# 2.3 행렬의 성질

여러숫자여기 때문

행렬은 여러 개의 숫자로 이루어져 있으므로 실수처럼 부호나 크기를 정의하기 어렵다. 하지만 부호/크기와 유사한 개념은 정의할 수 있다. 여기에서는 이러한 개념을 살펴본다.

## 정부호와 준정부호

영 벡터가 아닌 모든 벡터 $x$에 대해 다음 부등식이 성립하면 행렬 $A$가 **양의 정부호(positive definite)**라고 한다.

$$x^T A x > 0 \tag{2.3.1}$$

만약 이 식이 등호를 포함한다면 **양의 준정부호(positive semi-definite)**라고 한다.

$$x^T A x \geq 0 \tag{2.3.2}$$

위 방법에 따르면 모든 행렬에 대해 양의 정부호와 준정부호를 정의할 수 있지만 보통 대칭행렬에 대해서만 정의한다.

예를 들어 항등행렬(identity matrix) $I$는 양의 정부호다. 다음 식에서 벡터 $x$가 영벡터(zeros-vector)가 아니라는 점에 주의한다.

양의 정부호.

$$x^T I x = \begin{bmatrix} x_1 & x_2 & \cdots & x_N \end{bmatrix} \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} = x_1^2 + x_2^2 + \cdots + x_N^2 > 0 \tag{2.3.3}$$

다음과 같은 행렬도 양의 정부호다.

$$A = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix} \tag{2.3.4}$$

이는 다음처럼 증명할 수 있다.

모든 벡터 $x^T = [x_1\ x_2\ x_3]$에 대해

$$\begin{aligned} x^T A x &= \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \\ &= \begin{bmatrix} (2x_1 - x_2) & (-x_1 + 2x_2 - x_3) & (-x_2 + 2x_3) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \\ &= 2{x_1}^2 - 2x_1x_2 + 2{x_2}^2 - 2x_2x_3 + 2{x_3}^2 \\ &= {x_1}^2 + (x_1 - x_2)^2 + (x_2 - x_3)^2 + {x_3}^2 \end{aligned} \tag{2.3.5}$$

이 성립한다. 그리고 이 값은 제곱의 합으로 이루어졌으므로 $x$가 영벡터인 경우($x_1 = x_2 = x_3 = 0$)를 제외하고는 항상 0보다 크다.

$${x_1}^2 + (x_1 - x_2)^2 + (x_2 - x_3)^2 + {x_3}^2 > 0 \tag{2.3.6}$$

### 연습 문제 2.3.1

다음 행렬이 양의 정부호인지 양의 준정부호인지 혹은 어떤 것에도 해당되지 않는지 판단하라.

A이고, $x = [x_1, x_2]^T$ 라할때 (단 $x_1, x_2$는 0이아님)

$$\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \tag{2.3.7}$$

$x^T \cdot A \cdot x \geq 0$ 인지 파악한다.

행렬의 부호와 마찬가지로 행렬 크기를 정의하는 것도 어려운 일이다. 하나의 행렬에 대해 실수 하나를 대응시키는 개념의 **놈(norm)**, **대각합(trace)**, **행렬식(determinant)**이란 연산은 크기와 유사한 개념이다.

$$[x_1\ \ x_2] \cdot \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = [x_1\ \ x_2] \cdot \begin{bmatrix} x_1 + x_2 \\ x_1 + x_2 \end{bmatrix} = x_1(x_1 + x_2) + x_2(x_1 + x_2)$$
$$= (x_1 + x_2)(x_1 + x_2) = (x_1 + x_2)^2$$

$x_1, x_2$는 0이 아닌 실수 인데, $x_1 + x_2$의 제곱은 항상 양수 $\therefore$ $x^T \cdot A \cdot x > 0$ 이므로 정부호

## 행렬 놈

행렬의 **놈(norm)**은 행렬 $A$에 대해 다음 식으로 정의되는 숫자다. 보통 $\|A\|_p$로 표기한다. 이 식에서 $a_{ij}$는 행렬 $A$의 $i$번째 행, $j$번째 열의 원소다. 행렬의 놈에도 여러 정의가 있는데 여기에서는 요소별 행렬 놈(entrywise matrix norm)의 정의를 따른다.

$$\|A\|_p = \left( \sum_{i=1}^{N} \sum_{j=1}^{M} |a_{ij}|^p \right)^{1/p} \tag{2.3.8}$$

$p$는 보통 1, 2 또는 무한대($\infty$)가 사용되는데 이 중 $p = 2$인 경우가 가장 많이 쓰이므로 $p$값 표시가 없는 경우는 $p = 2$인 놈이라고 생각하면 된다. $p = 2$인 놈은 **프로베니우스 놈(Frobenius norm)**이라고 불리며 $\|A\|_F$이라고 표기하기도 한다.

$$\|A\| = \|A\|_2 = \|A\|_F = \sqrt{\sum_{i=1}^{N} \sum_{j=1}^{M} a_{ij}^2} \tag{2.3.9}$$

놈의 정의에서 **놈은 항상 0보다 같거나 크다**는 것을 알 수 있다.

Contents

놈은 모든 크기의 행렬에 대해서 정의할 수 있으므로 벡터에 대해서도 정의할 수 있다. 벡터의 놈에서 중요한 성질은 **벡터의 놈의 제곱이 벡터의 제곱합과 같다**는 것이다.

Norm 제곱 벡터 제곱합

$$\|x\|^2 = \sum_{i=1}^{N} x_i^2 = x^T x \tag{2.3.10}$$

놈은 0 또는 양수이므로 놈의 제곱이 가장 작을 때 놈도 가장 작아진다. 따라서 **놈을 최소화하는 것은 벡터의 제곱합을 최소화하는 것**과 같다.

넘파이에서는 linalg 서브패키지의 `norm()` 명령으로 행렬의 놈을 계산할 수 있다.

```
import numpy as np

A = (np.arange(9) - 4).reshape((3, 3))
A
```

```
array([[-4, -3, -2],
       [-1,  0,  1],
       [ 2,  3,  4]])
```

```
np.linalg.norm(A)
```

```
7.745966692414834
```

## 연습 문제 2.3.2

행렬 $A$, $(A \in \mathbf{R}^{N \times M})$의 놈의 제곱 $\|A\|^2$이 그 행렬을 이루는 행 벡터 $r_i$의 놈의 제곱의 합 또는 열 벡터 $c_i$의 놈의 제곱의 합과 같음을 증명하라.

$$\|A\|^2 = \sum_{i=1}^{N} \|r_i\|^2 = \sum_{j=1}^{M} \|c_j\|^2 \tag{2.3.11}$$

사실 위에서 쓴 놈의 공식은 공식적인 정의가 아니다. 정확한 놈의 정의는 다음 4가지 성질이 성립하는 행렬 연산을 말한다. 이러한 연산이 여러 개 존재하기 때문에 놈의 정의도 다양하다.

- 놈의 값은 0이상이다. 영행렬일 때만 놈의 값이 0이 된다.

$$\|A\| \geq 0 \tag{2.3.12}$$

- 행렬에 스칼라를 곱하면 놈의 값도 그 스칼라의 절대값을 곱한 것과 같다.

$$\|\alpha A\| = |\alpha| \|A\| \tag{2.3.13}$$

- 행렬의 합의 놈은 각 행렬의 놈의 합보다 작거나 같다.

$$\|A + B\| \leq \|A\| + \|B\| \tag{2.3.14}$$

- 정방행렬의 곱의 놈은 각 정방행렬의 놈의 곱보다 작거나 같다.

$$\|AB\| \leq \|A\| \|B\| \tag{2.3.15}$$

## 대각합 (Trace)

**대각합(trace)**은 정방행렬에 대해서만 정의되며 다음처럼 대각원소의 합으로 계산된다.

$$\text{tr}(A) = a_{11} + a_{22} + \cdots + a_{NN} = \sum_{i=1}^{N} a_{ii} \tag{2.3.16}$$

예를 들어 $N$차원 항등행렬(identity matrix)의 대각합은 $N$이다.

$\begin{bmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{bmatrix}_{N \times N}$

$$\text{tr}(I_N) = N \tag{2.3.17}$$

대각합을 구할 때는 절댓값을 취하거나 제곱을 하지 않기 때문에 대각합의 값은 놈과 달리 **음수가 될 수도 있다.**

대각합은 다음과 같은 성질이 있다. 아래의 식에서 $c$는 스칼라이고 $A, B, C$는 행렬이다.

대각합 성질

① $Tr(cA) = c\,Tr(A)$

② $Tr(A^T) = Tr(A)$

③ $Tr(A+B) = Tr(A) + tr(B)$

④ $Tr(AB) = tr(BA)$

⑤ $tr(ABC) = tr(ACB) \cdots$

- 스칼라를 곱하면 대각합은 스칼라와 원래의 대각합의 곱이다.

$$\text{tr}(cA) = c\,\text{tr}(A) \tag{2.3.18}$$

- 전치연산을 해도 대각합이 달라지지 않는다.

$$\text{tr}(A^T) = \text{tr}(A) \tag{2.3.19}$$

- 두 행렬의 합의 대각합은 두 행렬의 대각합의 합이다.

$$\text{tr}(A + B) = \text{tr}(A) + \text{tr}(B) \tag{2.3.20}$$

- 두 행렬의 곱의 대각합은 행렬의 순서를 바꾸어도 달라지지 않는다.

$$\text{tr}(AB) = \text{tr}(BA) \tag{2.3.21}$$

- 세 행렬의 곱의 대각합은 다음과 같이 순서를 순환시켜도 달라지지 않는다.

$$\mathrm{tr}(ABC) = \mathrm{tr}(BCA) = \mathrm{tr}(CAB) \tag{2.3.22}$$

마지막 식은 곱셈에서 순서가 바뀌어도 대각합이 같다는 것을 이용하여 증명할 수 있다.

$$\mathrm{tr}((AB)C) = \mathrm{tr}(C(AB)) = \mathrm{tr}((CA)B) = \mathrm{tr}(B(CA)) \tag{2.3.23}$$

특히 마지막 식은 **트레이스 트릭(trace trick)**이라고 하여 이차 형식(quadratic form)의 미분을 구하는 데 유용하게 사용된다. 이 두 식에서는 $A, B, C$가 각각 정방행렬일 필요는 없다. 최종적으로 대각합을 구하는 행렬만 정방행렬이기만 하면 된다.

이차 형식의 트레이스 트릭 공식은 다음과 같다.

$$x^TAx = \mathrm{tr}(x^TAx) = \mathrm{tr}(Axx^T) = \mathrm{tr}(xx^TA) \tag{2.3.24}$$

이 식은 원래의 트레이스 트릭 식의 $A, B, C$ 에 각각 $x^T, A, x$를 대입한 것이다. 이차 형식은 스칼라값이기 때문에 대각합을 취해도 원래의 값과 같다.

넘파이에서는 `trace()` 명령으로 대각합을 계산할 수 있다.

```
np.trace(np.eye(3))
```

$\begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \end{bmatrix}_{3\times3}$

```
3.0
```

## 연습 문제 2.3.3

$x, A$가 각각 크기가 2 인 벡터, 크기가 2x2 인 정방행렬일 때 이차 형식의 트레이스 트릭이 성립함을 보인다.

## 연습 문제 2.3.4

$N \times M$ 행렬 $X$에 대해 다음 식을 증명하라.

$$\mathrm{tr}(X(X^TX)^{-1}X^T) = M \tag{2.3.25}$$

위 식에서 $(X^TX)^{-1}$은 $X^TX$의 역행렬(inverse matrix)로 $X^TX$와 곱하면 항등행렬이 되는 행렬이다. 역행렬에 대해서는 나중에 자세히 공부한다.

$$(X^TX)^{-1}X^TX = X^TX(X^TX)^{-1} = I \tag{2.3.26}$$

## 연습 문제 2.3.5

행렬 $A$ ($A \in \mathbf{R}^{2\times2}$)의 놈의 제곱 $\|A\|^2$이 다음과 같음을 증명하라.

$$\|A\|^2 = \mathrm{tr}(A^TA) \tag{2.3.27}$$

# 행렬식

정방행렬 $A$의 **행렬식(determinant)**은 $\det(A)$, $\det A$, 또는 $|A|$라는 기호로 표기한다.

행렬식은 다음처럼 재귀적인 방법으로 정의된다.

우선 행렬 $A$가 $1 \times 1$ 즉 스칼라인 경우에는 행렬식이 자기 자신의 값이 된다.

$$\det([a]) = a \tag{2.3.28}$$

행렬 $A$가 스칼라가 아니면 **여인수 전개(cofactor expansion)**라고 불리는 다음 식을 이용하여 계산한다. 이 식에서 $a_{i,j}$는 $A$의 $i$행, $j$열 원소이고 $i_0$(또는 $j_0$)는 계산하는 사람이 임의로 선택한 행번호(또는 열번호)이다.

$$\det(A) = \sum_{i=1}^{N} \left\{ (-1)^{i+j_0} M_{i,j_0} \right\} a_{i,j_0} \tag{2.3.29}$$

또는

$$\det(A) = \sum_{j=1}^{N} \left\{ (-1)^{i_0+j} M_{i_0,j} \right\} a_{i_0,j} \tag{2.3.30}$$

위에서 '또는'이라고 한 이유는 두 식 중 아무거나 써도 같은 결과가 나오기 때문이다. 즉, 행렬에서 임의의 행 $i_0$ 하나를 선택하거나 임의의 열 $j_0$ 하나를 선택한 다음 이 값에 가중치 $(-1)^{i+j_0}M_{i,j_0}$ 또는 $(-1)^{i_0+j}M_{i_0,j}$를 곱하여 더한 것이다.

가중치로 사용된 $M_{i,j}$는 **마이너(minor, 소행렬식)**라고 하며 정방행렬 $A$ 에서 $i$행과 $j$열을 지워서 얻어진 (원래의 행렬보다 크기가 1만큼 작은) 행렬의 행렬식이다.

마이너값도 행렬식이므로 마찬가지로 위의 정의를 이용하여 계산한다. 이처럼 점점 크기가 작은 행렬의 행렬식을 구하다 보면 스칼라인 행렬이 나오게 되는데 행렬식의 값이 자기 자신이 된다. 따라서 행렬식을 구하는 방법은 **재귀적(recursive)**이다.

마이너에 $(-1)^{i+j}$를 곱한 값 $(-1)^{i+j}M_{i,j}$을 **여인수(cofactor)**라고 하며 $C_{i,j}$로 표기한다.

$$C_{i,j} = (-1)^{i+j}M_{i,j} \tag{2.3.31}$$

여인수를 사용하여 위 여인수 전개식을 다시 표현하면 다음과 같다.

$$\det(A) = \sum_{i=1}^{N} C_{i,j_0} a_{i,j_0} = \sum_{j=1}^{N} C_{i_0,j} a_{i_0,j} \tag{2.3.32}$$

예를 들어 다음과 같은 행렬을 생각해보자.

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \tag{2.3.33}$$

여기에서 임의의 행 또는 열을 선택한다. 행이든 열이든 상관없다.

만약 첫 번째 열을 선택했다고 하면($j_0 = 1$), 이 행렬의 행렬식은 다음처럼 계산한다.

$$\begin{aligned} \det(A) &= \left\{(-1)^{1+1} M_{1,1}\right\} a_{1,1} + \left\{(-1)^{2+1} M_{2,1}\right\} a_{2,1} + \left\{(-1)^{3+1} M_{3,1}\right\} a_{3,1} \\ &= M_{1,1} a_{1,1} - M_{2,1} a_{2,1} + M_{3,1} a_{3,1} \\ &= M_{1,1} - M_{2,1} \cdot 4 + M_{3,1} \cdot 7 \end{aligned} \tag{2.3.34}$$

이때 마이너값 $M_{1,1}, M_{2,1}, M_{3,1}$는 각각 다음과 같은 행렬의 행렬식이다.

$M_{1,1}$은 원래의 행렬에서 1번째 행과 1번째 열을 지워서 만들어진 행렬의 행렬식이다.

$$\begin{bmatrix} \backslash\text{cancel}1 & \backslash\text{cancel}2 & \backslash\text{cancel}3 \\ \backslash\text{cancel}4 & 5 & 6 \\ \backslash\text{cancel}7 & 8 & 9 \end{bmatrix} \rightarrow M_{1,1} = \det\left(\begin{bmatrix} 5 & 6 \\ 8 & 9 \end{bmatrix}\right) \tag{2.3.35}$$

$M_{2,1}$은 원래의 행렬에서 2번째 행과 1번째 열을 지워서 만들어진 행렬의 행렬식이다.

$$\begin{bmatrix} \backslash\text{cancel}1 & 2 & 3 \\ \backslash\text{cancel}4 & \backslash\text{cancel}5 & \backslash\text{cancel}6 \\ \backslash\text{cancel}7 & 8 & 9 \end{bmatrix} \rightarrow M_{2,1} = \det\left(\begin{bmatrix} 2 & 3 \\ 8 & 9 \end{bmatrix}\right) \tag{2.3.36}$$

$M_{3,1}$은 원래의 행렬에서 3번째 행과 1번째 열을 지워서 만들어진 행렬의 행렬식이다.

$$\begin{bmatrix} \backslash\text{cancel}1 & 2 & 3 \\ \backslash\text{cancel}4 & 5 & 6 \\ \backslash\text{cancel}7 & \backslash\text{cancel}8 & \backslash\text{cancel}9 \end{bmatrix} \rightarrow M_{3,1} = \det\left(\begin{bmatrix} 2 & 3 \\ 5 & 6 \end{bmatrix}\right) \tag{2.3.37}$$

이 마이너값 $M_{1,1}, M_{2,1}, M_{3,1}$은 마찬가지로 여인수 공식을 이용해서 계산할 수 있다.

예를 들어 $M_{1,1}$을 구하는 데 있어 첫 번째 행을 선택하기로 했다고 하자($i_0 = 1$), 그럼 여인수 전개식은 다음과 같아진다.

$$M_{1,1} = \left\{(-1)^{1+1} M'_{1,1}\right\} a'_{1,1} + \left\{(-1)^{2+1} M'_{1,2}\right\} a'_{1,2} \tag{2.3.38}$$

$$\begin{bmatrix} \backslash\text{cancel}5 & \backslash\text{cancel}6 \\ \backslash\text{cancel}8 & 9 \end{bmatrix} \rightarrow M'_{1,1} = \det([9]) = 9 \tag{2.3.39}$$

$$\begin{bmatrix} \backslash\text{cancel}5 & \backslash\text{cancel}6 \\ 8 & \backslash\text{cancel}9 \end{bmatrix} \rightarrow M'_{1,2} = \det([8]) = 8 \tag{2.3.40}$$

$$M_{1,1} = 9 \cdot 5 - 8 \cdot 6 = -3 \tag{2.3.41}$$

마찬가지 방법으로

$$M_{2,1} = -6 \tag{2.3.42}$$

$$M_{3,1} = -3 \tag{2.3.43}$$

가 되고 원래의 행렬식의 값은

$$\det(A) = -3 - (-6) \cdot 4 + (-3) \cdot 7 = 0 \tag{2.3.44}$$

넘파이에서는 linalg 서브패키지의 `det()` 명령으로 행렬식을 계산할 수 있다.

```
A = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
np.linalg.det(A)
```

```
-9.51619735392994e-16
```

위의 정의를 사용하면 크기가 2x2, 3x3인 정방행렬의 행렬식의 값은 다음 공식으로 계산할 수 있다.

- 2×2 행렬의 행렬식

$$\det\left(\begin{bmatrix} a & b \\ c & d \end{bmatrix}\right) = ad - bc \tag{2.3.45}$$

- 3×3 행렬의 행렬식

$$\det\left(\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}\right) = aei + bfg + cdh - ceg - bdi - afh \tag{2.3.46}$$

행렬식의 값도 대각합과 마찬가지로 **음수가 될 수 있다.**

## 연습 문제 2.3.6

행렬식의 정의를 사용하여 2x2 행렬과 3x3 행렬의 행렬식이 각각 위와 같이 된다는 것을 증명하라.

행렬식은 다음과 같은 성질을 만족한다.

- 전치 행렬의 행렬식은 원래의 행렬의 행렬식과 같다.

$$\det(A^T) = \det(A) \tag{2.3.47}$$

- 항등 행렬의 행렬식은 1이다.

$$\det(I) = 1 \tag{2.3.48}$$

- 두 행렬의 곱의 행렬식은 각 행렬의 행렬식의 곱과 같다.

$$\det(AB) = \det(A)\det(B) \tag{2.3.49}$$

- **역행렬** $A^{-1}$은 원래의 행렬 $A$와 다음 관계를 만족하는 정방행렬을 말한다. $I$는 항등 행렬(identity matrix)이다.

$$A^{-1}A = AA^{-1} = I \tag{2.3.50}$$

- 역행렬의 행렬식은 원래의 행렬의 행렬식의 역수와 같다.

$$\det(A^{-1}) = \frac{1}{\det(A)} \tag{2.3.51}$$

위 식은 역행렬의 정의와 여인수 전개식을 사용하여 증명할 수 있다.

$$\det(A)\det(A^{-1}) = \det I = 1 \tag{2.3.52}$$

## 연습 문제 2.3.7

다음 행렬이 양의 정부호인지, 양의 준정부호인지 혹은 두가지 중 어느것도 아닌지 판단하라. 그리고 행렬의 대각합과 행렬식을 구하라.

(1)

Print to PD

$$\begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix} \tag{2.3.53}$$

(2)

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \tag{2.3.54}$$

2 Comments - *powered by utteranc.es*

chiwanii commented on 2021년 6월 6일

import numpy as np # 넘파이 패키지 임포트
from sklearn.datasets import fetch_olivetti_faces

A1 = np.array([[2, -1, 0],[-1, 2, -1],[0, -1, 2]])
A2 = np.arange(4).reshape((2,2))
B1 = np.linalg.norm(A1)
B2 = np.linalg.norm(A2)
C1 = np.trace(A1)
C2 = np.trace(A2)
D1 = np.linalg.det(A1)
D2 = np.linalg.det(A2)

ghdakrk commented on 2021년 9월 27일

def det(a):
if len(a) == 1:
return a[0]

```
result = 0
for i in range(len(a)):   # j = 1
    m = a
    m = np.delete(m, 0, axis=1)
    m = np.delete(m, i, axis=0)
    result += ((-1)**(i+1)) * det(m) * a[i][0]  # j-1

return result
```

det(a)

Write Preview

Sign in to comment

By 김도형