보간 (Interpolation) 및 회귀 (Regression)

김기택

국민대학교 소프트웨어학과

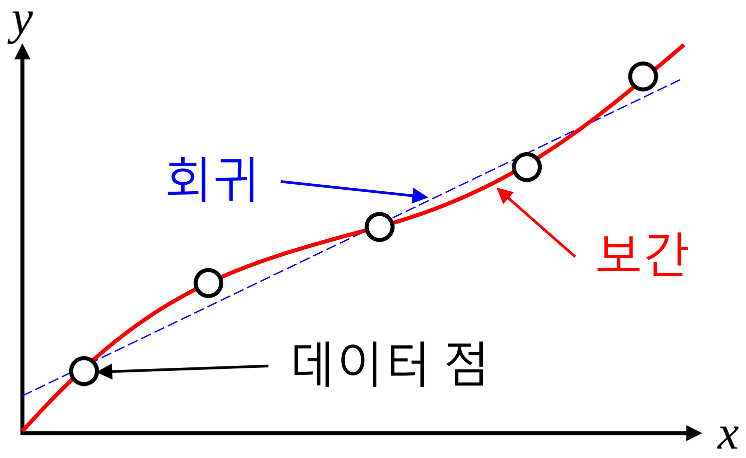
개요

- 보간과 회귀는 다음 경우에 해당한다.
  - $n + 1$ 개의 데이터 점 (x_i, y_i), i=0, 1, 2, …, n 이 주어질 때 $y(x)$ 를 추정

- 일반적으로 이산 데이터가 주어지는 경우에 보간이나 회귀를 실행한다.

x_0	x_1	x_2	…	x_n
y_0	y_1	y_2	…	y_n

- 보간과 회귀의 차이
  - 보간: 주어진 모든 점을 통과하는 곡선을 구한다.
  - 회귀: 데이터와 오차를 최소로 하는 직선이나 곡선을 구한다.



다항식 보간
(polynomial interpolation)

Lagrange 다항식 보간

- 보간법 중 가장 간단한 형태는 다항식이다. $n+1$ 개의 개별 데이터 포인트를 통과하는 차수 n 의 고유 다항식을 구성하는 것이 항상 가능하다. (n 차 다항식의 계수는 $n+1$ 개이다.)

$$y(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \cdots + a_n x^n$$

- 이 다항식을 얻는 한 가지 방법은 **Lagrange 공식**이다.

$$P_n(x) = \sum_{i=0}^{n} y_i l_i(x) \qquad (3.1a)$$

컴퓨터를 이용하는데 알맞은 방법

  - 여기서 첨자 n 은 다항식의 차수를 나타낸다.

$$l_i(x) = \frac{x - x_0}{x_i - x_0} \cdot \frac{x - x_1}{x_i - x_1} \cdots \frac{x - x_{i-1}}{x_i - x_{i-1}} \cdot \frac{x - x_{i+1}}{x_i - x_{i+1}} \cdots \frac{x - x_n}{x_i - x_n} = \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j} \qquad (3.1b)$$

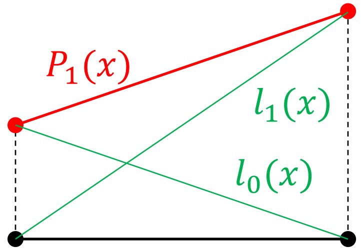
  - 식 (3.1b) 의 ***$l_i(x)$* 를 기수(cardinal) 함수**라고 한다.

Cardinal function: 기준이 되는 함수

Lagrange 다항식 선형 보간

- 예를 들어 $n = 1$ 인 경우 보간 함수는 직선 $P_1(x) = y_0 l_0(x) + y_1 l_1(x)$ 이다.
  - 여기서

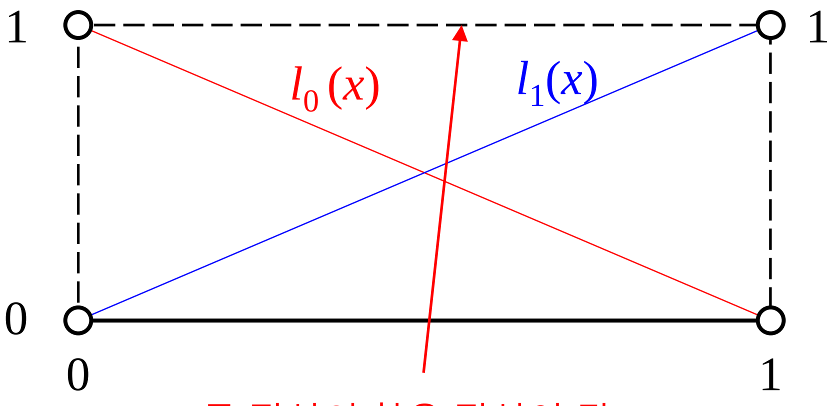
$$l_0(x) = \frac{x - x_1}{x_0 - x_1} \qquad l_1(x) = \frac{x - x_0}{x_1 - x_0}$$



- $n = 1$ 의 경우 $x_0 = 0, x_1 = 1$ 에 대해 그래프로 나타내면 다음과 같다.

$$l_0(x) = \frac{x - x_1}{x_0 - x_1}$$

0 에서 1이 되고 선형으로 감소하여
1 에서 0이 되는 직선



두 직선의 합은 직선이 됨

$$l_1(x) = \frac{x - x_0}{x_1 - x_0}$$

1 에서 1이 되고 선형으로 감소하여
0 에서 0이 되는 직선

Lagrange 다항식 2차 보간

- $n = 2$ 인 경우, 보간은 포물선 $P_2(x) = y_0 l_0(x) + y_1 l_1(x) + y_2 l_2(x)$이며, 기수함수는

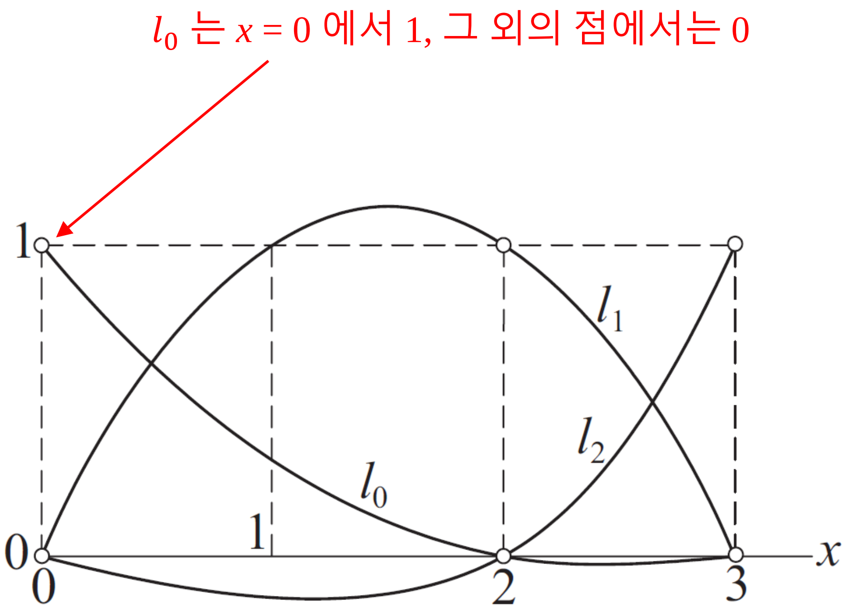
$$l_0(x) = \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)}$$
$$l_1(x) = \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)}$$
$$l_2(x) = \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)}$$

- 기수 함수는 차수 n 의 다항식이며

$$l_i(x_j) = \begin{Bmatrix} 0 \text{ if } i \neq j \\ 1 \text{ if } i = j \end{Bmatrix} = \delta_{ij} \quad (3.2)$$

  - 여기서 δ_{ij} 는 크로네커 델타 (Kronecker delta) 이다.

- 이를 $x_0 = 0, x_1 = 2, x_2 = 3$ 인 경우에 대해 3점 보간을 하면 그래프는 옆 그림과 같아 진다.



다항식 보간 오차

- 보간 다항식이 데이터 포인트를 통과함을 증명하려면 $x = x_j$ 를 공식 (3.1a) 에 대입한 후, 식 (3.2)를 사용한다. 결과는 다음과 같다.

$$P_n(x_j) = \sum_{i=0}^{n} y_i l_i(x_j) = \sum_{i=0}^{n} y_i \delta_{ij} = y_j$$

- 다항식 보간 오차는 다음과 같다.

$(n + 1)$ 차 도함수

$$f(x) - P_n(x) = \frac{(x - x_0)(x - x_1) \dots (x - x_n)}{(n+1)!} f^{(n+1)}(\xi)$$

  - 여기서 ξ 는 간격 (x_0, x_n) 사이에 존재하지만 정확한 위치는 알 수 없다.
  - 데이터 포인트가 x 에서 멀수록 x 에서의 오류에 더 많이 기여한다는 점에 유의한다.

예제 3.1

다음 주어진 데이터 점들에 대해 Lagrange 보간법을 사용하여 $x = 1$ 에서 y 값을 추정하라.

x	0	2	3
y	7	11	28

주어진 점이 3개 이므로 $n = 2$

[풀이]

$$l_0 = \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} = \frac{(1-2)(1-3)}{(0-2)(0-3)} = \frac{1}{3}$$

$$l_1 = \frac{(x-x_0)(x-x_2)}{(x_1-x_1)(x_1-x_2)} = \frac{(1-0)(1-3)}{(2-0)(2-3)} = 1$$

$$l_2 = \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} = \frac{(1-0)(1-2)}{(3-0)(3-2)} = -\frac{1}{3}$$

$$y = y_0 l_0 + y_1 l_1 + y_2 l_2 = \frac{7}{3} + 11 - \frac{28}{3} = 4$$

Newton 보간법

- Lagrange 방법은 개념적으로 간단하지만 효율적인 알고리즘에는 적합하지 않다. 보간 다항식이 다음과 같은 형식으로 작성되는 **Newton 방법이 더 나은 계산 절차**를 얻는다.

$$P_n(x) = a_0 + (x - x_0)a_1 + (x - x_0)(x - x_1)a_2 + \cdots + (x - x_0)(x - x_1)\cdots(x - x_{n-1})a_n$$

- 이 다항식은 효율적인 평가 절차에 적합하다. 예를 들어 4개의 데이터 포인트 $(n = 3)$를 고려할 때 보간 다항식은 아래와 같다.

$$\begin{aligned} P_3(x) &= a_0 + (x - x_0)a_1 + (x - x_0)(x - x_1)a_2 + (x - x_0)(x - x_1)(x - x_2)a_3 \\ &= a_0 + (x - x_0)\left\{\underbrace{a_1 + (x - x_1)\left[\underbrace{a_2 + (x - x_2)a_3}_{\text{1차 다항식}}\right]}_{\text{2차 다항식}}\right\} \end{aligned}$$

- 다음과 같은 반복 관계를 통해 역으로 평가할 수 있다.

$$\begin{aligned} P_3(x) &= a_0 + (x - x_0)P_2(x) \\ P_2(x) &= a_1 + (x - x_1)P_1(x) \\ P_1(x) &= a_2 + (x - x_2)P_0(x) \\ P_0(x) &= a_3 \end{aligned}$$

Newton 다항식 표현

- 임의의 n 에 대해 다음과 같은 재귀적인(recursive) 표현이 만들어진다.

$$P_0(x) = a_n \qquad P_k(x) = a_{n-k} + (x - x_{n-k})P_{k-1}(x), \qquad k = 1, 2, \ldots, n \quad (3.4)$$

- 이를 구현하는 알고리즘은 다음과 같다. (x 좌표 배열은 xData 에 저장됨)

```
p = a[n]
for k in range(1, n+1):
    p = a[n-k] + (x – xData[n-k]) * p
```

- P_n 의 계수는 다항식이 각 데이터 포인트를 통과하도록 결정된다. $y_i = P_n(x_i), i = 0,1, \ldots, n$
  - 다음과 같은 방정식을 산출한다.

$$\begin{aligned}
y_0 &= a_0 \\
y_1 &= a_0 + (x_1 - x_0)a_1 \\
y_2 &= a_0 + (x_2 - x_0)a_1 + (x_2 - x_0)(x_2 - x_1)a_2 \qquad \text{(a)} \\
&\vdots \\
y_n &= a_0 + (x_n - x_0)a_1 + \cdots + (x_n - x_0)(x_n - x_1)\cdots(x_n - x_{n-1})a_n
\end{aligned}$$

Newton 다항식 계수의 분할차분 표현

- 여기에서 분할 차분(divided difference)를 소개하면,

$$\nabla y_i = \frac{y_i - y_0}{x_i - x_0}, \qquad i = 1,2,\ldots,n$$

$$\nabla^2 y_i = \frac{\nabla y_i - \nabla y_1}{x_i - x_1}, \qquad i = 2,3,\ldots,n$$

$$\nabla^2 y_i = \frac{\nabla y_i - \nabla y_1}{x_i - x_1} = \frac{\dfrac{y_i - y_0}{x_i - x_0} - \dfrac{y_1 - y_0}{x_1 - x_0}}{x_i - x_1} = \frac{(y_i - y_0)(x_1 - x_0) - (y_1 - y_0)(x_i - x_0)}{(x_i - x_1)(x_i - x_0)(x_1 - x_0)}$$

$$\nabla^3 y_i = \frac{\nabla^2 y_i - \nabla^2 y_2}{x_i - x_2}, \qquad i = 3,4,\ldots,n$$

$$\vdots$$

$$\nabla^n y_n = \frac{\nabla^{n-1} y_n - \nabla^{n-1} y_{n-1}}{x_n - x_{n-1}}$$

- 방정식 (a) 를 분할차분 표현으로 나타내어 계수를 구하면,

$$a_0 = y_0 \quad a_1 = \nabla y_1 \quad a_2 = \nabla^2 y_2 \quad \cdots \quad a_n = \nabla^n y_n$$

$$y_1 = a_0 + (x_1 - x_0)a_1$$

$$a_1 = \frac{y_1 - a_0}{x_1 - x_0} = \frac{y_1 - y_0}{x_1 - x_0}$$

$$y_2 = a_0 + (x_2 - x_0)a_1 + (x_2 - x_0)(x_2 - x_1)a_2$$

$$a_2 = \frac{y_2 - a_0 - (x_2 - x_0)a_1}{(x_2 - x_0)(x_2 - x_1)} = \frac{(y_2 - y_0)(x_1 - x_0) - (y_1 - y_0)(x_2 - x_0)}{(x_2 - x_1)(x_2 - x_0)(x_1 - x_0)} = \nabla^2 y_2$$

Newton 다항식 계수 계산

- 계수를 수동으로 계산하는 경우 표를 이용하면 편리하다.
  - 대각선 항이 다항식의 계수이다. $n+1$ 개의 개별 데이터 포인트를 보간하는 차수 n 의 다항식은 고유하다.

a_0 → **인덱스 i 진행**

인덱스 k 진행

x_0	y_0	a_1			
x_1	y_1	∇y_1	a_2		
x_2	y_2	∇y_2	$\nabla^2 y_2$	a_3	
x_3	y_3	∇y_3	$\nabla^2 y_3$	$\nabla^3 y_3$	a_4
x_4	y_4	∇y_4	$\nabla^2 y_4$	$\nabla^3 y_4$	$\nabla^4 y_4$

- 이를 다음 알고리즘을 사용하여 계수 계산을 수행할 수 있다.
  - 인덱스 k 루프(k 개 포인트)를 진행함에 각 루프 안에서 표의 오른쪽으로 진행한다.

```
a = yData.copy()
for k in range(1, m):            # m = n+1 – number of data point   a0 는 계산할 필요 없음
    for i in range(k, m):
        a[i] = (a[i] – a[k-1]) / (xData[i] – xData[k-1])   유한 차분 계산
```

Newton 다항식 보간 프로그램

- newtonPoly 모듈은 다음 2가지를 수행한다.
  - 주어진 xData, yData 에 대해 계수를 계산한다.
  - 완성된 다항식에 따라 요구된 지점, *x* 에서 값을 계산한다.

```
## module newtonPoly
''' p = evalPoly(a,xData,x).
    Evaluates Newton's polynomial p at x. The coefficient
    vector {a} can be computed by the function 'coeffts'.
    a = coeffts(xData,yData).
    Computes the coefficients of Newton's polynomial.
'''

def evalPoly(a, xData, x):
    n = len(xData) - 1        # Degree of polynomial
    p = a[n]
    for k in range(1,n+1):
        p = a[n-k] + (x -xData[n-k])*p
    return p
```

```
def coeffts(xData, yData):
    m = len(xData)                    # Number of data points
    a = yData.copy()
    for k in range(1,m):                          배열 연산 (vectorization)
        a[k:m] = (a[k:m] - a[k-1])/(xData[k:m] - xData[k-1])
    return a    계수 배열 반환
```

예제 3.2

다음 데이터 포인트에 대해 Newton 다항식 보간을 하라. 표 3.1과 유사한 표를 작성하여 진행하라.

x	-2	1	4	-1	3	-4
y	-1	2	59	4	24	-53

[풀이]

표의 값을 계산하는 몇 가지 예를 보자.

$$\nabla y_2 = \frac{y_2 - y_0}{x_2 - x_0} = \frac{59 - (-1)}{4 - (-2)} = 10$$

$$\nabla^2 y_2 = \frac{\nabla y_2 - \nabla y_1}{x_2 - x_1} = \frac{10 - 1}{4 - 1} = 3$$

$$\nabla^3 y_5 = \frac{\nabla^2 y_5 - \nabla^2 y_2}{x_5 - x_2} = \frac{-5 - 3}{-4 - 4} = 1$$

예제 3.2 (continued)

앞의 공식에 따라 표의 값을 계산하면, 3차 다항식 임을 알 수 있다.

i	x_i	y_i	∇y_i	$\nabla^2 y_i$	$\nabla^3 y_i$	$\nabla^4 y_i$	$\nabla^5 y_i$
0	-2	-1 a_0					
1	1	2	1 a_1				
2	4	59	10	3 a_2			
3	-1	4	5	-2	1 a_3		
4	3	24	5	2	1	0	
5	-4	-53	26	-5	1	0	0

3차(cubic) 다항식

0

Neville 보간법 (1)

- Newton 보간법에서는 동일한 다항식을 사용하여 여러 위치의 x 값의 보간값을 계산하는데 유용하다. 그러나, **다항식 대신 한 점에서만 보간값**을 구하고 싶을 때는 Neville 보간법을 사용하여 한 단계로 계산하는 것이 더 편리하다.
  - $P_k[x_i, x_{i+1}, \dots, x_{i+k}]$ 는 $k+1$ 데이터 포인트 $(x_i, y_i), (x_{i+1}, y_{i+1}), \dots, (x_{i+k}, y_{i+k})$ 를 통과하는 k 차수 다항식을 나타낸다고 하자.
- 단일 데이터 포인트의 경우, 다음과 같은 식을 가진다.

$$P_0[x_i] = y_i$$

  - 두 데이터 포인트를 기반으로 하는 보간은

$$P_1[x_i, x_{i+1}] = \frac{(x - x_{i+1})P_0[x_i] + (x_i - x)P_0[x_{i+1}]}{x_i - x_{i+1}}$$

양끝점 간의 간격

  - $P_1[x_i, x_{i+1}]$ 가 두 데이터 포인트를 통과하는지 쉽게 확인할 수 있다. $x = x_i$ 인 경우$P_1[x_i, x_{i+1}] = y_i$ 이고, $x = x_{i+1}$ 인 경우$P_1[x_i, x_{i+1}] = y_{i+1}$ 이다.
  - 세 데이터 포인트를 기반으로 하는 보간은 다음과 같다.

$$P_2[x_i, x_{i+1}, x_{i+2}] = \frac{(x - x_{i+2})P_1[x_i, x_{i+1}] + (x_i - x)P_1[x_{i+1}, x_{i+2}]}{x_i - x_{i+2}}$$

Neville 보간법 (2)

- 이 보간식이 각각의 데이터 포인트를 통과하는지 알아 보자
  - 먼저 $x = x_i$ 를 대입하면,

$$P_2[x_i, x_{i+1}, x_{i+2}] = P_1[x_i, x_{i+1}] = y_i$$

  - 다음으로 $x = x_{i+2}$ 의 경우

$$P_2[x_i, x_{i+1}, x_{i+2}] = P_1[x_{i+1}, x_{i+2}] = y_{i+1}$$

  - 마지막으로 $x = x_{i+1}$ 의 경우는

$P_1[x_i, x_{i+1}] = P_1[x_{i+1}, x_{i+2}] = y_{i+1}$

$$P_2[x_i, x_{i+1}, x_{i+2}] = \frac{(x_{i+1} - x_{i+2})\boxed{P_1[x_i, x_{i+1}]} + (x_i - x_{i+1})\boxed{P_1[x_{i+1}, x_{i+2}]}}{x_i - x_{i+2}}$$

$$= \frac{(x_{i+1} - x_{i+2})y_{i+1} + (x_i - x_{i+1})y_{i+1}}{x_i - x_{i+2}} = y_{i+1}$$

- 이 과정의 패턴을 통해 다음 재귀 공식을 추론할 수 있다.

$$P_k[x_i, x_{i+1}, \dots, x_{i+k}] = \frac{(x - x_{i+k})P_{k-1}[x_i, x_{i+1}, \dots, x_{i+k-1}] + (x_i - x)P_{k-1}[x_{i+1}, x_{i+2}, \dots, x_{i+k}]}{x_i - x_{i+k}}$$

Neville 보간 알고리즘

- x 값이 주어지면 계산은 다음 표 형식으로 수행할 수 있다. (4개의 데이터 포인트, $k = 3$)

인덱스 i 진행

	$k=0$	$k=1$	$k=2$	$k=3$
x_0	$P_0[x_0]=y_0$	$P_1[x_0,x_1]$	$P_2[x_0,x_1,x_2]$	$P_3[x_0,x_1,x_2,x_3]$
x_1	$P_0[x_1]=y_1$	$P_1[x_1,x_2]$	$P_2[x_1,x_2,x_3]$	
x_2	$P_0[x_2]=y_2$	$P_1[x_2,x_3]$		
x_3	$P_0[x_3]=y_3$			

인덱스 k 진행

- 데이터 포인트 수를 m 이라 하면, 알고리즘은 다음과 같다.

```
y = yData.copy()
for k in range(1, m):          # m = n+1 – number of data point
    for i in range(m - k):
        y[i] = ((x – xData[i+k])*y[i] + (xData[i] – x)*y[i+1]) / (xData[i] – xData[i+k])
```

- k 루프가 끝나면 표의 대각선 요소가 y 에 저장된다.

Neville 다항식 보간 프로그램

- 다음 프로그램은 주어진 데이터 포인트를 통과하는 보간식에 따른 x 위치의 보간값을 반환한다.

```
## module neville
''' p = neville(xData, yData, x).
    Evaluates the polynomial interpolant p(x) that passes
    through the specified data points by Neville's method.
'''
def neville(xData, yData, x):
    m = len(xData)          # number of data points
    y = yData.copy()
    for k in range(1,m):
        y[0:m-k] = ((x - xData[k:m])*y[0:m-k] + (xData[0:m-k] - x)*y[1:m-k+1])/ \
                      (xData[0:m-k] - xData[k:m])
    return y[0]
```

배열 연산 (vectorization)

예제 3.3

다음 데이터 포인트에 대해 Neville 보간법으로 $y(x) = 0$ 의 근을 구하라.

x	4.0	3.9	3.8	3.7
y	-0.06604	-0.02724	0.01282	0.05383

[풀이]

이 문제는 x 와 y 의 역할이 바뀌는 역 보간의 예이다. 주어진 x 에서 y 를 계산하는 대신 주어진 y 에 대해 해당하는 x 를 찾는다. 표 3.2의 형식을 사용하면 (x 와 y 를 바꾸어 실행한다.)

$y = 0$ 에서 x 의 값?

i	y_i	$P_0[\ ] = x_i$	$P_1[\ ,]$	$P_2[\ ,,]$	$P_3[\ ,,,]$
0	-0.06604	4.0	3.8298	3.8316	3.8317
1	-0.02724	3.9	3.8320	3.8318	
2	0.01282	3.8	3.8313		
3	0.05383	3.7			

예제 3.3 (continued)

표에 사용된 계산의 예이다.

$$P_1[y_0, y_1] = \frac{(y - y_1)P_0[y_0] + (y_0 - y)P_0[y_1]}{y_0 - y_1}$$

$$= \frac{(0 + 0.02724)(4.0) + (-0.06604 - 0)(3.9)}{-0.06604 + 0.02724} = 3.8298$$

$$P_2[y_1, y_2, y_3] = \frac{(y - y_3)P_1[y_1, y_2] + (y_1 - y)P_1[y_2, y_3]}{y_1 - y_3}$$

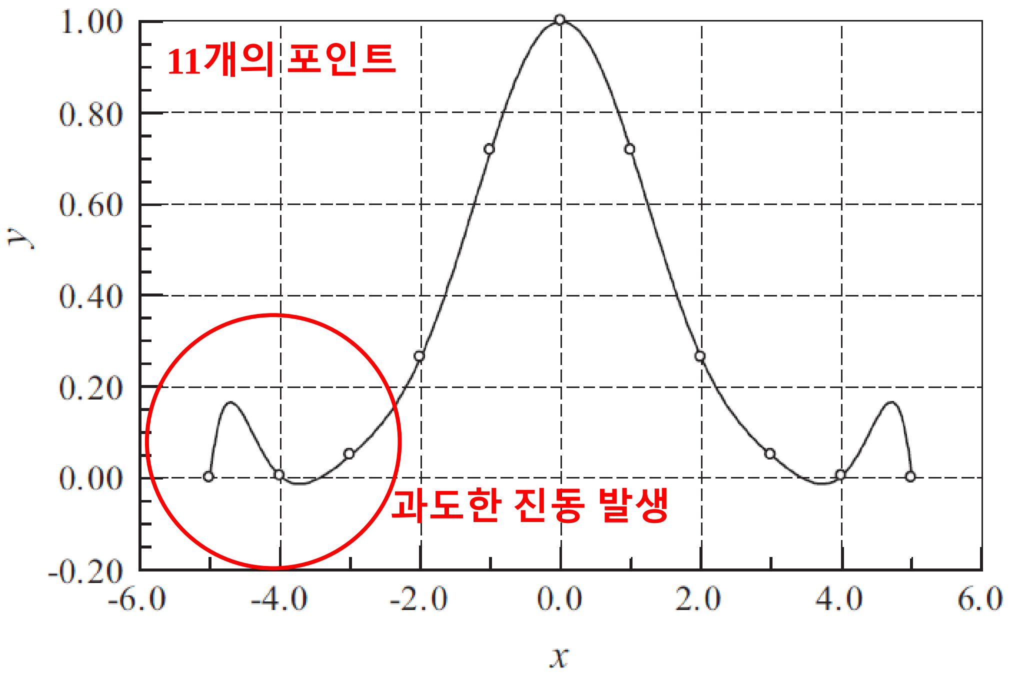
$$= \frac{(0 - 0.05383)(3.8320) + (-0.0274 - 0)(3.8313)}{-0.0274 - 0.05383} = 3.8318$$

표의 모든 P 값은 서로 다른 데이터 포인트를 포함하는 서로 다른 보간 순서로 인해 발생하는 근의 추정값이다. 예를 들어, $P_1[y_0, y_1]$ 은 처음 두 점을 기준으로 선형 보간에서 얻은 근이다. $P_2[y_1, y_2, y_3]$ 는 마지막 세점을 사용한 2차 보간 결과이다. 4개의 모든 데이터에 대해 3차 보간에서 얻은 근은 $x = P_3[y_0, y_1, y_2, y_3] = 3.8317$ 이다.

다항식 보간의 한계

- 다항식 보간은 가능한 적은 수의 데이터 포인트로 수행해야 한다.
  - 두 점 간의 보간은 직선이면 충분하다.
  - 3점에서 6점 사이의 보간은 대부분 좋은 결과를 낸다.
  - 6점 이상의 점을 보간할 때는 관심 지점에서 멀리 떨어진 지점의 데이터가 관심 지점의 정확도에 기여하지 않으며 오히려 해로울 수 있다.

- 과도한 진동을 발생하는 예
  - 관심 지점 주변의 4개의 점으로 3차 보간을 하는 것이 오히려 좋은 결과를 얻을 수 있다.



다항식 외삽(extrapolation)의 위험성

- 다항식으로 외삽을 하는 경우 주어진 데이터 포인트 범위 밖은 위험할 때가 있다.
  - 데이터의 경향에 따라 로그-로그 축의 보간이 더 좋은 결과를 만들 때도 있다.