핸즈온 머신러닝 4장

PART. 선형 연립방정식과 역행렬

- 선형 예측모형

입력 데이터 벡터와 가중치 벡터의 내적으로 계산된 예측값이 실제 출력 데이터와 유사한 값을 출력하도록 하는 모형
⇒ 올바른 가중치 벡터를 구하기 위해서 연립방정식과 역행렬을 이용하여 선형 예측모형의 가중치 벡터를 구한다.

- 선형 연립방정식

복수의 미지수를 포함하는 복수의 선형 방정식을 **선형 연립방정식** 또는 연립일차방정식이라고 한다.

다음은 3개의 미지수와 3개의 선형 방정식을 가지는 선형 연립방정식의 한 예다.

$$\begin{array}{ccccccc} x_1 & + & x_2 & & & = & 2 \\ & & x_2 & + & x_3 & = & 2 \\ x_1 & + & x_2 & + & x_3 & = & 3 \end{array} \tag{2.4.1}$$

$x_1, x_2, \cdots, x_M$ 이라는 M 개의 미지수를 가지는 N개의 선형 연립방정식은 일반적으로 다음과 같은 형태가 된다. 이 식에서 a와 b는 방정식의 계수다.

$$\begin{array}{ccccccc} a_{11}x_1 & + & a_{12}x_2 & + \cdots + & a_{1M}x_M & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & + \cdots + & a_{2M}x_M & = & b_2 \\ \vdots & & \vdots & & \vdots & & \vdots \\ a_{N1}x_1 & + & a_{N2}x_2 & + \cdots + & a_{NM}x_M & = & b_N \end{array} \tag{2.4.2}$$

PART. 선형 연립방정식과 역행렬

행렬과 벡터의 곱셈을 이용하면 위 선형 연립방정식은 다음처럼 간단하게 쓸 수 있다.

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1M} \\ a_{21} & a_{22} & \cdots & a_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N1} & a_{N2} & \cdots & a_{NM} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_M \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix} \tag{2.4.3}$$

이 식에서

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1M} \\ a_{21} & a_{22} & \cdots & a_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N1} & a_{N2} & \cdots & a_{NM} \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_M \end{bmatrix}, \quad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix} \tag{2.4.4}$$

라고 하면 다음처럼 쓸 수 있다.

PART. 선형 연립방정식과 역행렬

$$Ax = b \tag{2.4.5}$$

A, x, b 는 각각 **계수행렬**(coefficient matrix), **미지수벡터**(unknown vector), **상수벡터**(constant vector)라고 부른다.

이 표현을 따르면 앞에서 예로 든 선형 연립방정식은 다음처럼 표현할 수 있다.

$$Ax = b \tag{2.4.6}$$

$$A = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \quad b = \begin{bmatrix} 2 \\ 2 \\ 3 \end{bmatrix} \tag{2.4.7}$$

만약 A, x, b가 행렬이 아닌 스칼라 실수라면 이 방정식은 나눗셈을 사용하여 다음처럼 쉽게 풀 수도 있을 것이다.

$$x = \frac{b}{A} \tag{2.4.8}$$

그러나 행렬에서는 나눗셈이 정의되지 않으므로 이 방법은 사용할 수 없다. 행렬에서는 나눗셈 대신 역행렬이라는 것을 사용한다.

PART. 선형 연립방정식과 역행렬

- ### 역행렬

정방 행렬 A에 대한 **역행렬(inverse matrix)** A^{-1}은 원래의 행렬 A와 다음 관계를 만족하는 정방 행렬을 말한다. I는 항등 행렬(identity matrix)이다.

$$A^{-1}A = AA^{-1} = I \tag{2.4.9}$$

역행렬은 항상 존재하는 것이 아니라 **행렬 A에 따라서는 존재하지 않을 수도 있다**. 역행렬이 존재하는 행렬을 **가역행렬(invertible matrix)**, 정칙행렬(regular matrix) 또는 비특이행렬(non-singular matrix)이라고 한다. 반대로 역행렬이 존재하지 않는 행렬을 비가역행렬(non-invertible matrix) 또는 **특이행렬(singular matrix)**, 퇴화행렬(degenerate matrix)이라고 한다.

PART. 선형 연립방정식과 역행렬

- 역행렬의 성질

역행렬은 다음 성질을 만족한다. 이 식에서 행렬 A, B, C는 모두 각각 역행렬 A^{-1}, B^{-1}, C^{-1}이 존재한다고 가정한다.

- 전치 행렬의 역행렬은 역행렬의 전치 행렬과 같다. 따라서 대칭 행렬의 역행렬도 대칭 행렬이다.

$$(A^T)^{-1} = (A^{-1})^T \tag{2.4.11}$$

- 두 개 이상의 정방 행렬의 곱은 같은 크기의 정방 행렬이 되는데 이러한 행렬의 곱의 역행렬은 다음 성질이 성립한다.

$$(AB)^{-1} = B^{-1}A^{-1} \tag{2.4.12}$$

$$(ABC)^{-1} = C^{-1}B^{-1}A^{-1} \tag{2.4.13}$$

PART. 선형 연립방정식과 역행렬

- 역행렬의 성질

역행렬의 계산

역행렬은 행렬식을 이용하여 다음처럼 계산할 수 있다. 증명은 생략한다.

$$A^{-1} = \frac{1}{\det(A)} C^T = \frac{1}{\det(A)} \begin{bmatrix} C_{1,1} & \cdots & C_{N,1} \\ \vdots & \ddots & \vdots \\ C_{1,N} & \cdots & C_{N,N} \end{bmatrix} \tag{2.4.14}$$

이 식에서 $C_{i,j}$는 A의 i, j번째 원소에 대해 정의한 코팩터(cofactor)다.

코팩터로 이루어진 행렬 C을 **여인수행렬(matrix of cofactors, 또는 cofactor matrix, comatrix)**이라고 한다. 또 여인수행렬의 전치행렬 C^T를 **어드조인트행렬(adjoint matrix, adjugate matrix, 수반행렬)**이라고 하며 $\text{adj}(A)$로 표기하기도 한다.

위 식에서 $\det(A) = 0$이면 역수가 존재하지 않으므로 **역행렬은 행렬식이 0이 아닌 경우에만 존재한다**는 것을 알 수 있다.

PART. 선형 연립방정식과 역행렬

- 역행렬에 대한 정리

역행렬에 대한 몇 가지 정리를 알아두면 도움이 된다.

셔먼-모리슨(Sherman-Morrison) 공식

정방행렬 A와 벡터 u, v에 대해 다음 공식이 성립한다.

$$(A + uv^T)^{-1} = A^{-1} - \frac{A^{-1}uv^TA^{-1}}{1 + v^TA^{-1}u} \tag{2.4.21}$$

우드베리(Woodbury) 공식

정방행렬 A와 이에 대응하는 적절한 크기의 행렬 U, V, C에 대해 다음 공식이 성립한다.

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1} \tag{2.4.22}$$

분할행렬의 역행렬

4개 블록(block)으로 분할된 행렬(partitioned matrix)의 역행렬은 각 분할행렬을 이용하여 계산할 수 있다.

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}^{-1} = \begin{bmatrix} A_{11}^{-1}(I + A_{12}FA_{11}^{-1}) & -A_{11}^{-1}A_{12}F \\ -FA_{21}A_{11}^{-1} & F \end{bmatrix} \tag{2.4.23}$$

PART. 선형 연립방정식과 역행렬

- **넘파이를 사용한 역행렬 계산**

넘파이의 linalg 서브패키지에는 역행렬을 구하는 `inv()` 라는 명령어가 존재한다. 앞에서 예로 든 선형 연립방정식의 행렬 A의 역행렬은 다음처럼 구할 수 있다.

In [1]:

```
import numpy as np

A = np.array([[1, 1, 0], [0, 1, 1], [1, 1, 1]])
A
```

```
array([[1, 1, 0],
       [0, 1, 1],
       [1, 1, 1]])
```

In [2]:

```
Ainv = np.linalg.inv(A)
Ainv
```

```
array([[ 0., -1.,  1.],
       [ 1.,  1., -1.],
       [-1.,  0.,  1.]])
```

PART. 선형 연립방정식과 역행렬

- **역행렬과 선형 연립방정식의 해**

선형 연립방정식에서 미지수의 수와 방정식의 수가 같다면 계수행렬 A는 정방행렬이 된다. 만약 행렬 A의 역행렬 A^{-1} 이 존재한다면 역행렬의 정의로부터 선형 연립방정식의 해는 다음처럼 구할 수 있다. 행렬과 벡터의 순서에 주의하라.

$$Ax = b \tag{2.4.26}$$

$$A^{-1}Ax = A^{-1}b \tag{2.4.27}$$

$$Ix = A^{-1}b \tag{2.4.28}$$

$$x = A^{-1}b \tag{2.4.29}$$

수학에서 **square matrix (정방행렬)**은 행렬인데, 같은 수의 행과 열을 가지는 행렬을 의미한다.
아래의 예는 4x4의 정방행렬에 속한다고 볼 수 있다.

$$\begin{bmatrix} 9 & 13 & 5 & 2 \\ 1 & 11 & 7 & 6 \\ 3 & 7 & 4 & 1 \\ 6 & 0 & 7 & 10 \end{bmatrix}$$

PART. 선형 연립방정식과 역행렬

- `lstsq()` 명령은 행렬 A와 b를 모두 인수로 받고 뒤에서 설명할 최소자승문제(least square problem)의 답 `x`, 잔차제곱합(residual sum of squares) `resid`, 랭크(rank) `rank`, 특잇값(singular value) `s`를 반환한다. 미지수와 방정식의 개수가 같고 행렬 A의 역행렬이 존재하면 최소자승문제의 답과 선형 연립방정식의 답이 같으므로 `lstsq()` 명령으로 선형 연립방정식을 풀 수도 있다. 최소자승문제, 랭크, 특잇값에 대해서는 뒤에서 자세히 설명할 것이다.

  다음 코드에서 `lstsq()` 명령으로 구한 답이 `inv()` 명령으로 구한 답과 같음을 알 수 있다.

In [6]:

```
x, resid, rank, s = np.linalg.lstsq(A, b)
x
```

```
array([[1.],
       [1.],
       [1.]])
```

PART. 선형 연립방정식과 역행렬

- 역행렬의 성질

In [11]:

```
A @ x
```

```
array([[1.98333333],
       [2.        ],
       [3.03333333],
       [4.08333333]])
```

`lstsq()` 명령으로 바로 구해도 같은 값이 나온다.

In [12]:

```
x, resid, rank, s = np.linalg.lstsq(A, b)
x
```

```
array([[1.03333333],
       [0.95      ],
       [1.05      ]])
```

위 코드에서 `resid`는 잔차벡터의 $e = Ax - b$의 제곱합, 즉 놈의 제곱이다.

In [13]:

```
resid, np.linalg.norm(A @ x - b) ** 2
```

```
(array([0.00166667]), 0.0016666666666666655)
```

PART. 선형 연립방정식과 역행렬

- **선형 연립방정식과 선형 예측모형**

선형 예측모형의 가중치벡터를 구하는 문제는 선형 연립방정식을 푸는 것과 같다. 예를 들어 N 개의 입력차원을 가지는 특징벡터 N개를 입력 데이터로 이용하고 이 입력에 대응하는 목푯값벡터를 출력하는 선형 예측모형을 생각하자.

$$\begin{array}{ccccccc} x_{11}w_1 & + & x_{12}w_2 & +\cdots+ & x_{1N}w_N & = & y_1 \\ x_{21}w_1 & + & x_{22}w_2 & +\cdots+ & x_{2N}w_N & = & y_2 \\ \vdots & & \vdots & & \vdots & & \vdots \\ x_{N1}w_1 & + & x_{N2}w_2 & +\cdots+ & x_{NN}w_N & = & y_N \end{array} \tag{2.4.30}$$

즉,

$$Xw = y \tag{2.4.31}$$

이 예측 모형의 가중치벡터 w를 찾는 것은 계수행렬이 X, 미지수벡터가 w, 상수벡터가 y인 선형 연립방정식의 답을 찾는 것과 같다. 그리고 만약 계수행렬, 여기에서는 특징행렬 X의 역행렬 X^{-1}이 존재하면 다음처럼 가중치벡터를 구할 수 있다.

$$w = X^{-1}y \tag{2.4.32}$$

PART. 선형 연립방정식과 역행렬

- 이때는 x_2가 어떤 값이 되더라도 $x_1 = x_3 = 2 - x_2$만 만족하면 되므로 무한히 많은 해가 존재한다. 예들 들어 다음 x 벡터는 모두 위 선형 연립방정식의 해다.

$$x = \begin{bmatrix} 2 \\ 0 \\ 2 \end{bmatrix}, \quad x = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \quad x = \begin{bmatrix} 0 \\ 2 \\ 0 \end{bmatrix}, \quad \cdots \tag{2.4.34}$$

3번의 경우, 즉 방정식의 수가 미지수의 수보다 많을 때는 2번과 반대로 모든 조건을 만족하는 해가 하나도 존재할 수 없을 수 있다. 예를 들어 다음 선형 연립방정식을 생각해보자. 미지수는 3개지만 방정식은 4개다.

$$\begin{array}{ccccccc} x_1 & + & x_2 & & & = & 2 \\ & & x_2 & + & x_3 & = & 2 \\ x_1 & + & x_2 & + & x_3 & = & 3 \\ x_1 & + & x_2 & + & 2x_3 & = & 5 \end{array} \tag{2.4.35}$$

위의 3개 방정식을 동시에 만족하는 해는 $x_1 = x_2 = x_3 = 1$인데 이 값은 4번째 방정식을 만족하지 못한다.

$$x_1 + x_2 + 2x_3 = 4 \tag{2.4.36}$$

따라서 4개의 방정식을 모두 만족하는 해는 존재하지 않는다.

PART. 선형 연립방정식과 역행렬

- 이때는 x_2가 어떤 값이 되더라도 $x_1 = x_3 = 2 - x_2$만 만족하면 되므로 무한히 많은 해가 존재한다. 예를 들어 다음 x 벡터는 모두 위 선형 연립방정식의 해다.

$$x = \begin{bmatrix} 2 \\ 0 \\ \vdots \end{bmatrix}, \quad x = \begin{bmatrix} 1 \\ 1 \\ \vdots \end{bmatrix}, \quad x = \begin{bmatrix} 0 \\ 2 \\ \vdots \end{bmatrix}, \quad \cdots \tag{2.4.34}$$

선형 예측모형을 구하는 문제는 계수행렬이 특징행렬 XX, 미지수벡터가 가중치벡터 ww인 선형 연립방정식 문제이다.
그런데 보통 데이터의 수는 입력차원보다 큰 경우가 많다.
예를 들어 면적, 층수, 한강이 보이는지의 여부로 집값을 결정하는 모형을 만들기 위해서 딱 3가구의 아파트 가격만 조사하는 경우는 없을 것이다. 보통은 10 가구 혹은 100 가구의 아파트 가격을 수집하여 이용하는 것이 일반적이다.
다시 말해 선형 예측모형을 구할 때는 3번과 같은 경우가 많다는 것을 알 수 있다.
이때는 선형 연립방정식의 해가 존재하지 않으므로 선형 연립방정식을 푸는 방식으로는
선형 예측모형의 가중치벡터를 구할 수 없다.

$$\begin{matrix} & & x_2 & + & x_3 & = & \cdots \\ x_1 & + & x_2 & + & x_3 & = & 3 \\ x_1 & + & x_2 & + & 2x_3 & = & 5 \end{matrix} \tag{2.4.35}$$

위의 3개 방정식을 동시에 만족하는 해는 $x_1 = x_2 = x_3 = 1$인데 이 값은 4번째 방정식을 만족하지 못한다.

$$x_1 + x_2 + 2x_3 = 4 \tag{2.4.36}$$

따라서 4개의 방정식을 모두 만족하는 해는 존재하지 않는다.

PART. 선형 연립방정식과 역행렬

- 최소자승문제

이렇게 선형 연립방정식의 해가 존재하지 않는다면 선형 예측모형은 어떻게 구할까? 모형을 구하는 것을 포기해야 하는가? 그럴 필요는 없다. 이 문제에 대한 힌트를 얻기 위해 다음과 같은 선형 연립방정식을 생각해보자.

$$\begin{array}{ccccccc} x_1 & + & x_2 & & & = & 2 \\ & & x_2 & + & x_3 & = & 2 \\ x_1 & + & x_2 & + & x_3 & = & 3 \\ x_1 & + & x_2 & + & 2x_3 & = & 4.1 \end{array} \tag{2.4.37}$$

위에서 보았듯이 이 선형 연립방정식의 해는 존재하지 않는다.

하지만 꼭 양변이 **정확하게 똑같지 않아도 된다면** 어떨까? $x_1 = x_2 = x_3 = 1$를 위 방정식에 대입하면 결과는 다음과 같다.

$$\begin{array}{ccccccccc} x_1 & + & x_2 & & & = & 2 & & \\ & & x_2 & + & x_3 & = & 2 & & \\ x_1 & + & x_2 & + & x_3 & = & 3 & & \\ x_1 & + & x_2 & + & 2x_3 & = & 4 & \approx & 4.1 \end{array} \tag{2.4.38}$$

선형 예측모형에서 좌변을 예측값, 우변을 목푯값이라고 생각한다면 100% 정확히 예측하지는 못했지만 상당히 비슷하게 예측한 값이라고 할 수 있다.

따라서 미지수의 개수보다 방정식의 개수가 많아서 선형 연립방정식으로 풀수 없는 문제는 좌변과 우변의 차이를 최소화하는 문제로 바꾸어 풀 수 있다. 앞서 예측값과 목푯값의 차이를 잔차(residual)라고 한다고 했다.

$$e = Ax - b \tag{2.4.39}$$

핸즈온 머신러닝 4장

PART. 선형 연립방정식과 역행렬

- 잔차는 벡터이므로 최소자승문제에서는 벡터의 크기 중에서 **벡터의 놈(norm)을 최소화**하는 문제를 푼다. 앞 절에서 놈을 최소화하는 것은 놈의 제곱을 최소화하는 것과 같다고 했다. 여기에서는 잔차제곱합이 놈의 제곱이 된다.

$$e^Te = \|e\|^2 = (Ax-b)^T(Ax-b) \tag{2.4.40}$$

이 값을 최소화하는 x값은 수식으로 다음처럼 표현한다.

$$x = \arg\min_x e^Te = \arg\min_x \, (Ax-b)^T(Ax-b) \tag{2.4.41}$$

위 식에서 $\arg\min_x f(x)$는 함수 $f(x)$를 가장 작게 만드는 x값을 의미한다. 이러한 문제를 **최소자승문제(least square problem)**라고 한다.

A^TA가 항상 정방 행렬이 된다는 점을 이용하여 다음과 같이 최소 자승 문제의 답이 어떤 형태가 되는지 살펴보자. 여기에서는 답의 형태만 살펴보고 엄밀한 증명은 하지 않을 것이다.

$$Ax \approx b \tag{2.4.42}$$

이 식의 양변에 A^T를 곱하면 각각 A^TAx와 A^Tb 가 된다. 이 두 개의 벡터의 값이 같다고 일단 가정하자.

$$A^TAx = A^Tb \tag{2.4.43}$$

만약 정방 행렬 A^TA의 역행렬 $(A^TA)^{-1}$이 존재한다면

$$(A^TA)^{-1}(A^TA)x = (A^TA)^{-1}A^Tb \tag{2.4.44}$$

이 식을 정리하면 다음과 같다.

$$x = ((A^TA)^{-1}A^T)b \tag{2.4.45}$$

- 유사역행렬(의사역행렬)

여기에서 행렬 $(A^TA)^{-1}A^T$ 를 행렬 A의 **의사역행렬(pseudo inverse)**이라고 하며 다음처럼 A^+ 로 표기한다.

$$A^+ = (A^TA)^{-1}A^T \tag{2.4.46}$$

$$x = A^+b \tag{2.4.47}$$

핸즈온 머신러닝 4장

PART. 선형 연립방정식과 역행렬

여기에서 행렬 $(A^TA)^{-1}A^T$ 를 행렬 A의 **의사역행렬(pseudo inverse)**이라고 하며 다음처럼 A^+ 로 표기한다.

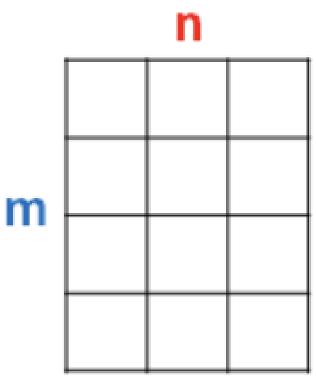
$$A^+ = (A^TA)^{-1}A^T \tag{2.4.46}$$

$$x = A^+b \tag{2.4.47}$$

- 유사역행렬(의사역행렬)

직사각형의 matrix에 자신의 transpose를 곱해서(A*A' or A' * A) 정사각형 matrix로 만든 다음 역행렬을 구하고 다시 transpose matrix를 곱해 원래의 직사각형 행렬로 돌아오면 됩니다.

세로가 긴 행렬과 가로가 긴 행렬을 처리하는 방법이 조금 다릅니다.



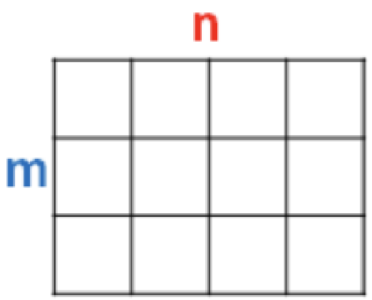
Size: m x n

m>n

Rank = n

Left Inverse

$(A^TA)^{-1}A^TA$



Size: m x n

m<n

Rank = m

Right Inverse

$AA^T(AA^T)^{-1}$

핸즈온 머신러닝 4장

PART. 선형 연립방정식과 역행렬

행의 길이가 긴 경우는 Left Inverse와 같이 처리하면 됩니다. 열의 길이가 긴 경우는 오른쪽과 같이 Right Inverse처럼 처리하면 됩니다.
Left Inverse와 Right Inverse로 구분하여 처리해야 하는 이유는 **Rank** 때문입니다.
matrix A의 크기는 m * n 입니다. A'의 크기는 n * m 입니다. 그리고 크기가 n * n인 square matrix의 역행렬의 크기는 역시 n * n 입니다.

Left Inverse의 경우 (A' * A)^-1 * A'*A의 Rank를 구해보면 다음과 같습니다.
(n *m)(m * n)(n * m)(m * n) => n * n이 됩니다. 따라서 rank는 n이 나옵니다.

Right Inverse의 경우 (m*n)(n*m)(m*n)(n*m) => m*m으로 rank가 m이 나옵니다.
만약 Left와 Right의 구분을 하기 싫다면 SVD(Singular Value Decomposition)을 이용하면 됩니다.

PART. 선형 연립방정식과 역행렬

- 특이값 분해(singular value decomposition: SVD)

  정방행렬은 고유분해로 고윳값과 고유벡터를 찾을 수 있었다. 정방행렬이 아닌 행렬은 고유분해가 불가능하다. 하지만 대신 고유분해와 비슷한 특이분해를 할 수 있다.

- 특이값과 특이벡터

  N×M크기의 행렬 A를 다음과 같은 3개의 행렬의 곱으로 나타내는 것을
  특이분해(singular-decomposition) 또는 **특잇값 분해(singular value decomposition)**라고 한다.

$$A = U\Sigma V^T \tag{3.4.1}$$

PART. 선형 연립방정식과 역행렬

- 특이값 분해(singular value decomposition: SVD)

여기에서 U, Σ, V는 다음 조건을 만족해야 한다.

- 대각성분이 양수인 대각행렬이어야 한다. 큰 수부터 작은 수 순서로 배열한다.

$$\Sigma \in \mathbf{R}^{N \times M} \tag{3.4.2}$$

- U는 N차원 정방행렬로 모든 열벡터가 단위벡터이고 서로 직교해야 한다.

$$U \in \mathbf{R}^{N \times N} \tag{3.4.3}$$

- V는 M차원 정방행렬로 모든 열벡터가 단위벡터이고 서로 직교해야 한다.

$$V \in \mathbf{R}^{M \times M} \tag{3.4.4}$$

위 조건을 만족하는 행렬 Σ의 대각성분들을 **특잇값(singular value)**, 행렬 U의 열벡터들을 **왼쪽 특이벡터**(left singular vector), 행렬 V의 행벡터들을 **오른쪽 특이벡터**(right singular vector)라고 부른다.

직교행렬의 역행렬은 직교행렬 자신의 전치행렬(transpose matrix)와 같다