3.3 고윳값 분해

고윳값 분해와 다음에 설명할 특잇값 분해는 행렬의 내부 구조를 살펴보거나 행렬을 이용한 연산을 더 효율적으로 할 때 유용하다. 고윳값 분해의 정의를 살펴보고 이와 관련된 다양한 정리도 살펴보자. 고윳값 분해와 관련된 정리는 증명이 복잡하고 이 책의 범위를 넘는 경우가 많으므로 대부분 증명을 생략하도록 하겠다. 하지만 정리 자체는 데이터 분석에서 많이 사용되기 때문에 알아두기 바란다.

고윳값과 고유벡터

정방 행렬 A에 대해 다음 식을 만족하는 영벡터가 아닌 벡터 v, 실수 λ를 찾을 수 있다고 가정하자.

$$Av = \lambda v \tag{3.3.1}$$

위 식을 만족하는 실수 λ를 **고윳값**(eigenvalue), 벡터 v를 **고유벡터**(eigenvector)라고 한다. 고윳값과 고유벡터를 찾는 작업을 **고유분해(eigen-decomposition)** 또는 **고윳값 분해(eigenvalue decomposition)**라고 한다.

행렬 A의 고유벡터는 행렬 A를 곱해서 변환을 해도 방향이 바뀌지 않는 벡터다. 고윳값은 변환된 고유벡터와 원래 고유벡터의 크기 비율이다.

위 식은 다음처럼 쓸 수도 있다.

$$Av - \lambda v = (A - \lambda I)v = 0 \tag{3.3.2}$$

예제

행렬 A

$$A = \begin{bmatrix} 1 & -2 \\ 2 & -3 \end{bmatrix} \tag{3.3.3}$$

에 대해 다음 스칼라 값과 벡터는 각각 고윳값, 고유벡터가 된다.

$$\lambda = -1 \tag{3.3.4}$$

$$v = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{3.3.5}$$

$$Av = \begin{bmatrix} 1 & -2 \\ 2 & -3 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \end{bmatrix} = (-1) \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \lambda v \tag{3.3.6}$$

어떤 벡터 v가 고유벡터가 되면 이 벡터에 실수를 곱한 벡터 cv,즉 v와 방향이 같은 벡터는 모두 고유벡터가 된다.

$$A(cv) = cAv = c\lambda v = \lambda(cv) \tag{3.3.7}$$

그래서 보통 고유벡터를 표시할 때는 길이가 1인 단위벡터가 되도록 다음처럼 정규화(normalization)를 한다.

$$\frac{v}{\|v\|} \tag{3.3.8}$$

따라서 위 행렬 A의 고유값-고유벡터는 보통 다음처럼 나타낸다.

$$\lambda = -1 \tag{3.3.9}$$

$$v = \begin{bmatrix} \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} \end{bmatrix} \approx \begin{bmatrix} 0.7071 \\ 0.7071 \end{bmatrix} \tag{3.3.10}$$

연습 문제 3.3.1

다음 행렬 B가

$$B = \begin{bmatrix} 2 & 3 \\ 2 & 1 \end{bmatrix} \tag{3.3.11}$$

다음과 같은 두 가지 고윳값-고유벡터를 가짐을 증명하라.

$$\lambda_1 = 4, \quad v_1 = \begin{bmatrix} 3 \\ 2 \end{bmatrix} \tag{3.3.12}$$

$$\lambda_2 = -1, \quad v_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix} \tag{3.3.13}$$

또는

$$\lambda_1 = 4, \quad v_1 = \begin{bmatrix} \dfrac{3}{\sqrt{13}} \\ \dfrac{2}{\sqrt{13}} \end{bmatrix} \approx \begin{bmatrix} 0.8321 \\ 0.5547 \end{bmatrix} \tag{3.3.14}$$

$$\lambda_2 = -1, \quad v_2 = \begin{bmatrix} -\dfrac{1}{\sqrt{2}} \\ \dfrac{1}{\sqrt{2}} \end{bmatrix} \approx \begin{bmatrix} -0.7071 \\ 0.7071 \end{bmatrix} \tag{3.3.15}$$

특성방정식

지금까지는 행렬과 그 행렬의 고윳값-고유벡터를 주고 이들이 정말 고윳값-고유벡터인지를 계산으로 증명했다. 그러면 행렬만 주어졌을 대 고윳값-고유벡터는 어떻게 구할 수 있을까?

행렬 A의 고유값은 $A - \lambda I$의 행렬식이 0이 되도록 하는 **특성방정식(characteristic equation)**의 해를 구하면 된다.

$$\det\left(A - \lambda I\right) = 0 \tag{3.3.16}$$

이 조건은 행렬 $A - \lambda I$가 역행렬이 존재하지 않는다는 뜻이다. 만약 $A - \lambda I$의 역행렬이 존재한다면 고윳값 조건을 만족하는 벡터가 항상 영벡터가 되기 때문이다.

$$(A - \lambda I)^{-1}(A - \lambda I)v = 0 \quad \rightarrow \quad v = 0 \tag{3.3.17}$$

예제

행렬

$$A = \begin{bmatrix} 1 & -2 \\ 2 & -3 \end{bmatrix} \tag{3.3.18}$$

에 대해서는 특성방정식이 다음과 같다.

$$\begin{aligned} \det\left(A - \lambda I\right) &= \det\left(\begin{bmatrix} 1 & -2 \\ 2 & -3 \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix}\right) \\ &= \det\begin{bmatrix} 1-\lambda & -2 \\ 2 & -3-\lambda \end{bmatrix} \\ &= (1-\lambda)(-3-\lambda) + 4 \\ &= \lambda^2 + 2\lambda + 1 = 0 \end{aligned} \tag{3.3.19}$$

인수분해를 하여 이차방정식인 특성방정식을 풀면

$$\lambda^2 + 2\lambda + 1 = (\lambda + 1)^2 = 0 \tag{3.3.20}$$

에서 고윳값은 -1이다.

원래 이차방정식은 해를 최대 2개 가질 수 있지만, 이 경우에는 하나만 존재하기 때문에 이러한 해를 **중복고윳값(repeated eigenvalue)**이라고 한다.

예제

행렬

$$B = \begin{bmatrix} 2 & 3 \\ 2 & 1 \end{bmatrix} \tag{3.3.21}$$

에 대해서는 특성방정식이 다음과 같다.

$$\begin{aligned} \det\left(B - \lambda I\right) &= \det\left(\begin{bmatrix} 2 & 3 \\ 2 & 1 \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix}\right) \\ &= \det\begin{bmatrix} 2-\lambda & 3 \\ 2 & 1-\lambda \end{bmatrix} \\ &= (2-\lambda)(1-\lambda) - 6 \\ &= \lambda^2 - 3\lambda - 4 = 0 \end{aligned} \tag{3.3.22}$$

인수분해를 하여 이차방정식인 특성방정식을 풀면

$$\lambda^2 - 3\lambda - 4 = (\lambda - 4)(\lambda + 1) = 0 \tag{3.3.23}$$

에서 고윳값은 4와 -1이다.

예제

2차 방정식의 실수 해가 존재하지 않는 경우도 있기 때문에 실수 고유값이 없는 행렬도 있을 수 있다.

행렬

$$C = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \tag{3.3.24}$$

의 특성방정식은 다음과 같다.

$$\begin{aligned} \det\left(C - \lambda I\right) &= \det\left(\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix}\right) \\ &= \lambda^2 + 1 \\ &= 0 \end{aligned} \tag{3.3.25}$$

이 특성방정식의 실수해는 존재하지 않음을 알 수 있다. 따라서 행렬 C는 실수인 고유값을 가지지 않는다.

만약 고유값-고유벡터가 복소수(complex number)가 되어도 괜찮다면 행렬 C는 2개의 고윳값을 가진다고 할 수 있다.

$$\lambda = i, \quad \lambda = -i \tag{3.3.26}$$

연습 문제 3.3.2

특성방정식을 이용하여 다음 행렬의 고윳값을 구하라.

$$D = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \tag{3.3.27}$$

고윳값의 개수

N차방정식이 항상 N개의 복소수 해를 가진다는 사실을 이용하면 N차원 정방행렬의 고윳값의 개수에 대해 다음 정리가 성립한다.

> **[정리] 중복된 고윳값을 각각 별개로 생각하고 복소수인 고윳값도 고려한다면 N차원 정방행렬의 고윳값은 항상 N개다.**

고윳값과 대각합/행렬식

어떤 행렬의 고윳값이 $\lambda_1, \lambda_2, \cdots, \lambda_N$이라고 하면 **모든 고윳값의 곱은 행렬식의 값**과 같고 **모든 고윳값의 합은 대각합(trace)의 값**과 같다.

$$\det(A) = \prod_{i=1}^{N} \lambda_i \tag{3.3.28}$$

$$\mathrm{tr}(A) = \sum_{i=1}^{N} \lambda_i \tag{3.3.29}$$

예제

행렬 A에 대해서 대각합과 행렬식은 다음과 같다.

$$\mathrm{tr}(A) = 1 + (-3) = -2 \tag{3.3.30}$$

$$\det(A) = 1 \cdot (-3) - 2 \cdot (-2) = 1 \tag{3.3.31}$$

그런데 고윳값이 $\lambda_1 = -1, \lambda_2 = -1$ (중복된 고윳값)이므로

$$\lambda_1 + \lambda_2 = -2 = \mathrm{tr}(A) \tag{3.3.32}$$

$$\lambda_1 \cdot \lambda_2 = 1 = \det(A) \tag{3.3.33}$$

가 성립한다.

예제

행렬 B에 대해서도 고윳값이 $\lambda_1 = 4, \lambda_2 = -1$이고

$$\lambda_1 + \lambda_2 = 3 = \mathrm{tr}(B) = 2 + 1 = 3 \tag{3.3.34}$$

$$\lambda_1 \cdot \lambda_2 = -4 = \det(B) = 2 \cdot 1 - 2 \cdot 3 = -4 \tag{3.3.35}$$

가 성립한다.

고유벡터의 계산

고윳값을 알면 다음 연립 방정식을 풀어 고유벡터를 구할 수 있다.

$$(A - \lambda I)v = 0 \tag{3.3.36}$$

예제

앞에서 예로 든 행렬 A에 대해서는

$$\begin{bmatrix} 1+1 & -2 \\ 2 & -3+1 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = 0 \tag{3.3.37}$$

$$\begin{bmatrix} 2 & -2 \\ 2 & -2 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = 0 \tag{3.3.38}$$

이므로

$$2v_1 - 2v_2 = 0 \tag{3.3.39}$$

즉,

$$v_1 = v_2 \tag{3.3.40}$$

를 만족하는 모든 벡터가 고유벡터임을 알 수 있다. 즉

$$v = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{3.3.41}$$

또는 단위벡터

$$v = \begin{bmatrix} \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} \end{bmatrix} \tag{3.3.42}$$

가 유일한 고유벡터다. 중복된(repeated) 고유벡터라고도 한다.

예제

고윳값이 중복되었다고 고유벡터도 항상 중복되는 것은 아니다. 예를 들어 항등행렬 I의 고윳값은 1로 중복된 고윳값을 가진다.

$$\det(I-\lambda I)=\det(\left(\begin{bmatrix}1-\lambda & 0\\ 0 & 1-\lambda\end{bmatrix}\right)=(\lambda-1)^2=0 \tag{3.3.43}$$

하지만 이 값을 고유값과 고유벡터 정의에 대입하면

$$\begin{bmatrix}0 & 0\\ 0 & 0\end{bmatrix}\begin{bmatrix}v_1\\ v_2\end{bmatrix}=0 \tag{3.3.44}$$

으로 임의의 2차원 벡터는 모두 고유벡터가 된다. 즉

$$\begin{bmatrix}1\\ 0\end{bmatrix},\quad \begin{bmatrix}0\\ 1\end{bmatrix} \tag{3.3.45}$$

둘 다 고유벡터이다.

연습 문제 3.3.3

특성방정식을 이용하여 다음 행렬의 고윳값과 고유벡터를 구하라.

(1)

$$E=\begin{bmatrix}2 & 3\\ 2 & 1\end{bmatrix} \tag{3.3.46}$$

(2)

$$F=\begin{bmatrix}1 & 1\\ 0 & 1\end{bmatrix} \tag{3.3.47}$$

연습 문제 3.3.4

중복된 고윳값 λ에 대해 서로 다른 고유벡터 v_1, v_2가 존재하면 이 두 벡터의 선형조합

$$c_1v_1+c_2v_2 \tag{3.3.48}$$

도 고윳값 λ에 대한 고유벡터임을 증명하라.

넘파이를 사용한 고유분해

넘파이의 linalg 서브패키지에서는 고윳값과 고유벡터를 구할 수 있는 `eig()` 명령을 제공한다. 고윳값은 벡터의 형태로, 고유벡터는 고유벡터 행렬의 형태로 묶여서 나오고 고유벡터는 크기가 1인 단위벡터로 정규화가 되어 있다. 실수인 고윳값이 존재하지 않는 행렬에 대해서는 복소수인 고윳값과 고유벡터를 계산한다.

`eig()` 명령의 결과로 나오는 고유벡터 행렬은 행이 아니라 **열을 고유벡터로 가진다**는 점에 주의한다. 수치계산의 오류로 인해 중복되는 고윳값이 미세하게 다른 값으로 계산될 수도 있다.

In [1]:

```
A = np.array([[1, -2], [2, -3]])
w1, V1 = np.linalg.eig(A)

print(w1)
print(V1)
```

```
[-0.99999998 -1.00000002]
[[0.70710678 0.70710678]
 [0.70710678 0.70710678]]
```

In [2]:

```
B = np.array([[2, 3], [2, 1]])
w2, V2 = np.linalg.eig(B)

print(w2)
print(V2)
```

```
[ 4. -1.]
[[ 0.83205029 -0.70710678]
 [ 0.5547002   0.70710678]]
```

In [3]:

```
C = np.array([[0, -1], [1, 0]])
w3, V3 = np.linalg.eig(C)

print(w3)
print(V3)
```

```
[0.+1.j 0.-1.j]
[[0.70710678+0.j         0.70710678-0.j        ]
 [0.        -0.70710678j 0.        +0.70710678j]]
```

연습 문제 3.3.5

지금까지 연습 문제에 나온 행렬들에 대해 NumPy를 사용하여 고유분해를 하라.

대각화

N 차원의 정방 행렬 A가 N개의 복소수 고윳값과 이에 대응하는 고유벡터를 가진다는 성질을 이용하면 다음처럼 행렬을 분해할 수 있다.

행렬 A의 고윳값과 이에 대응하는 단위벡터인 고유벡터를 각각

$$\lambda_1, \lambda_2, \cdots, \lambda_N \quad v_1, v_2, \cdots, v_N \tag{3.3.49}$$

이라고 하자.

이 고윳값과 고유벡터를 묶어서 다음과 같이 고유벡터행렬, 고윳값행렬을 정의할 수 있다.

고유벡터행렬 V은 고유벡터를 열벡터로 옆으로 쌓아서 만든 행렬이다.

$$V = [v_1 \cdots v_N] \tag{3.3.50}$$

$$V \in \mathbf{R}^{N \times N} \tag{3.3.51}$$

고윳값행렬 Λ은 고윳값을 대각성분으로 가지는 대각행렬이다.

$$\Lambda = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_N \end{bmatrix} \tag{3.3.52}$$

$$\Lambda \in \mathbf{R}^{N \times N} \tag{3.3.53}$$

위와 같이 고유벡터행렬과 고윳값행렬을 정의하면 **행렬과 고유벡터행렬의 곱은 고유벡터행렬과 고윳값행렬의 곱과 같다**.

$$\begin{aligned} AV &= A \begin{bmatrix} v_1 \cdots v_N \end{bmatrix} \\ &= \begin{bmatrix} Av_1 \cdots Av_N \end{bmatrix} \\ &= \begin{bmatrix} \lambda_1 v_1 \cdots \lambda_N v_N \end{bmatrix} \\ &= \begin{bmatrix} v_1 \cdots v_N \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_N \end{bmatrix} \\ &= V\Lambda \end{aligned} \tag{3.3.54}$$

즉,

$$AV = V\Lambda \tag{3.3.55}$$

만약 **고유벡터행렬 V의 역행렬이 존재한다면** 행렬을 다음처럼 고유벡터행렬과 고윳값행렬의 곱으로 표현할 수 있다. 이를 행렬의 **대각화(diagonalization)**라고 한다.

$$A = V\Lambda V^{-1} \tag{3.3.56}$$

예제

위에서 예로 든 행렬 B를 대각화하면 다음과 같다.

$$V = \begin{bmatrix} \frac{3}{\sqrt{13}} & -\frac{1}{\sqrt{2}} \\ \frac{2}{\sqrt{13}} & \frac{1}{\sqrt{2}} \end{bmatrix} \tag{3.3.57}$$

$$\Lambda = \begin{bmatrix} 4 & 0 \\ 0 & -1 \end{bmatrix} \tag{3.3.58}$$

$$V^{-1} = \frac{1}{5}\begin{bmatrix} \sqrt{13} & \sqrt{13} \\ -2\sqrt{2} & 3\sqrt{2} \end{bmatrix} \tag{3.3.59}$$

$$B = \begin{bmatrix} 2 & 3 \\ 2 & 1 \end{bmatrix} = V\Lambda V^{-1} = \frac{1}{5}\begin{bmatrix} \frac{3}{\sqrt{13}} & -\frac{1}{\sqrt{2}} \\ \frac{2}{\sqrt{13}} & \frac{1}{\sqrt{2}} \end{bmatrix}\begin{bmatrix} 4 & 0 \\ 0 & -1 \end{bmatrix}\begin{bmatrix} \sqrt{13} & \sqrt{13} \\ -2\sqrt{2} & 3\sqrt{2} \end{bmatrix} \tag{3.3.60}$$

NumPy을 이용하여 위 식을 계산하면 좌변과 우변이 같음을 확인할 수 있다.

In [4]:

```
V2
```

Out[4]:

```
array([[ 0.83205029, -0.70710678],
       [ 0.5547002 ,  0.70710678]])
```

In [5]:

```
V2_inv = np.linalg.inv(V2)
V2_inv
```

Out[5]:

```
array([[ 0.72111026,  0.72111026],
       [-0.56568542,  0.84852814]])
```

In [6]:

```
V2 @ np.diag(w2) @ V2_inv
```

Out[6]:

```
array([[2., 3.],
       [2., 1.]])
```

연습 문제 3.3.6

다음 행렬을 고윳값과 고유벡터로 대각화하라.

$$\begin{bmatrix} 2 & 3 \\ 2 & 1 \end{bmatrix} \tag{3.3.61}$$

연습 문제 3.3.7

다음 행렬은 고윳값과 고유벡터로 대각화 가능한가?

$$\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \tag{3.3.62}$$

그림 3.3.1 : 정방행렬에 대한 고윳값-고유벡터의 성질

대각화가능

> **[정리] 행렬이 대각화가능하려면 고유벡터가 선형독립이어야 한다.**

행렬을 대각화할 수 있으면 **대각화가능(diagonalizable) 행렬**이라고 한다. 앞서 이야기했듯이 고유벡터인 열벡터로 이루어진 행렬에 역행렬이 존재하면 대각화가능이라고 했다. 그런데 앞절에서 정방행렬의 역행렬이 존재할 조건은 정방행렬의 열벡터 즉, 고유벡터들이 선형독립인 경우이다. 따라서 행렬이 대각화가능하려면 고유벡터가 선형독립이어야한다.

고윳값과 역행렬

> **[정리] 대각화가능한 행렬에 0인 고유값이 없으면 항상 역행렬이 존재한다.**

이는 다음과 같이 증명할 수 있다. 행렬 A가 대각화가능하면 다음처럼 표현할 수 있다.

$$A = V\Lambda V^{-1} \tag{3.3.63}$$

이 행렬의 역행렬은 다음처럼 계산한다.

$$A^{-1} = (V\Lambda V^{-1})^{-1} = V\Lambda^{-1}V^{-1} \tag{3.3.64}$$

대각행렬의 역행렬은 각 대각성분의 역수로 이루어진 대각행렬이므로 0인 고유값만 없으면 항상 역행렬이 존재한다.

연습 문제 3.3.8

다음 행렬

$$\begin{bmatrix} 2 & 3 \\ 2 & 1 \end{bmatrix} \tag{3.3.65}$$

의 고윳값과 고유벡터는 다음과 같다. 이 정보를 이용하여 역행렬을 계산하라.

$$\lambda_1 = 4, \quad v_1 = \begin{bmatrix} \dfrac{3}{\sqrt{13}} \\ \dfrac{2}{\sqrt{13}} \end{bmatrix} \tag{3.3.66}$$

$$\lambda_2 = -1, \quad v_2 = \begin{bmatrix} -\dfrac{1}{\sqrt{2}} \\ \dfrac{1}{\sqrt{2}} \end{bmatrix} \tag{3.3.67}$$

대칭행렬의 고유분해

대칭행렬에 대해서는 다음 정리가 성립한다.

> **[정리] 행렬 A가 실수인 대칭행렬이면 고유값이 실수이고 고유벡터는 서로 직교(orthogonal)한다.**

만약 고유벡터들이 크기가 1이 되도록 정규화된 상태라면 고유벡터 행렬 V는 정규직교(orthonormal) 행렬이므로 **전치행렬이 역행렬**이다.

$$V^T V = V V^T = I \tag{3.3.68}$$

$$V^{-1} = V^T \tag{3.3.69}$$

따라서 대각화가 가능하고 다음처럼 쓸 수 있다.

$$A = V \Lambda V^T \tag{3.3.70}$$

이 사실로부터 다음 정리도 도출된다.

> **[정리] 실수인 대칭행렬은 항상 대각화가능**하다.

대칭행렬을 랭크-1 행렬의 합으로 분해

N차원 대칭행렬 A는 다음처럼 N개의 랭크-1 행렬 $A_i = v_i v_i^T$ 의 합으로 표시할 수 있다.

$$\begin{aligned} A &= V\Lambda V^T \\ &= \begin{bmatrix} v_1 & v_2 & \cdots & v_N \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_N \end{bmatrix} \begin{bmatrix} v_1^T \\ v_2^T \\ \vdots \\ v_N^T \end{bmatrix} \\ &= \begin{bmatrix} \lambda_1 v_1 & \lambda_2 v_2 & \cdots & \lambda_N v_N \end{bmatrix} \begin{bmatrix} v_1^T \\ v_2^T \\ \vdots \\ v_N^T \end{bmatrix} \end{aligned} \tag{3.3.71}$$

따라서 N차원 대칭행렬 A는

$$A = \sum_{i=1}^{N} \lambda_i v_i v_i^T = \sum_{i=1}^{N} \lambda_i A_i = \lambda_1 A_1 + \cdots + \lambda_N A_N \tag{3.3.72}$$

예제

대칭행렬

$$\begin{bmatrix} 60 & 30 & 20 \\ 30 & 20 & 15 \\ 20 & 15 & 12 \end{bmatrix} \tag{3.3.73}$$

를 NumPy를 사용하여 다음처럼 세 개의 랭크-1 행렬로 나눌 수 있다.

In [7]:

```
A = np.array([[60., 30., 20.],
              [30., 20., 15.],
              [20., 15., 12.]])

w, V = np.linalg.eig(A)
w1, w2, w3 = w
v1 = V[:, 0:1]
v2 = V[:, 1:2]
v3 = V[:, 2:3]
A1 = v1 @ v1.T
A2 = v2 @ v2.T
A3 = v3 @ v3.T
```

In [8]:

```
w
```

Out[8]:

```
array([84.49913563,  7.33962395,  0.16124042])
```

In [9]:

```
w1 * A1
```

Out[9]:

```
array([[57.79768857, 32.13739648, 22.59357583],
       [32.13739648, 17.8694387 , 12.56276371],
       [22.59357583, 12.56276371,  8.83200836]])
```

In [10]:

```
w2 * A2
```

Out[10]:

```
array([[ 2.19968372, -2.12270483, -2.60775134],
       [-2.12270483,  2.04841985,  2.51649195],
       [-2.60775134,  2.51649195,  3.09152039]])
```

In [11]:

```
w3 * A3
```

Out[11]:

```
array([[ 0.00262772, -0.01469165,  0.01417551],
       [-0.01469165,  0.08214145, -0.07925566],
       [ 0.01417551, -0.07925566,  0.07647125]])
```

In [12]:

```
w1 * A1 + w2 * A2 + w3 * A3
```

Out[12]:

```
array([[60., 30., 20.],
       [30., 20., 15.],
       [20., 15., 12.]])
```

만약 0인 고윳값이 없다면 역행렬도 다음처럼 N개의 랭크-1 행렬 $A_i = v_i v_i^T$ 의 합으로 표시할 수 있다.

$$A^{-1} = V\Lambda^{-1}V^T = \sum_{i=1}^{N} \frac{1}{\lambda_i} v_i v_i^T = \frac{1}{\lambda_1} A_1 + \cdots + \frac{1}{\lambda_N} A_N \tag{3.3.74}$$

앞에서 예로 든 대칭행렬의 역행렬도 다음처럼 랭크-1 행렬의 합으로 나타난다.

In [13]:

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

Out[13]:

```
array([[ 0.15, -0.6 ,  0.5 ],
       [-0.6 ,  3.2 , -3.  ],
       [ 0.5 , -3.  ,  3.  ]])
```

In [14]:

```
1 / w1 * A1
```

Out[14]:

```
array([[0.0080948 , 0.00450097, 0.00316432],
       [0.00450097, 0.00250269, 0.00175947],
       [0.00316432, 0.00175947, 0.00123696]])
```

In [15]:

```
1 / w2 * A2
```

Out[15]:

```
array([[ 0.04083313, -0.03940415, -0.04840816],
       [-0.03940415,  0.03802519,  0.04671409],
       [-0.04840816,  0.04671409,  0.05738845]])
```

In [16]:

```
1 / w3 * A3
```

Out[16]:

```
array([[ 0.10107208, -0.56509682,  0.54524384],
       [-0.56509682,  3.15947213, -3.04847356],
       [ 0.54524384, -3.04847356,  2.94137459]])
```

In [17]:

```
1 / w1 * A1 + 1 / w2 * A2 + 1 / w3 * A3
```

Out[17]:

```
array([[ 0.15, -0.6 ,  0.5 ],
       [-0.6 ,  3.2 , -3.  ],
       [ 0.5 , -3.  ,  3.  ]])
```

대칭행렬의 고윳값 부호

대칭행렬이 위와 같이 랭크-1 행렬의 합으로 표시되고 고유벡터가 서로 직교한다는 성질을 이용하면 다음 정리를 증명할 수 있다.

> **[정리] 대칭행렬이 양의 정부호(positive definite)이면 고윳값은 모두 양수다. 역도 성립한다.**
>
> **[정리] 대칭행렬이 양의 준정부호(positive semidefinite)이면 고윳값은 모두 0이거나 양수다. 역도 성립한다.**

여기에서는 첫 번째 정리만 증명해보자. 두 번째 정리도 비슷한 방법으로 증명할 수 있다.

대칭행렬은 랭크-1 행렬의 합으로 표시된다고 하였다.

$$A = \sum_{i=1}^{N} \lambda_i v_i v_i^T \tag{3.3.75}$$

만약 대칭행렬이 양의 정부호이면 어떤 벡터 x를 행렬 A의 앞뒤에 곱해 이차형식을 만들어도 0보다 커야 하므로 j번째 고유벡터 $x = v_j$를 선택하여 곱해도 마찬가지다.

$$v_j^T A v_j > 0 \tag{3.3.76}$$

그런데 대칭행렬은 고유벡터들이 서로 직교한다.

$$v_i^T v_j = 0 \;(\text{if } i \neq j) \tag{3.3.77}$$

$$v_i^T v_i = 1 \tag{3.3.78}$$

따라서

$$v_j^T A v_j = v_j^T \left(\sum_{i=1}^{N} \lambda_i v_i v_i^T \right) v_j = \sum_{i=1}^{N} \lambda_i v_j^T v_i v_i^T v_j = \lambda_j > 0 \tag{3.3.79}$$

이므로 양수인 고윳값만 가진다.

반대로 대칭행렬의 고윳값이 모두 양수이면 그 행렬은 양의 정부호가 됨을 증명하자. 우선 고유분해로 만들어진 랭크-1 행렬 $A_i = v_i v_i^T$는 양의 준정부호(positive semidefinite)임을 증명할 수 있다.

$$x^T A_i x = x^T v_i v_i^T x = (x^T v_i)(x^T v_i)^T = (x^T v_i)(x^T v_i) = \|x^T v_i\|^2 \geq 0 \tag{3.3.80}$$

이 식에서 x가 v_i와 직교(orthogonal)인 경우에만 0이 된다는 것을 알 수 있다. 고윳값 λ_i가 모두 양수이므로 따라서 행렬 $\lambda_i A_i$를 모두 더한 행렬 $\lambda_1 A_1 + \cdots + \lambda_N A_N$도 양의 준정부호다.

$$\begin{aligned} x^T A x &= \lambda_1 x^T A_1 x + \cdots + \lambda_N x^T A_N x \\ &= \lambda_1 \|x^T v_1\|^2 + \cdots + \lambda_N \|x^T v_N\|^2 \geq 0 \end{aligned} \tag{3.3.81}$$

그런데 이 값은 실제로는 0이 될 수 없다. 왜냐하면 이 값이 0이려면 모든 $x^T v_i$가 0, 다시 말해 x와 모든 v_i가 직교해야 하는데 대칭행렬의 고유벡터의 집합은 N 차원에서 기저벡터를 이루기 때문에 동시에 모든 기저벡터와 직교인 벡터는 존재하지 않기 때문이다. 따라서 양의 정부호다.

그림 3.3.2 : 대칭행렬에 대한 고윳값-고유벡터의 성질

분산행렬

임의의 실수 행렬 X에 대해 $X^T X$인 정방행렬을 **분산행렬(scatter matrix)**이라고 한다. 분산행렬의 의미는 확률 분포에서 더 자세하게 공부할 것이다. 일단은 위와 같은 방법으로 계산되는 행렬을 가리키는 명칭이라는 것만 알아두자.

분산행렬에 대해서는 다음 정리가 성립한다.

> **[정리] 분산행렬은 양의 준정부호(positive semidefinite)이고 고윳값은 0보다 같거나 크다.**

임의의 영벡터가 아닌 벡터 x에 대해 분산행렬에 대한 이차형식을 구하면

$$x^T(X^T X)x = (Xx)^T(Xx) = u^T u \geq 0 \tag{3.3.82}$$

로 어떤 벡터 u의 제곱합이 된다. 따라서 이 값은 0보다 같거나 크고 분산행렬은 양의 준정부호다. 그런데 분산행렬은 대칭행렬이므로 양의준정부호이면 고유값이 모두 0이상이다.

연습 문제 3.3.9

(1) 붓꽃(Iris) 특징데이터 행렬 X의 분산행렬을 구하고 이 분산행렬의 고윳값들을 구하라.

(2) 보스턴 집값(Boston House Price) 특징데이터 행렬 X의 분산행렬을 구하고 이 분산행렬의 고윳값들을 구하라.

그림 3.3.3 : 분산 행렬에 대한 고윳값-고유벡터의 성질

분산행렬의 역행렬

분산행렬에서는 다음 정리가 성립한다.

> **[정리] 행렬 $X \in \mathbf{R}^{N \times M} (N \geq M)$가 풀랭크이면 이 행렬의 분산행렬 $X^T X$의 역행렬이 존재한다.**

행렬 $X \in \mathbf{R}^{N \times M} (N \geq M)$가 풀랭크이면 X의 열벡터가 기저벡터를 이루기 때문에 영벡터가 아닌 모든 벡터 v에 대해 $Xv = u$는 영벡터가 될 수 없다. (만약 영벡터 u를 만드는 영벡터가 아닌 v가 존재한다면 서로 독립이 아니다.) 그러면 $X^T X$의 이차형식은 항상 양수가 된다.

$$v^T(X^T X)v = (Xv)^T(Xv) = u^T u > 0 \tag{3.3.83}$$

따라서 분산행렬은 양의 정부호이고 역행렬이 존재한다.

연습 문제 3.3.10

(1) 양의 정부호인 대칭행렬은 항상 역행렬이 존재하는가?

(2) 역으로 역행렬이 존재하는 대칭행렬은 항상 양의 정부호인가?

그림 3.3.4 : 양의 정부호 행렬에 대한 고윳값-고유벡터의 성질

요약: 고유분해의 성질

지금까지 나왔던 고유분해와 관련된 정리를 다시 한 번 요약하였다. 이 정리들은 데이터 분석에서 자주 사용되므로 잘 알아두자.

N차원 정방행렬 A에 대해

1. 행렬 A는 N개의 고윳값-고유벡터를 가진다(복소수인 경우와 중복인 경우를 포함).
2. 행렬의 **대각합은 모든 고윳값의 합**과 같다.
3. 행렬의 **행렬식은 모든 고윳값의 곱**과 같다.
4. 행렬 A가 **대칭행렬**이면 N개의 **실수 고윳값**을 가지며 고유벡터들이 서로 **직교(orthogonal)**이다.
5. 행렬 A가 **대칭행렬**이고 고윳값이 모두 **양수**이면 **양의 정부호(positive-definite)이고 역행렬이 존재한다**. 역도 성립한다.
6. 행렬 A가 어떤 행렬 X의 **분산행렬** $X^T X$이면 **0 또는 양의 고윳값을 가진다**.
7. 행렬 $X \in \mathbf{R}^{N \times M} (N \geq M)$가 **풀랭크이면 분산행렬** $X^T X$**은 역행렬이 존재**한다.