파이썬으로 배우는
머신러닝의 교과서

Contents

Contents

Contents

Contents

CHAPTER 04 머신러닝에 필요한 수학의 기본

머신러닝을 이해하는 데 필요한 최소한의 프로그래밍 지식을 알아본다

1.1 벡터란

- 벡터는 몇 가지 숫자를 세로로 나란히 나타낸 것.
- 세로로 늘어 놓은 것을 세로 벡터라고 함.
- 옆으로 숫자를 늘어 놓은 것은 가로 벡터라고 함.

$$\mathbf{a} = \begin{bmatrix} 1 \\ 3 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

[세로 벡터]

$$\mathbf{c} = [1\ 2], \quad \mathbf{d} = [1\ 3\ 5\ 4]$$

[가로 벡터]

- 벡터를 구성하는 숫자 하나하나를 요소라 하며, 벡터가 가지는 요소의 수를 벡터의 차원이라고 함.
- '일반적인 숫자의 묶음(집합)'을 스칼라라고 부름.
- T라는 기호는 벡터의 오른쪽 위에 쓰임.
- 세로 벡터를 가로 벡터로, 가로 벡터를 세로 벡터로 변환한다는 의미이며, 이를 전치라고 부름.

$$\mathbf{a}^{\mathrm{T}} = \begin{bmatrix} 1 \\ 3 \end{bmatrix}^{\mathrm{T}} = [1\ 3], \quad \mathbf{d}^{\mathrm{T}} = [1\ 3\ 5\ 4]^{\mathrm{T}} = \begin{bmatrix} 1 \\ 3 \\ 5 \\ 4 \end{bmatrix}$$

1.2 파이썬으로 벡터를 정의하기

- 벡터를 사용하려면 넘파이 라이브러리를 import 함.

In
```
# 리스트 4-1-(1)
import numpy as np
```

- np.array를 사용하여 벡터 a를 정의.

In
```
# 리스트 4-1-(2)
a=np.array([2,1])
print(a)
```

Out
```
[2 1]
```

- type을 사용하면 a는 numpy.ndarray 형임을 알 수 있음.

In
```
# 리스트 4-1-(3)
type(a)
```

Out
```
numpy.ndarray
```

1.3 세로 벡터를 나타내기

- 1차원 ndarray 형은 가로, 세로를 구분하지 않고 항상 가로 벡터로 표시
- 특별한 형태의 2차원 ndarray으로 세로 벡터를 나타낼 수도 있음.

- ndarray 형과 같이 2×2의 2차원 배열(행렬)을 나타냄.

In
```
# 리스트 4-1-(4)
c=np.array([[1,2],[3,4]])
print(c)
```

Out
```
[[1 2]
 [3 4]]
```

- 이 방식으로 2×1의 2차원 배열을 만들면, 세로 벡터를 나타냄

In
```
# 리스트 4-1-(5)
d=np.array([[1],[2]])
print(d)
```

Out
```
[[1]
 [2]]
```

1.4 전치를 나타내기

- 전치는 변수명.T로 나타냄.

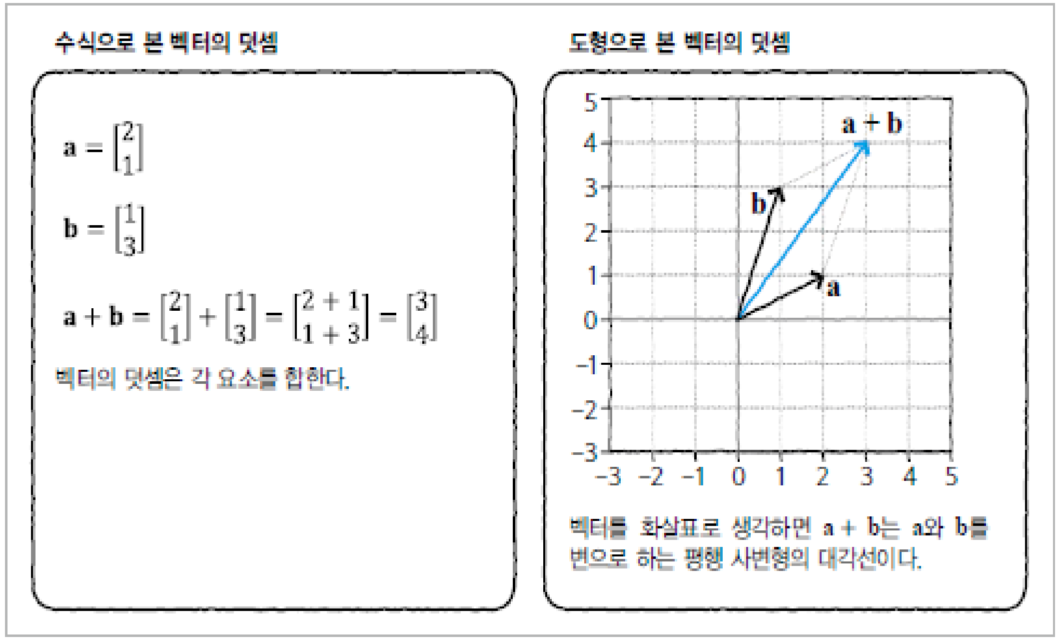
전치는 **변수명.T**로 나타냅니다(리스트 4-1-(6)).

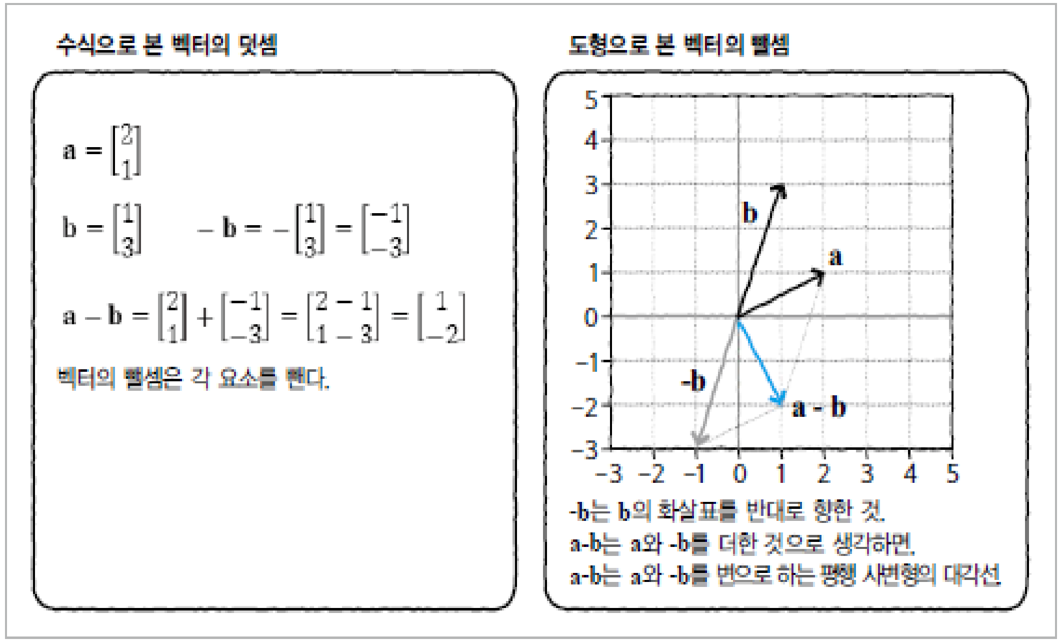
In
```
# 리스트 4-1-(6)
print(d.T)
```

Out
```
[[1 2]]
```

1.5 덧셈과 뺄셈

- 각 요소를 더하는 벡터의 덧셈과 뺄샘은, a와 b를 변으로 하는 평행 사변형의 '대각선을 구하는 연산'으로 해석할 수 있음.





1.6 스칼라의 곱셈

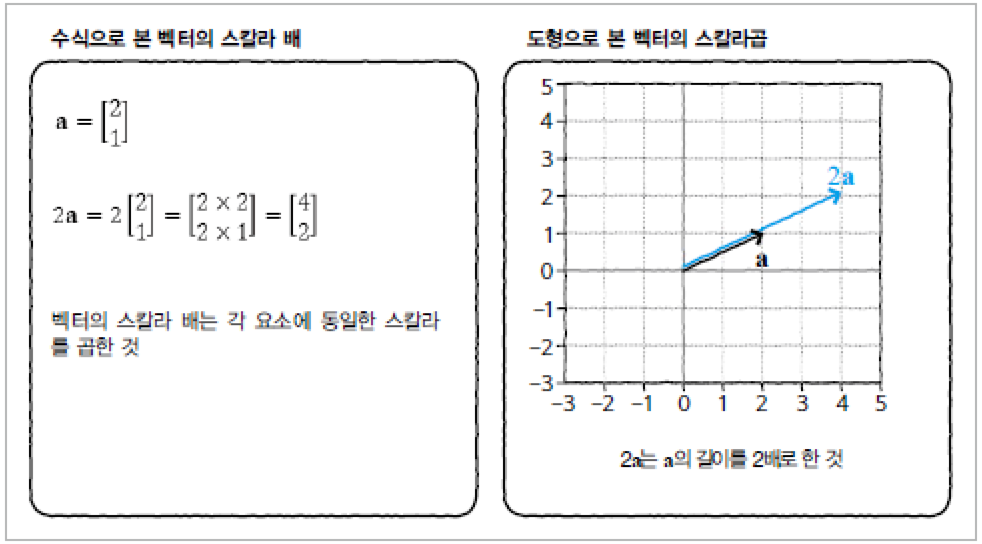
- 스칼라에 벡터를 곱하면 스칼라 값이 벡터의 요소 전체에 적용 됨.

$$2\mathbf{a} = 2\times\begin{bmatrix}2\\1\end{bmatrix} = \begin{bmatrix}2\times2\\2\times1\end{bmatrix} = \begin{bmatrix}4\\2\end{bmatrix}$$

In
```
# 리스트 4-1-(9)
print(2 * a)
```

Out
```
[4 2]
```

[파이썬]



[벡터의 스칼라 배]

1.7 내적

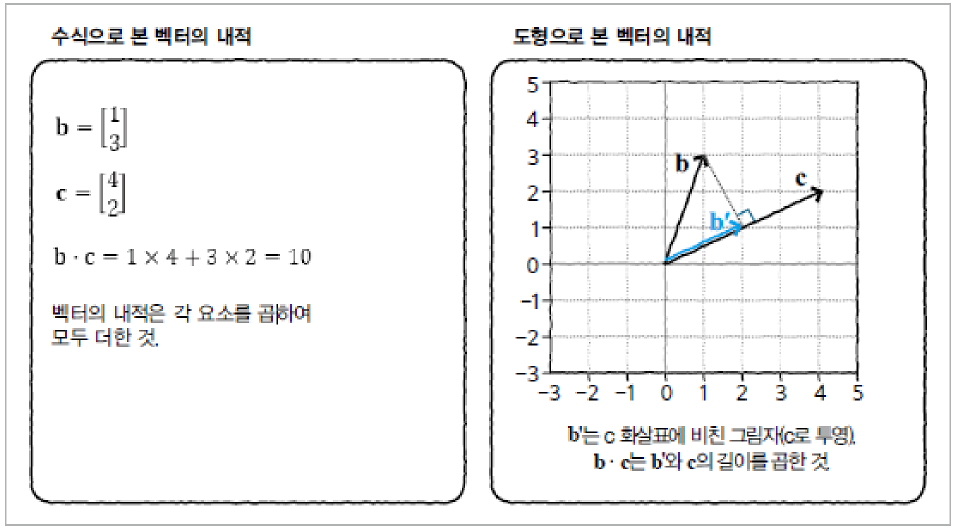
- 내적은 같은 차원을 가진 두 벡터 간의 연산에서 "·"로 나타냄.
- 대응하는 요소들을 곱한 뒤 더한 값을 취함.

$$\mathbf{b} \cdot \mathbf{c} = \begin{bmatrix} 1 \\ 3 \end{bmatrix} \cdot \begin{bmatrix} 4 \\ 2 \end{bmatrix} = 1 \times 4 + 3 \times 2 = 10$$

In
```
# 리스트 4-1-(10)
b = np.array([1, 3])
c = np.array([4, 2])
print(b.dot(c))
```

Out
```
10
```

[파이썬 변수명1.dot(변수명2)로 내적을 계산]



[벡터의 내적]

1.8 벡터의 크기

- 벡터의 크기는 | 와 | 의 사이에 나타냄

$$|\mathbf{a}| = \left\| \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} \right\| = \sqrt{a_0^2 + a_1^2}$$

[2차원 벡터의 크기]

$$|\mathbf{a}| = \left\| \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} \right\| = \sqrt{a_0^2 + a_1^2 + a_2^2}$$

[3차원 벡터의 크기]

$$|\mathbf{a}| = \left\| \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{D-1} \end{bmatrix} \right\| = \sqrt{a_0^2 + a_1^2 + \cdots + a_{D-1}^2}$$

[D차원 벡터의 크기]

- 파이썬에서는 np.linalg.norm () 으로 벡터의 크기를 구할 수 있음.

In

```
# 리스트 4-1-(11)
a = np.array([1, 3])
print(np.linalg.norm(a))
```

Out

```
3.1622776601683795
```

- 합의 기호 Σ (시그마)는 긴 덧셈을 간결하게 나타내는 방법임.

$$\sum_{n=a}^{b} f(n) = f(a) + f(a+1) + \cdots + f(b)$$

$f(n)$의 n을 a부터 1씩 늘려 b가 될 때까지 변화시키고, 모든 $f(n)$을 더한다

$$\sum_{n=1}^{5} n = 1 + 2 + 3 + 4 + 5$$ (식 4-12)

$$\sum_{n=2}^{5} n^2 = 2^2 + 3^3 + 4^4 + 5^2$$ (식 4-14)

$$\sum_{n=1}^{5} 3 = 3 + 3 + 3 + 3 + 3 = 3 \times 5$$ (식 4-15) n과 관계 없이 숫자의 합은, 합의 수만큼 곱한다

$$\sum_{n=1}^{3} 2n^2 = 2\sum_{n=1}^{3} n^2$$ (식 4-16) 스칼라는 Σ의 왼쪽에 적는다

$$\sum_{n=1}^{5} [2n^2 + 3n + 4] = 2\sum_{n=1}^{5} n^2 + 3\sum_{n=1}^{5} n + 4 \times 5$$ (식 4-17) 전개 가능하다

2.1 합의 기호가 들어간 수식을 변형시키기

- 합의 기호의 오른쪽 함수 f 가 n의 함수로 구성되어 있지 않은 경우가 있음.
- 이 경우 더한 횟수만큼 f 를 곱하면 되므로, 합의 기호를 지울 수 있음.

$$\sum_{n=1}^{5} 3 = 3 + 3 + 3 + 3 + 3 = 3\times5 = 15$$

- f(n)이 '스칼라×n의 함수'인 경우는 스칼라를 합의 기호 밖에 낼 수 있음.

$$\sum_{n=1}^{3} 2n^2 = 2\times1^2 + 2\times2^2 + 2\times3^2 = 2(1^2 + 2^2 + 3^2) = 2\sum_{n=1}^{3} n^2$$

- 합의 기호가 다항식에 작용되는 경우, 합의 기호를 각 항으로 나눌 수 있음.

$$\sum_{n=1}^{5} [2n^2 + 3n + 4] = 2\sum_{n=1}^{5} n^2 + 3\sum_{n=1}^{5} n + 4\times5$$

SECTION 02 합의 기호

- 벡터의 내적을 합의 기호를 사용하여 작성할 수 있음.

$$\mathbf{w}\cdot\mathbf{x} = w_0x_0 + w_1x_1 + \cdots + w_{D-1}x_{D-1} = \sum_{i=0}^{D-1} w_i x_i$$

- 행렬과 성분의 표기
- 왼쪽은 '행렬 표기(벡터 표기)', 오른쪽은 '성분 표기'라고 부름.

$$\overbrace{\mathbf{w}\cdot\mathbf{x}}^{\text{행렬 표기(벡터 표기)}} = \overbrace{w_0x_0 + w_1x_1 + \cdots + w_{D-1}x_{D-1} = \sum_{i=0}^{D-1} w_i x_i}^{\text{성분 표기}}$$

$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_{D-1} \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{D-1} \end{bmatrix}$$ 로 한다

2.1 합을 내적으로 계산하기

- Σ는 내적으로도 계산할 수 있음.

$$1+2+\cdots+1000=\begin{bmatrix}1\\1\\\vdots\\1\end{bmatrix}\cdot\begin{bmatrix}1\\2\\\vdots\\1000\end{bmatrix}$$

[1부터 1000까지의 합]

- 파이썬에서는 for 문을 사용하지 않아도 내적으로 계산할 수 있음.

In
```
# 리스트 4-2-(1)
import numpy as np
a = np.ones(1000) # [1 1 1 ... 1]
b = np.arange(1,1001) # [1 2 3 ... 1000]
print(a.dot(b))
```

Out
```
500500.0
```

SECTION 03 곱의 기호

- 합의 기호 Σ와 사용법이 비슷한 곱의 기호 ∏있음.
- ∏의 경우에는 모든 것을 곱함.

$$\prod_{n=a}^{b} f(n) = f(a) \times f(a+1) \times \cdots \times f(b)$$

$$\boxed{\prod_{n=a}^{b} f(n) = f(a) \times f(a+1) \times \cdots \times f(b)}$$

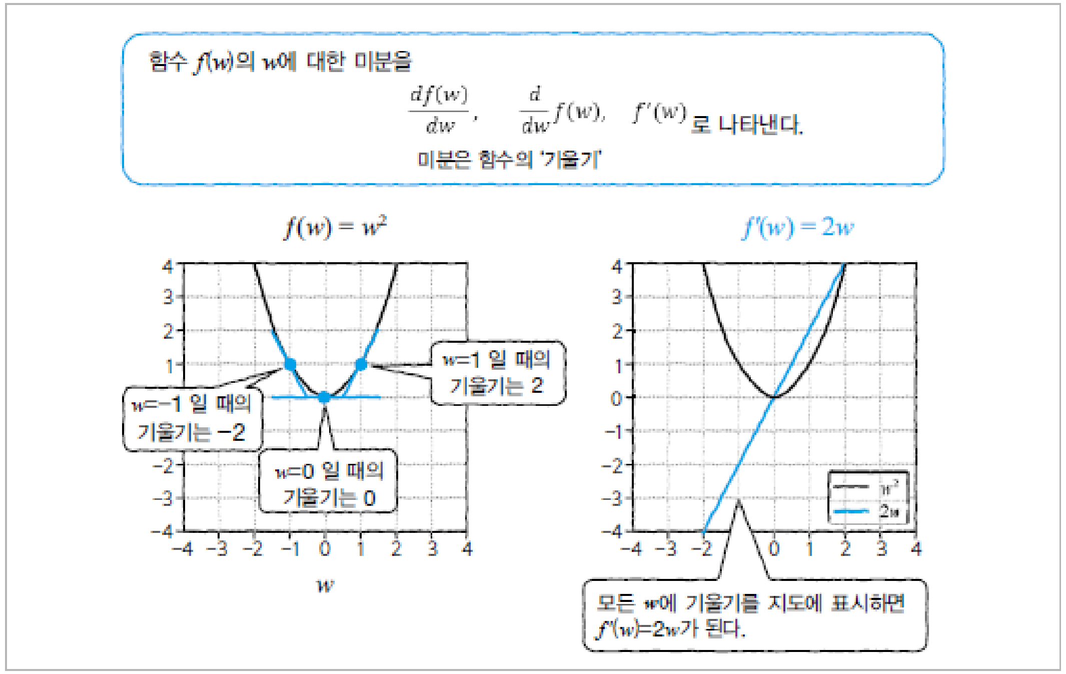
$f(n)$의 n을 a에서 1씩 증가시켜 b가 될 때까지 변화시키고,
모든 $f(n)$을 곱한다

$$\prod_{n=1}^{5} n = 1 \times 2 \times 3 \times 4 \times 5 \qquad \text{(식 4–21)}$$

$$\prod_{n=2}^{5} (2n+1) = (2 \cdot 2 + 1)(2 \cdot 3 + 1)(2 \cdot 4 + 1)(2 \cdot 5 + 1) \qquad \text{(식 4–22)}$$

4.1 다항식의 미분

- 함수의 기울기를 도출하는 방법이 '미분' 임.



[함수의 미분은 기울기]

n차식의 미분 공식

$$\frac{d}{dw}w^n = nw^{n-1} \qquad \text{(식 4-26)}$$

예) $\frac{d}{dw}w^2 = 2w$ (식 4-25)

$\frac{d}{dw}w^4 = 4w^{4-1} = 4w^3$ (식 4-27)

$\frac{d}{dw}w - 1w^{1-1} - 1w^0 - 1$ (식 4-28)

$\frac{d}{dw}(a^3 + xb^2 + 2) = 0$ (식 4-29)

$\frac{d}{dw}(2w^3 + 3w^2 + 2) = 2\frac{d}{dw}w^3 + 3\frac{d}{dw}w^2 + \frac{d}{dw}2 = 6w^2 + 6w$ (식 4-30)

[지수함수의 미분 공식]

4.2 미분 기호가 들어간 수식의 변형

- 미분의 *d/dw* 기호 는 오른쪽에만 작용함.
- 숫자가 *w* 의 앞에서 곱해진 경우, 그 숫자는 미분 기호 왼쪽에 나타낼 수 있음.
- *f(w)*에 *w*가 포함되어 있지 않으면 미분은 0 임.

$$\frac{d}{dw}2w^5 = 2\frac{d}{dw}w^5 = 2\times5w^4 = 10w^4$$

$$\frac{d}{dw}3 = 0$$

- *f(w)*에 *w*가 포함되어 있지 않은 경우

$$f(w) = a^3 + xb^2 + 2$$

$$\frac{d}{dw}f(w) = \frac{d}{dw}(a^3 + xb^2 + 2) = 0$$

- *f(w)* 가 *w를 포함한 여러 항목으로 구성되어 있을 때*

$$f(w) = 2w^3 + 3w^2 + 2$$

$$\frac{d}{dw}f(w) = 2\frac{d}{dw}w^3 + 3\frac{d}{dw}w^2 + \frac{d}{dw}2 = 6w^2 + 6w$$

4.3 중첩된 함수의 미분

- 머신러닝은 중첩 함수의 미분이 많음.

$$f(w) = g(w)^2$$

$$g(w) = aw + b$$

- 간단하게는 위의 식에 대입하고 그 식을 전개하면 미분을 계산할 수 있음.

$$f(w) = (aw + b)^2 = a^2w^2 + 2abw + b^2$$

$$\frac{d}{dw}f(w) = 2a^2w + 2ab$$

4.4 중첩된 함수의 미분: 연쇄 법칙

- 연쇄법칙은 식이 복잡하고 전개하기 힘든 경우 매우 편리하게 사용할 수 있는 공식임.
- 연쇄 법칙(chain rule)이란 미적분학에서, 연쇄 법칙은 함수의 합성의 도함수에 대한 공식을 말함.
- 연쇄 법칙의 공식은 다음과 같음.

중첩 함수의 미분 공식: 연쇄 법칙

$$\frac{d}{dw}f(g(w)) = \frac{df}{dg}\cdot\frac{dg}{dw} \qquad \text{(식 4-35)}$$

예) $f(g(w)) = g(w)^2$, $g(w) = aw + b$ 의 경우

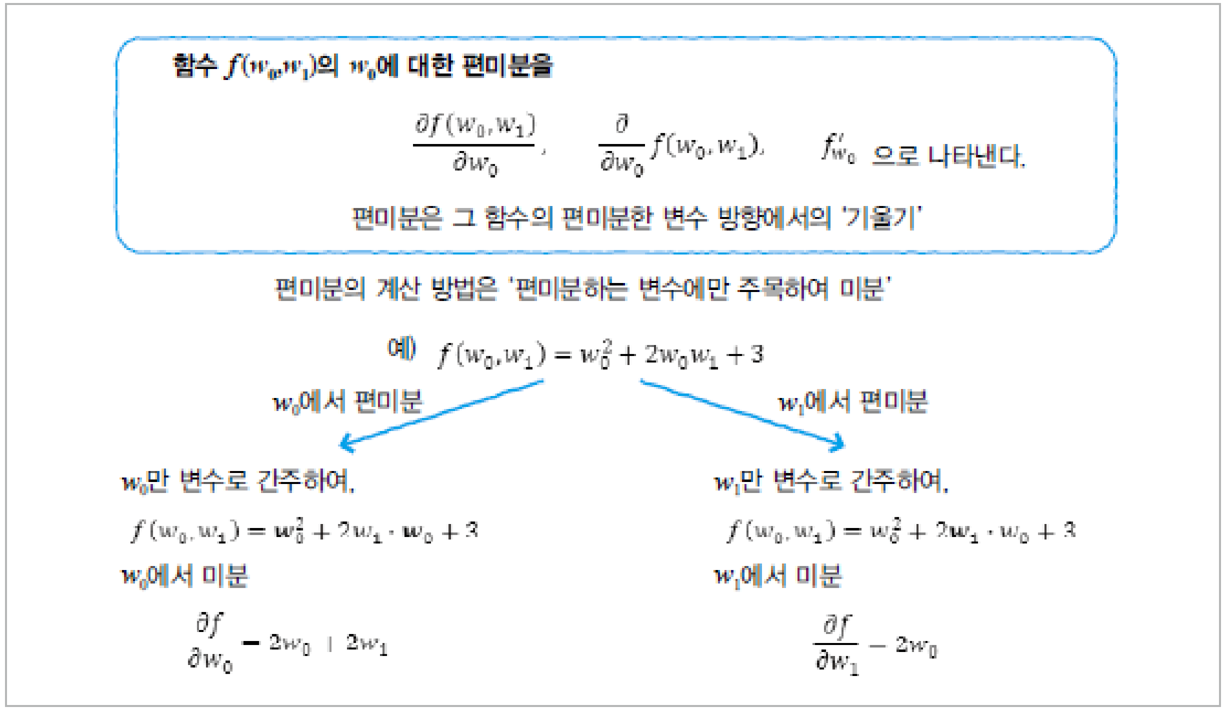
$$\frac{df}{dg} = \frac{d}{dg}g(w)^2 = 2g(w)$$

$$\frac{dg}{dw} = \frac{d}{dw}(aw+b) = a$$ 이기 때문에

$$\frac{df}{dw} = \frac{df}{dg}\cdot\frac{dg}{dw} = 2ga = 2(aw+b)a = 2a^2w + 2ab \qquad \text{(식 4-38)}$$

5.1 편미분이란

- 머신러닝에서 실제로 사용하는 것은 순수한 미분이 아닌, 편미분임.
- 편미분의 계산 방법은 '편미분하는 변수에만 주목해서 미분한다' 임.
- 계산 절차는 보통 미분과 동일함.


함수 $f(w_0,w_1)$의 w_0에 대한 편미분을

$\frac{\partial f(w_0,w_1)}{\partial w_0}$, $\frac{\partial}{\partial w_0} f(w_0,w_1)$, f'_{w_0} 으로 나타낸다.

편미분은 그 함수의 편미분한 변수 방향에서의 '기울기'

편미분의 계산 방법은 '편미분하는 변수에만 주목하여 미분'

예) $f(w_0,w_1) = w_0^2 + 2w_0w_1 + 3$

w_0에서 편미분

w_0만 변수로 간주하여,

$f(w_0,w_1) = w_0^2 + 2w_1 \cdot w_0 + 3$

w_0에서 미분

$$\frac{\partial f}{\partial w_0} = 2w_0 + 2w_1$$

w_1에서 편미분

w_1만 변수로 간주하여,

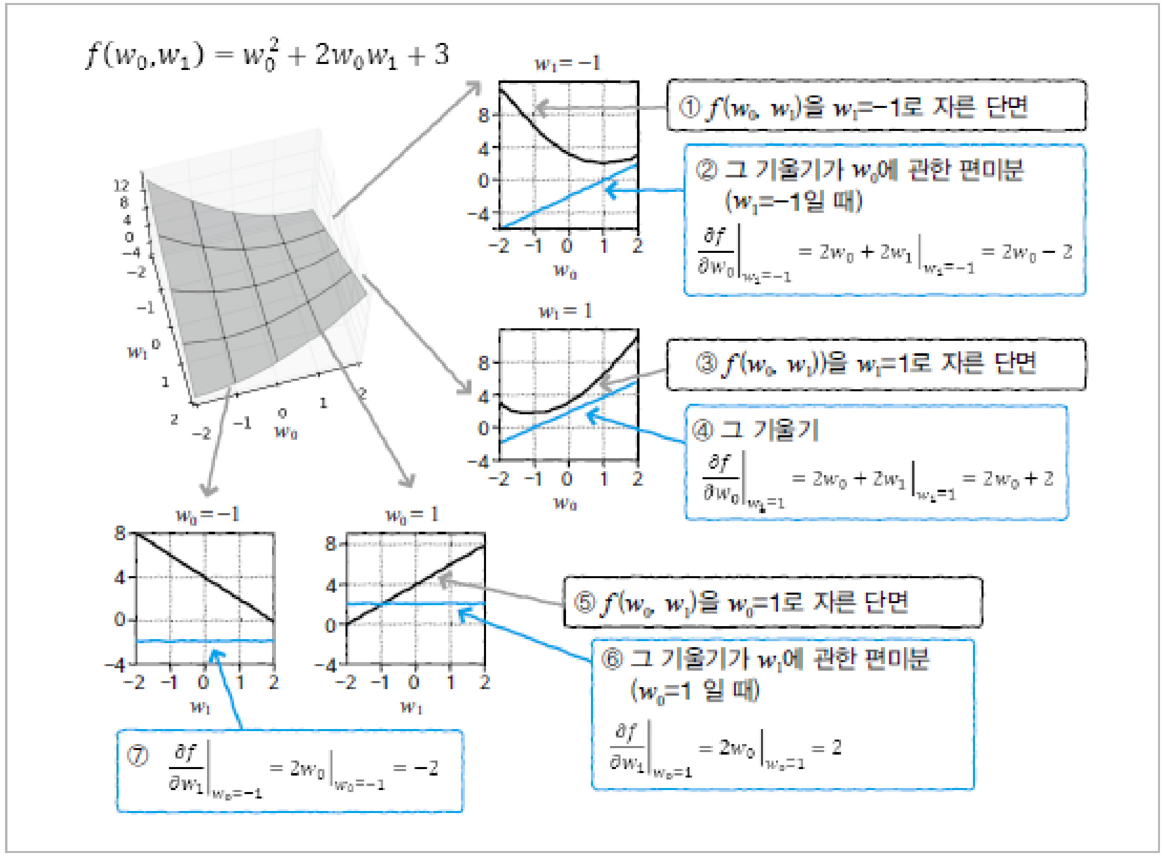
$f(w_0,w_1) = w_0^2 + 2w_1 \cdot w_0 + 3$

w_1에서 미분

$$\frac{\partial f}{\partial w_1} = 2w_0$$

5.2 편미분과 도형

- 편미분의 의미는 다음과 같음.



5.3 경사를 그림으로 나타내기

- 실제로 경사를 그림으로 나타내면 다음과 같음.

In

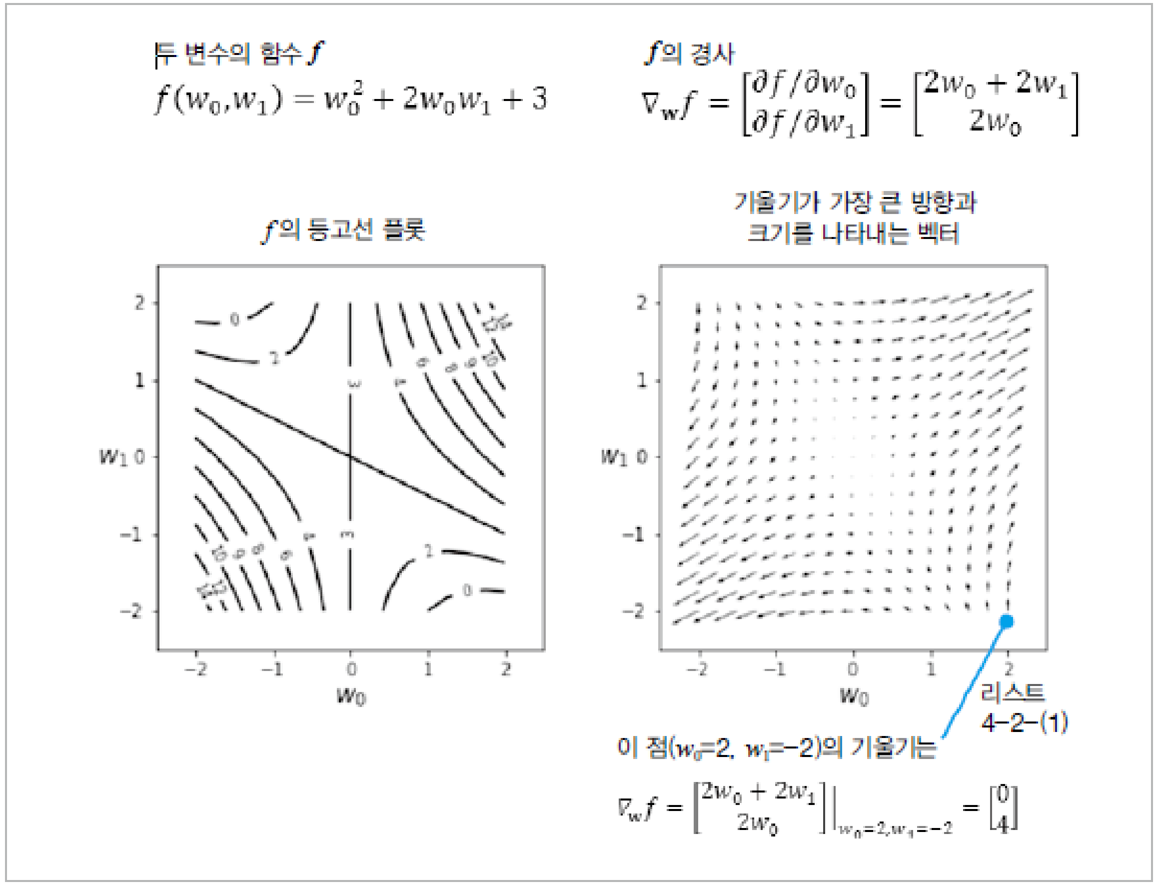
```
# 리스트 4-2-(2)
import numpy as np
import matplotlib.pyplot as plt

def f(w0, w1): # (A) f의 정의
    return w0**2 + 2 * w0 * w1 + 3
def df_dw0(w0, w1): # (B) f의 w0에 관한 편미분
    return 2 * w0 + 2 * w1
def df_dw1(w0, w1): # (C) f의 w1에 관한 편미분
    return 2 * w0 + 0 * w1

w_range = 2
dw = 0.25
w0 = np.arange(-w_range, w_range + dw, dw)
w1 = np.arange(-w_range, w_range + dw, dw)
wn = w0.shape[0]
ww0, ww1 = np.meshgrid(w0, w1) # (D)
ff = np.zeros((len(w0), len(w1)))
dff_dw0 = np.zeros((len(w0), len(w1)))
dff_dw1 = np.zeros((len(w0), len(w1)))
for i0 in range(wn): # (E)
    for i1 in range(wn):
        ff[i1, i0] = f(w0[i0], w1[i1])
        dff_dw0[i1, i0] = df_dw0(w0[i0], w1[i1])
        dff_dw1[i1, i0] = df_dw1(w0[i0], w1[i1])

plt.figure(figsize=(9, 4))
plt.subplots_adjust(wspace=0.3)
plt.subplot(1, 2, 1)
cont = plt.contour(ww0, ww1, ff, 10, colors='k') # (F) f의 등고선 표시
cont.clabel(fmt='%2.0f', fontsize=8)
plt.xticks(range(-w_range, w_range + 1, 1))
plt.yticks(range(-w_range, w_range + 1, 1))
plt.xlim(-w_range - 0.5, w_range + .5)
plt.ylim(-w_range - .5, w_range + .5)
plt.xlabel('$w_0$', fontsize=14)
plt.ylabel('$w_1$', fontsize=14)


plt.subplot(1, 2, 2)
plt.quiver(ww0, ww1, dff_dw0, dff_dw1) # (G) f의 경사 벡터 표시
plt.xlabel('$w_0$', fontsize=14)
plt.ylabel('$w_1$', fontsize=14)
plt.xticks(range(-w_range, w_range + 1, 1))
plt.yticks(range(-w_range, w_range + 1, 1))
plt.xlim(-w_range - 0.5, w_range + .5)
plt.ylim(-w_range - .5, w_range + .5)
plt.show()
```

Out

```
# 실행 결과는 [그림 4-13]을 참조
```

SECTION 05 편미분

그림 4-13 경사 벡터



5.4 다변수의 중첩 함수의 미분

- 다변수 함수(multivariate function)는 둘 이상의 독립 변수를 갖는 함수임.

편미분의 연쇄 법칙

$$\frac{\partial}{\partial w_0} f(g_0(w_0, w_1), g_1(w_0, w_1)) = \frac{\partial f}{\partial g_0}\cdot\frac{\partial g_0}{\partial w_0} + \frac{\partial f}{\partial g_1}\cdot\frac{\partial g_1}{\partial w_0} \qquad \text{(식 4–54)}$$

예) $f = (g_0 + 2g_1 - 1)^2$, $g_0 = w_0 + 2w_1 + 1$, $g_1 = 2w_0 + 3w_1 - 1$ 의 경우

$$\frac{\partial f}{\partial w_0} = \frac{\partial f}{\partial g_0}\cdot\frac{\partial g_0}{\partial w_0} + \frac{\partial f}{\partial g_1}\cdot\frac{\partial g_1}{\partial w_0} = 10g_0 + 20g_1 - 10$$

$\boxed{\frac{\partial f}{\partial g_0} = 2(g_0 + 2g_1 - 1)}$ $\boxed{\frac{\partial g_0}{\partial w_0} = 1}$ $\boxed{\frac{\partial f}{\partial g_1} = 2(g_0 + 2g_1 - 1)\cdot 2}$ $\boxed{\frac{\partial g_1}{\partial w_0} = 2}$

내부의 함수가 3개 이상이라면,

$$\frac{\partial}{\partial w_0} f(g_0(w_0, w_1), g_1(w_0, w_1), \ldots, g_M(w_0, w_1)) = \sum_{m=0}^{M}\frac{\partial f}{\partial g_m}\cdot\frac{d g_m}{d w_0} \qquad \text{(식 4–62)}$$

5.5 합과 미분의 교환

- 머신러닝에서는 계산 과정에서 합의 기호로 표현된 함수를 미분할 경우가 많음.
- 미분과 합의 기호의 교환

미분과 합의 기호는 순서를 바꿀 수 있다

$$\frac{\partial}{\partial w}\sum_{n} f_n(w) = \sum_{n} \frac{\partial}{\partial w} f_n(w) \qquad \text{(식 4-66)}$$

예) $J = \frac{1}{N}\sum_{n=0}^{N-1}(w_0 x_n + w_1 - t_n)^2$ 의 경우

미분 기호를 합의 기호 앞으로 이동할 수 있다

$$\frac{\partial J}{\partial w_0} = \frac{\partial}{\partial w_0}\frac{1}{N}\sum_{n=0}^{N-1}(w_0 x_n + w_1 - t_n)^2 = \frac{1}{N}\sum_{n=0}^{N-1}\boxed{\frac{\partial}{\partial w_0}}(w_0 x_n + w_1 - t_n)^2$$

$$= \frac{2}{N}\sum_{n=0}^{N-1}(w_0 x_n + w_1 - t_n)\, x_n$$

6.1 행렬이란

- 숫자를 가로 세로로 표처럼 늘어 놓은 것을 행렬로 부름.
- 행렬(matrix)은 수나 기호, 수식 등을 네모꼴로 배열한 것으로, 괄호로 묶어 표시함.

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

행렬

$$\begin{array}{cccc} & \text{0열째} & \text{1열째} & \text{2열째} & \\ \mathbf{A} = & \left[\begin{matrix} 1 \\ 4 \end{matrix}\right. & \begin{matrix} 2 \\ 5 \end{matrix} & \left.\begin{matrix} 3 \\ 6 \end{matrix}\right] & \begin{matrix} \text{0행째} \\ \text{1행째} \end{matrix} \end{array}$$

2행 3열의 행렬
2×3 행렬

$[\mathbf{A}]_{i,j}$ 로 쓰면
$\mathbf{A}$의 i행 j열의 요소

예) $[\mathbf{A}]_{0,1} = 2$

(이 책에서는 파이썬의 배열에 맞게 행렬도 0행 0열에서 시작합니다)

행렬의 요소를 변수로 쓸 때, 인덱스는 '행' '열'의 순서

$$\mathbf{A} = \begin{bmatrix} a_{0,0} & a_{0,1} & a_{0,2} \\ a_{1,0} & a_{1,1} & a_{1,2} \end{bmatrix}$$

$a_{i,j}$ i행 j열의 요소

6.2 행렬의 덧셈과 뺄셈

- 행렬의 덧셈·뺄셈 규칙
- 행렬의 덧셈과 뺄셈은 같은 크기의 행렬끼리가 아니면 할 수 없음.

행렬의 덧셈 · 뺄셈은 요소마다 실시

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \end{bmatrix} + \begin{bmatrix} b_{00} & b_{01} & b_{02} \\ b_{10} & b_{11} & b_{12} \end{bmatrix} = \begin{bmatrix} a_{00}+b_{00} & a_{01}+b_{01} & a_{02}+b_{02} \\ a_{10}+b_{10} & a_{11}+b_{11} & a_{12}+b_{12} \end{bmatrix}$$

두 행렬은 동일한 크기일때만 계산 가능

예)

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} + \begin{bmatrix} 7 & 8 & 9 \\ 10 & 11 & 12 \end{bmatrix} = \begin{bmatrix} 1+7 & 2+8 & 3+9 \\ 4+10 & 5+11 & 6+12 \end{bmatrix} = \begin{bmatrix} 8 & 10 & 12 \\ 14 & 16 & 18 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} - \begin{bmatrix} 7 & 8 & 9 \\ 10 & 11 & 12 \end{bmatrix} = \begin{bmatrix} 1-7 & 2-8 & 3-9 \\ 4-10 & 5-11 & 6-12 \end{bmatrix} = \begin{bmatrix} -6 & -6 & -6 \\ -6 & -6 & -6 \end{bmatrix}$$

- 파이썬에서 A + B, A - B는 다음과 같이 계산함.

In

```
# 리스트 4-3-(4)
print(A + B)
print(A - B)
```

Out

```
[[ 8 10 12]
 [14 16 18]]
[[-6 -6 -6]
 [-6 -6 -6]]
```

6.3 스칼라 배

- 행렬에 스칼라 값을 곱할 때는 모든 요소에 대해 곱함.

$$2\mathbf{A} = 2\times\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\end{bmatrix} = \begin{bmatrix}2\times1 & 2\times2 & 2\times3\\2\times4 & 2\times5 & 2\times6\end{bmatrix} = \begin{bmatrix}2 & 4 & 6\\8 & 10 & 12\end{bmatrix}$$

행렬의 스칼라 배는 요소 모두에 곱한다

$$c\begin{bmatrix}a_{00} & a_{01} & a_{02}\\a_{10} & a_{11} & a_{12}\end{bmatrix} = \begin{bmatrix}ca_{00} & ca_{01} & ca_{02}\\ca_{10} & ca_{11} & ca_{12}\end{bmatrix}$$

예) $2\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\end{bmatrix} = \begin{bmatrix}2\times1 & 2\times2 & 2\times3\\2\times4 & 2\times5 & 2\times6\end{bmatrix} = \begin{bmatrix}2 & 4 & 6\\8 & 10 & 12\end{bmatrix}$

In

```python
# 리스트 4-3-(5)
A=np.array([[1, 2, 3],[4, 5, 6]])
print(2*A)
```

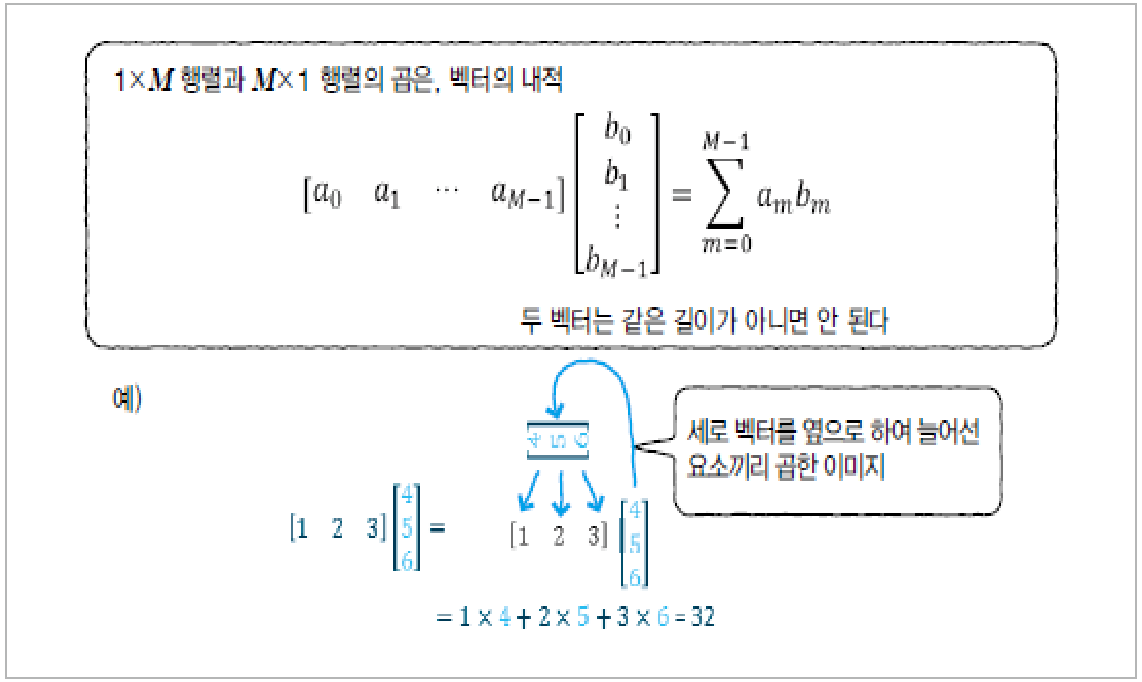
Out

```
[[2 4 6]
 [8 10 12]]
```

[파이썬]

6.4 행렬의 곱

- 1×M 행렬과 M×1 행렬의 곱

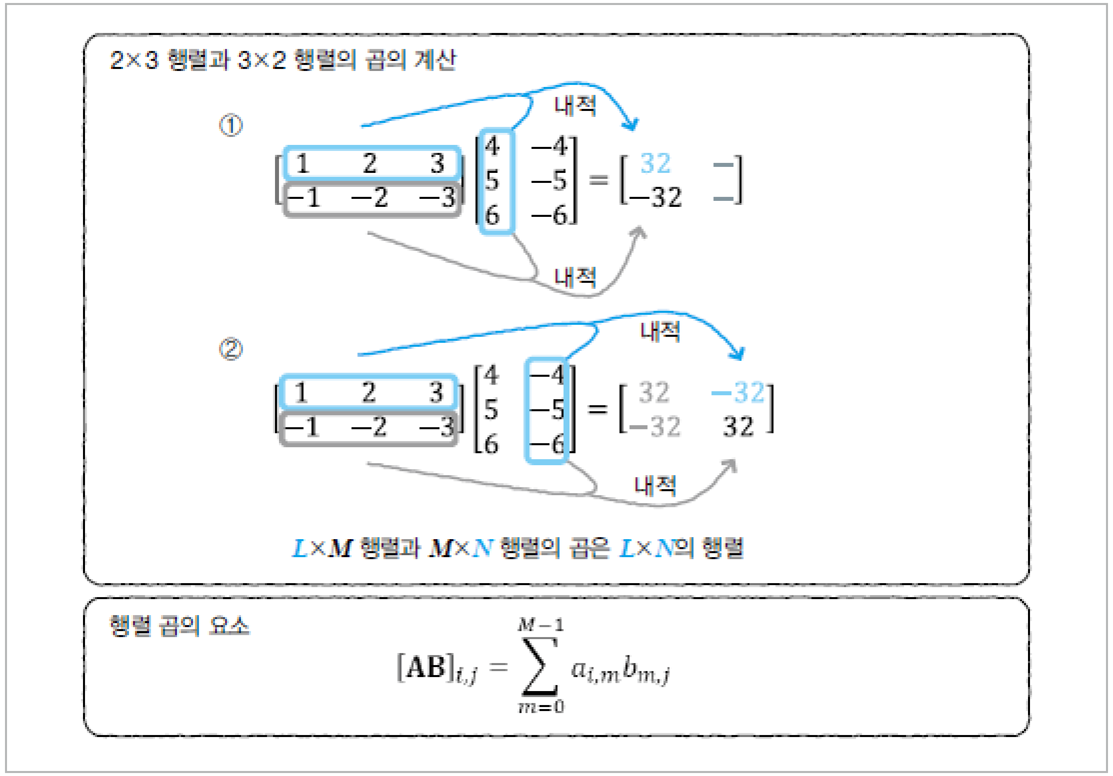


In
```
# 리스트 4-3-(6)
A=np.array([1, 2, 3])
B=np.array([4, 5, 6])
print(A.dot(B))
```

Out
```
32
```

[파이썬]

- L×M 행렬과 M×N 행렬의 곱



In

```
# 리스트 4-3-(9)
A = np.array([[1, 2, 3], [-1, -2, -3]])
B = np.array([[4, -4], [5, -5], [6, -6]])
print(A.dot(B))
```

Out

```
[[ 32 -32]
 [-32  32]]
```

[파이썬]

6.5 단위 행렬

- 정방 행렬의 경우 대각선 성분이 1이고, 그 이외에는 0인 특별한 행렬을 단위 행렬이라고 함.
- 선형대수학에서 행렬의 크기가 n인 단위행렬(identity matrix)은 주 대각선이 전부 1이고 나머지 원소는 0을 값으로 갖는 n×n 정사각행렬임.

단위 행렬은 대각 성분이 1인 행렬
다른 행렬을 곱해도 변하지 않는다

$$I=\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}\qquad I=\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}$$

3×3 4×4

예)

$$\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix}\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}=\begin{bmatrix}1+0+0&0+2+0&0+0+3\\4+0+0&0+5+0&0+0+6\\7+0+0&0+8+0&0+0+9\end{bmatrix}=\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix}\quad(\text{식 4-85})$$

In

```python
# 리스트 4-3-(10)
print(np.identity(3))
```

Out

```
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

[파이썬]

In

```python
# 리스트 4-3-(11)
A = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
I = np.identity(3)
print(A.dot(I))
```

Out

```
[[1. 2. 3.]
 [4. 5. 6.]
 [7. 8. 9.]]
```

6.6 역행렬

- 정방 행렬의 경우 대각선 성분이 1이고, 그 이외에는 0인 특별한 행렬을 단위 행렬이라고 함.
- 선형대수학에서 행렬의 크기가 n인 단위행렬(identity matrix)은 주 대각선이 전부 1이고 나머지 원소는 0을 값으로 갖는 n×n 정사각행렬임.

역행렬이란, 곱하면 단위 행렬 I가 되는 행렬

$$\mathbf{AA}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$$

A의 역행렬을 A^{-1}로 표시합니다

2×2 행렬의 $\mathbf{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ 역행렬은

$$\mathbf{A}^{-1} = \frac{1}{ad-bc}\begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

역행렬은 정방 행렬에서만 정의할 수 있다.
ad – bc = 0이 되는 2×2 행렬은 역행렬을 가지지 않는다.

예)

$\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ 의 역행렬은

$$\mathbf{A}^{-1} = \frac{1}{1\cdot 4 \quad 2\cdot 3}\begin{bmatrix} 4 & -2 \\ -3 & 1 \end{bmatrix} = -\frac{1}{2}\begin{bmatrix} 4 & -2 \\ -3 & 1 \end{bmatrix} = \begin{bmatrix} -2 & 1 \\ 1.5 & -0.5 \end{bmatrix} \text{이며,} \qquad \text{(식 4–89)}$$

$$\mathbf{AA}^{-1} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \cdot -\frac{1}{2}\begin{bmatrix} 4 & -2 \\ -3 & 1 \end{bmatrix} = -\frac{1}{2}\begin{bmatrix} -2 & 0 \\ 0 & -2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \text{가 된다.} \qquad \text{(식 4–90)}$$

6.7 전치

- 세로 벡터를 가로 벡터로, 가로 벡터를 세로 벡터로 만드는 연산인 전치 **T는** 행렬로 확장할 수 있음.

전치 행렬은 행과 열을 교환 교환한 행렬입니다.

A의 전치 행렬을 A^T로 나타냅니다

예를 들어,

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$ 의 전치 행렬은 $$\mathbf{A}^T = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$$

공식

$$(\mathbf{AB})^T = \mathbf{B}^T\mathbf{A}^T \qquad \text{(식 4-94)}$$

$$(\mathbf{ABC})^T = \mathbf{C}^T(\mathbf{AB})^T = \mathbf{C}^T\mathbf{B}^T\mathbf{A}^T \qquad \text{(식 4-95)}$$

In
```
# 리스트 4-3-(13)
A = np.array([[1, 2, 3], [4, 5, 6]])
print(A)
print(A.T)
```

Out
```
[[1 2 3]
 [4 5 6]]
[[1 4]
 [2 5]
 [3 6]]
```

[파이썬]

6.8 행렬과 연립 방정식

- 행렬을 사용하면 많은 연립 방정식을 하나의 식으로 나타낼 수 있어 매우 편리함.

$$y = 2x$$

$$y = -x + 3$$

[두 개의 연립 방정식]

행렬 표기로 연립 방정식을 풀기

이 방정식을 풀자!

$$\begin{aligned} 2x \quad -y &= 0 \\ x \quad +y &= 3 \end{aligned} \qquad \text{(식 4-98)}$$

행렬 표기로 고치고,

$$\begin{bmatrix} 2 & -1 \\ 1 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ 3 \end{bmatrix}$$

$$\begin{bmatrix} 2x - y \\ x + y \end{bmatrix} = \begin{bmatrix} 0 \\ 3 \end{bmatrix}$$

양쪽에 $\begin{bmatrix} 2 & -1 \\ 1 & 1 \end{bmatrix}$의 역행렬을 곱한다.

$$\begin{bmatrix} 2 & -1 \\ 1 & 1 \end{bmatrix}^{-1}\begin{bmatrix} 2 & -1 \\ 1 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 2 & -1 \\ 1 & 1 \end{bmatrix}^{-1}\begin{bmatrix} 0 \\ 3 \end{bmatrix}$$

좌변은 역행렬을 곱하여
단위 행렬이 되고,

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 2 & -1 \\ 1 & 1 \end{bmatrix}^{-1}\begin{bmatrix} 0 \\ 3 \end{bmatrix}$$

$\begin{bmatrix} x \\ y \end{bmatrix}$만 남았다.

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 2 & -1 \\ 1 & 1 \end{bmatrix}^{-1}\begin{bmatrix} 0 \\ 3 \end{bmatrix}$$

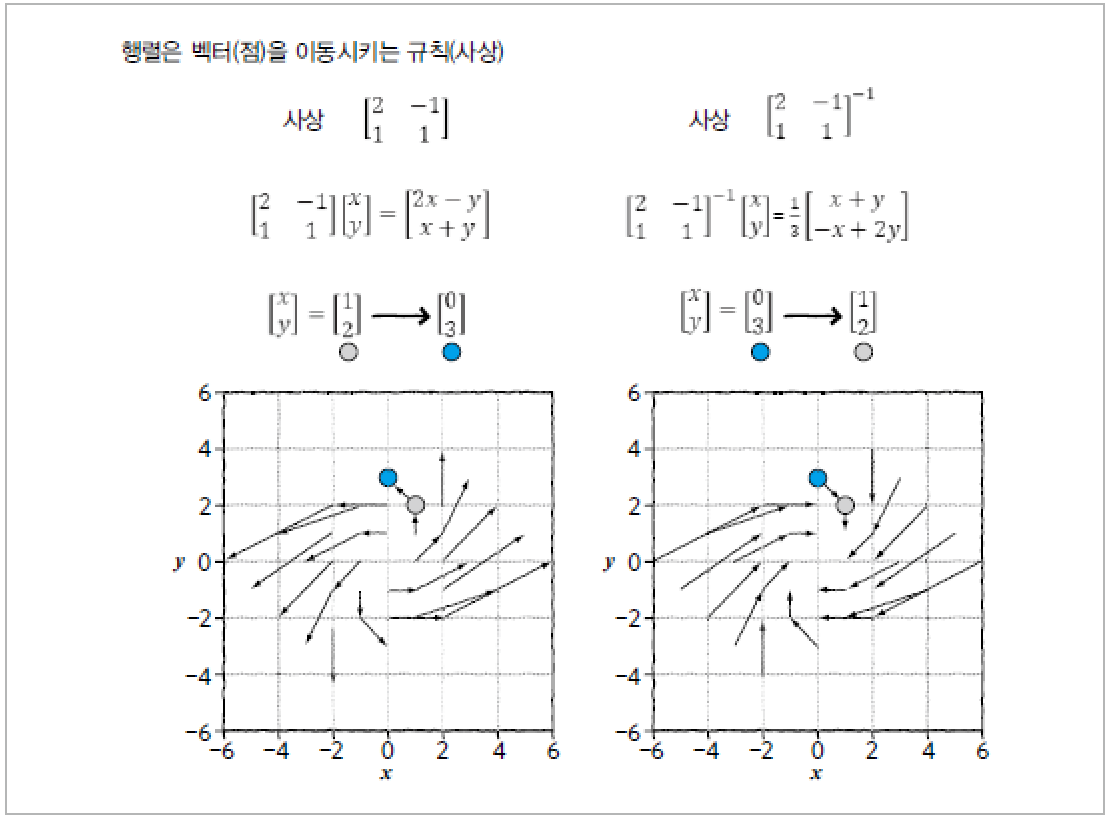
x=1, y=2를 얻었다!

우변은 역행렬의 공식을 사용해
계산하여 답을 얻는다.

$$\begin{bmatrix} x \\ y \end{bmatrix} = \frac{1}{3}\begin{bmatrix} 1 & 1 \\ -1 & 2 \end{bmatrix}\begin{bmatrix} 0 \\ 3 \end{bmatrix} = \frac{1}{3}\begin{bmatrix} 1 \times 0 + 1 \times 3 \\ (-1) \times 0 + 2 \times 3 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

6.9 행렬과 사상

- 행렬은 '벡터를 다른 벡터로 변환하는 규칙'으로 해석할 수 있음.
- 행렬은 '어떤 점을 다른 점으로 이동시키는 규칙'으로 파악할 수 있음.
- 그룹(벡터와 점)에서 그룹(벡터와 점)에 대응 관계를 제공하는 규칙을 사상이라고 함.



7.1 지수

- 지수는 '그 수를 여러 번 곱한다'는 곱셈의 횟수에서 출발한 개념
- 자연수뿐만 아니라 0에도, 음수에도, 실수에도 확장할 수 있음.
- 지수 함수(exponential function)란 거듭제곱의 지수를 변수로 하고, 정의역을 실수 전체로 정의하는 초월함수. 로그함수의 역함수임.

지수의 정의

$a > 0$、n 을 양의 정수라고 할 때

$a^0 = 1$ ... (1)

$a^{-n} = \frac{1}{a^n}$ ... (2)

$a^{1/n} = \sqrt[n]{a}$ ... (3)

지수의 공식

$a > 0$、$b > 0$、m、n 을 실수라고 할 때

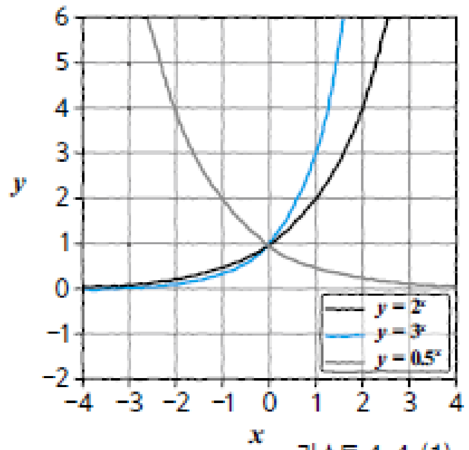
$a^n \times a^m = a^{n+m}$ ... (4)

$\frac{a^n}{a^m} = a^{n-m}$ ... (5)

$(a^n)^m = a^{n \times m}$ ... (6)

$(ab)^n = a^n b^n$ ... (7)

$y = a^x$



리스트 4-4-(1)

a를 밑으로 한 지수 함수

$y = a^x$

$a > 1$일 때 x가 증가하면 반드시 y도 증가한다.

단조 증가 함수가 되고, $0 < a < 1$의 경우는 단조 감소 함수가 된다.

밑 a가 클수록 그래프는 급격히 증가한다.

그래프는 항상 0보다 위에 있기 때문에, 지수 함수는 음수와 양수를 모두 양수로 옮기는 함수로 불린다.

7.2 로그

'로그 함수'는 지수 함수의 입력과 출력을 거꾸로 한 것임. 즉, 지수 함수의 역함수임.

$$y = a^y \qquad y = \log_a x$$

로그의 정의

a를 1이 아닌 양의 실수라고 할 때

$y = a^x$ 로 하면

$$\log_a y = x \quad \text{... (1)}$$

특별한 경우

$$\log_a a = 1 \quad \text{... (2)}$$

$$\log_a 1 = 0 \quad \text{... (3)}$$

로그의 공식

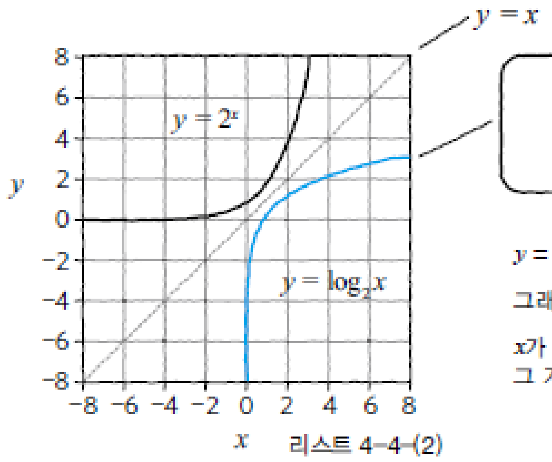
a, b를 1이 아닌 양의 실수라고 할 때

$$\log_a xy = \log_a x + \log_a y \quad \text{... (4)}$$

$$\log_a \frac{x}{y} = \log_a x - \log_a y \quad \text{... (5)}$$

$$\log_a x^y = y \log_a x \quad \text{... (6)}$$

$$\log_a x = \frac{\log_b x}{\log_b a} \quad \text{... (7)}$$



리스트 4-4-(2)

a를 밑으로 한 로그 함수

$$y = \log_a x$$

$y = a^x$의 그래프와 $y = x$ 선에서 대칭.

그래프는 $a > 0$의 범위에서만 정의된다.

x가 커지면 커질수록 그래프는 증가하지만, 그 기울기는 점점 완만해진다.

7.3 지수 함수의 미분

- 지수 $y = a^x$의 x 에 대한 미분은 다음과 같음.

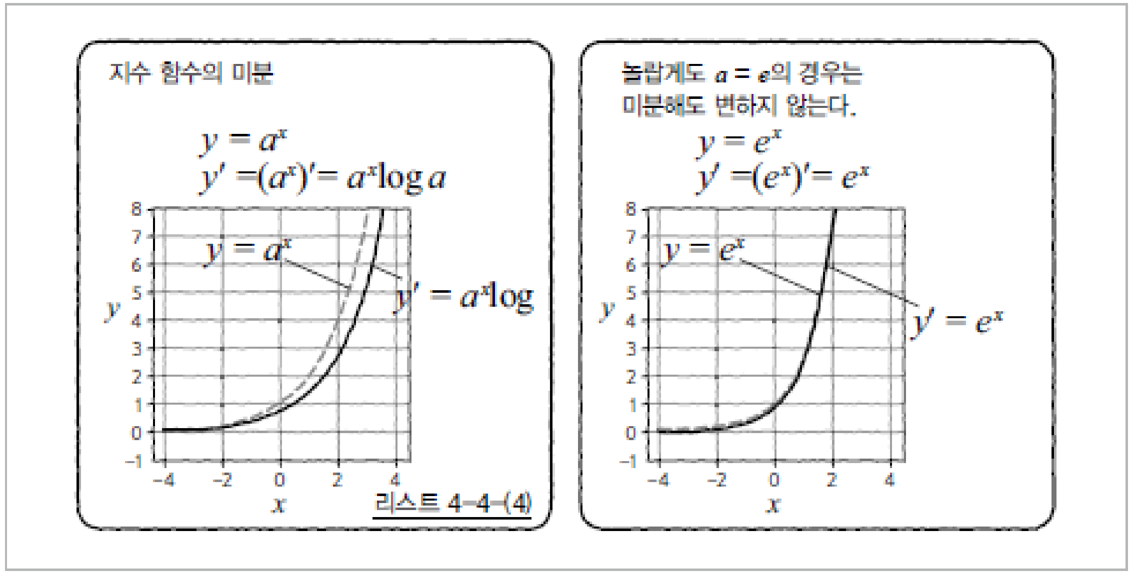
$$y' = (a^x)' = a^x \log a$$

In

```
# 리스트 4-4-(4)
x = np.linspace(-4, 4, 100)
a = 2 # 이 값을 여러 가지로 바꿔보자
y = a**x
dy = np.log(a) * y


plt.figure(figsize=(4, 4))
plt.plot(x, y, 'gray', linestyle='--', linewidth=3)
plt.plot(x, dy, color='black', linewidth=3)
plt.ylim(-1, 8)
plt.xlim(-4, 4)
plt.grid(True)
plt.show()
```

Out

```
# 실행 결과는 [그림 4-31]을 참조
```



7.4 로그 함수의 미분

- 로그 함수의 미분은 반비례의 식이 됩니다

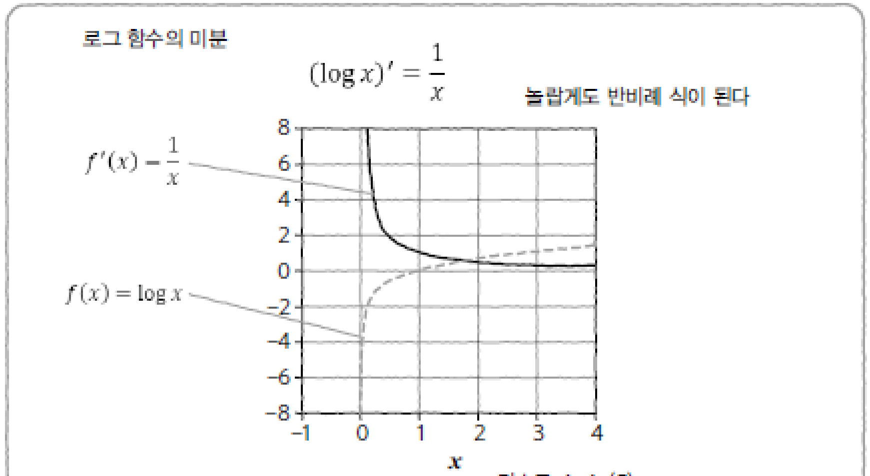
$$y'(x) = (\log x)' = \frac{1}{x}$$

In

```
# 리스트 4-4-(5)
x = np.linspace(0.0001, 4, 100) # 0 이하로 정의할 수 없는
y = np.log(x)
dy = 1 / x

plt.figure(figsize=(4, 4))
plt.plot(x, y, 'gray', linestyle='--', linewidth=3)
plt.plot(x, dy, color='black', linewidth=3)
plt.ylim(-8, 8)
plt.xlim(-1, 4)
plt.grid(True)
plt.show()
```

Out

```
# 실행 결과는 [그림 4-32]를 참조
```



리스트 4-4-(5)

7.5 시그모이드 함수

- 시그모이드 함수는 매끄러운 계단 같은 함수임.
- 음에서 양의 실수를 0에서 1까지의 사이로 변환하기 때문에 확률을 나타낼 때 자주 사용함.

$$y = \frac{1}{1 + e^{-x}}$$

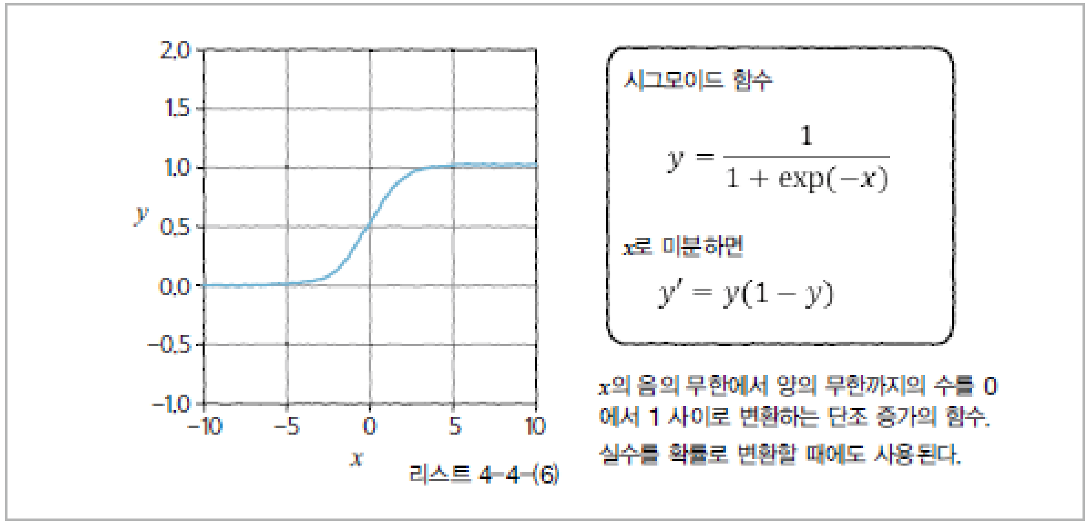
$$y = \frac{1}{1 + \exp(-x)}$$

In

```
# 리스트 4-4-(6)
x = np.linspace(-10, 10, 100)
y = 1 / (1 + np.exp(-x))

plt.figure(figsize=(4, 4))
plt.plot(x, y, 'black', linewidth=3)

plt.ylim(-1, 2)
plt.xlim(-10, 10)
plt.grid(True)
plt.show()
```

Out

```
# 실행 결과는 [그림 4-33]을 참조
```



7.6 소프트맥스 함수

- 소프트맥스 함수를 입력과 출력이 3차원이므로 그대로 그릴 수는 없음.
- 그래서 x2만 로 고정하여 다양한 x0과 x1을 입력했을 때의 y0과 y1를 플롯함.

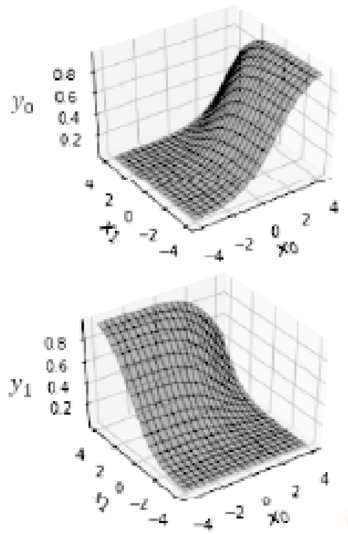
In

```
# 리스트 4-4-(8)

from mpl_toolkits.mplot3d import Axes3D

xn = 20
x0 = np.linspace(-4, 4, xn)
x1 = np.linspace(-4, 4, xn)

y = np.zeros((xn, xn, 3))
for i0 in range(xn):
    for i1 in range(xn):
        y[i1, i0, :] = softmax(x0[i0], x1[i1], 1)

xx0, xx1 = np.meshgrid(x0, x1)
plt.figure(figsize=(8, 3))
for i in range(2):
    ax = plt.subplot(1, 2, i + 1, projection='3d')
    ax.plot_surface(xx0, xx1, y[:, :, i],
                    rstride=1, cstride=1, alpha=0.3,
                    color='blue', edgecolor='black')
    ax.set_xlabel('$x_0$', fontsize=14)
    ax.set_ylabel('$x_1$', fontsize=14)
    ax.view_init(40, -125)

plt.show()
```

Out

```
# 실행 결과는 [그림 4-34]를 참조
```

$x_2 = 1$ 일 때의 3 변수 소프트맥스 함수의 출력



K 변수의 소프트맥스 함수

$$y_i = \frac{\exp(x_i)}{\sum_{j=0}^{K-1} \exp(x_j)}$$

x_i로 미분하면

$$\frac{\partial y_j}{\partial x_i} = y_j(I_{ij} - y_i)$$

I_{ij}는 $i = j$의 경우 1, $i \neq j$의 경우 0

복수 값 x_k의 대소 관계를 유지하면서, 확률로서의 값(각각의 값은 0에서 1로, 합은 1)으로 변환하는 함수.

리스트 4-4-(7, 8)

7.7 소프트맥스 함수와 시그모이드 함수

- 소프트맥스 함수와 시그모이드 함수는 닮아 있음.
- 두 변수의 소프트맥스 함수의 입력 x_0, x_1그 차이 $x = x_0 - x_1$낸 것이 시그모이드 함수임.
- 시그모이드 함수를 다변수로 확장한 것이 소프트맥스 함수라고 할 수 있음.

$$y = \frac{e^{x_0}}{e^{x_0} + e^{x_1}}$$

소프트맥스 함수

$$y = \frac{e^{x_0}e^{-x_0}}{e^{x_0}e^{-x_0} + e^{x_1}e^{-x_0}} = \frac{e^{x_0 - x_0}}{e^{x_0 - x_0} + e^{x_1 - x_0}} = \frac{1}{1 + e^{-(x_0 - x_1)}}$$

분모 분자에 e^{-x_0}을 곱하여 정리하면
$e^a e^{-b} = e^{a-b}$는 공식을 사용하여 내용 도출

$$y = \frac{1}{1 + e^{-x}}$$

$x = x_0 - x_1$ 로 두면 시그모이드 함수가 됨.

7.8 가우스 함수

- 가우스 함수(Gaussian function)는 다음과 같은 함수임

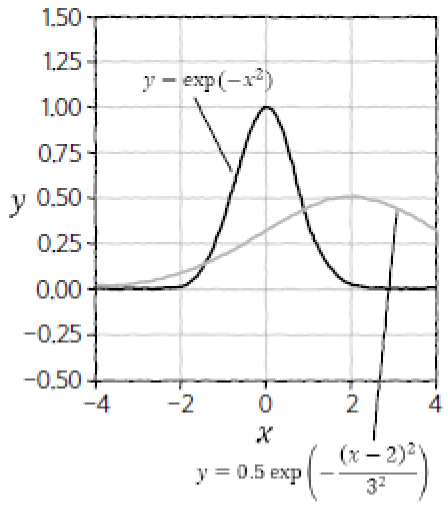
$$y = \exp(-x^2)$$

In

```
# 리스트 4-4-(9)
def gauss(mu, sigma, a):
    return a * np.exp(-(x - mu)**2 / sigma**2)


x = np.linspace(-4, 4, 100)
plt.figure(figsize=(4, 4))
plt.plot(x, gauss(0, 1, 1), 'black', linewidth=3)
plt.plot(x, gauss(2, 3, 0.5), 'gray', linewidth=3)
plt.ylim(-.5, 1.5)
plt.xlim(-4, 4)
plt.grid(True)
plt.show()
```

Out

```
# 실행 결과는 [그림 4-35]를 참조
```



리스트 4-4-(9)

가우스 함수

$$y = a \exp\left(-\frac{(x-\mu)^2}{\sigma^2}\right)$$

μ : 중심(평균)
σ : 확산(표준 편차)
a : 높이

x로 적분하여 1로 만들려면

$$a = \frac{1}{(2\pi\sigma^2)^{1/2}}$$ 로 한다.

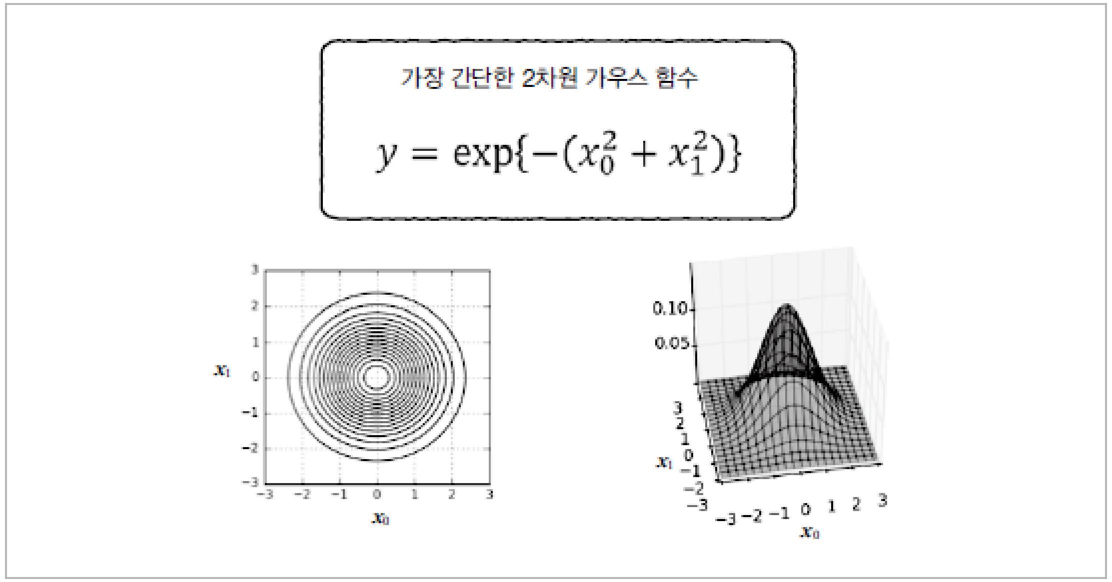
가우스 분포를 나타내는 함수이지만,
기저 함수로도 자주 사용된다.

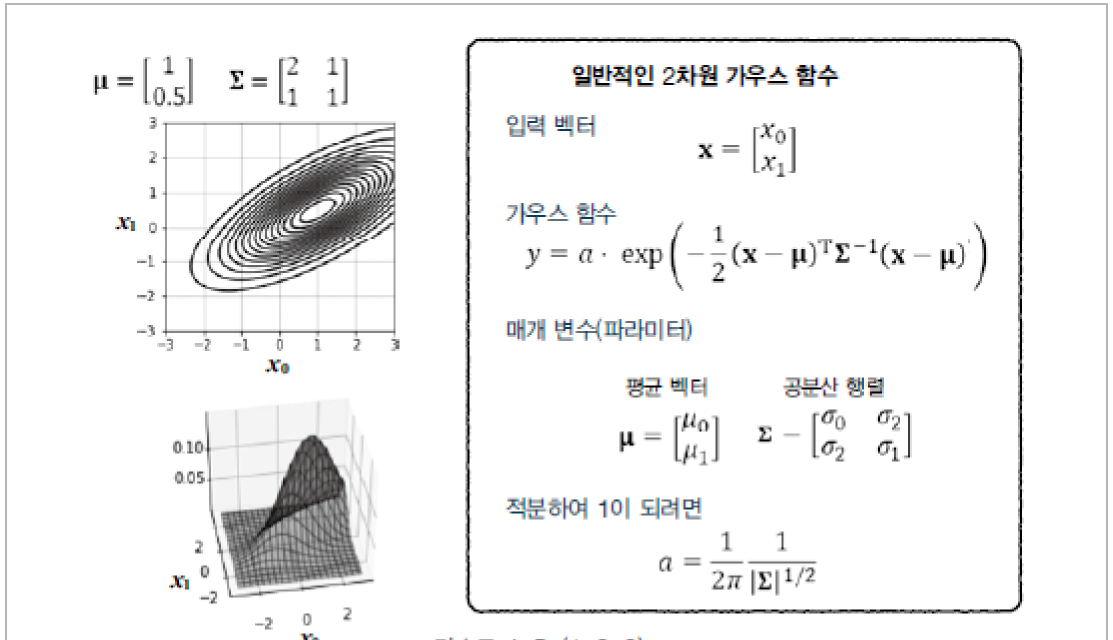
7.9 2차원 가우스 함수

- 가우스 함수를 2차원으로 확장할 수 있음.

- 입력을 2차원 벡터 라고 했을 때, 가우스 함수의 기본형은 다음과 같음.

$$y = \exp\left\{-(x_0^2 + x_1^2)\right\}$$





리스트 4-5-(1, 2, 3)