, f_1), \ldots, (x_n, f_n)$ the unique polynomial approximation is:

$$P_n(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$$

Now substitute

$$f_0 = a_0 + a_1 x_0 + a_2 x_0^2 + \cdots + a_n x_0^n$$

$$f_1 = a_0 + a_1 x_1 + a_2 x_1^2 + \cdots + a_n x_1^n$$

$$\vdots$$

$$f_n = a_0 + a_1 x_n + a_2 x_n^2 + \cdots + a_n x_n^2$$

Here you get unknown $a_0, a_1, a_2, \ldots, a_n$ solved by linear system methods.

Example :

x	$f(x)$
3.35	0.298507
3.40	0.294118
3.50	0.285714
3.60	0.277778

Interpolate $f(3.43)$.

Let us approximate by second-degree polynomial

$$P_2(x) = a_0 + a_1 x + a_2 x^2$$

Now form a polynomial using known points enclosing the requested point 3.43.

$\therefore$ Select $(x_0, f_0) \Rightarrow$ 3.35 0.298507

$(x_1, f_1) \rightarrow$ 3.40 0.294118

$(x_2, f_2) \rightarrow$ 3.50 0.285714

$\therefore \quad y_0 = 0.298507 = a_0 + 3.35\, a_1 + 11.2225\, a_2$

$y_1 = 0.294118 = a_0 + 3.40\, a_1 + 11.56\, a_2$

$y_2 = 0.285714 = a_0 + 3.50\, a_1 + 12.25\, a_2$

Solve this to get a_0, a_1 and a_2.

LAGRANGE POLYNOMIALS

In direct-fit you required to solve system to get coefficients

→ May become tedious.

Therefore, a better approach is to use Lagrange Polynomials

⇒ If you have two data points (x_0, y_0) and (x_1, y_1) then

$$P_1(x) = \frac{(x - x_1)}{(x_0 - x_1)} y_0 + \frac{(x - x_0)}{(x_1 - x_0)} y_1$$

You can check for $P_1(x_1)$ and $P_1(x_0)$.

⇒ If you have three points (x_0, y_0), (x_1, y_1), (x_2, y_2) you can fit a quadratic polynomial (Lagrange) as

$$P_2(x) = \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} f_0 + \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} f_1$$

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

In a general way, you can also form a n^{th} degree polynomial $P_n(x)$.

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

$$+ \cdots\cdots + \frac{(x-x_0)(x-x_1)\cdots(x-x_{n-1})}{(x_n-x_0)(x_n-x_1)\cdots(x_n-x_{n-1})} f_n$$