순천향대학교
SOON CHUN HYANG UNIVERSITY

2024년 2학기

선형대수

5장

Chapter 05

- SECTION 05-1 행렬 노름
- SECTION 05-2 행렬 공간(열, 행, 영)
- SECTION 05-3 계수
- SECTION 05-4 계수 응용
- SECTION 05-5 행렬식

SCH

SECTION 05-1 행렬 노름(1)

- 행렬은 여러 개의 서로 다른 노름을 가짐
  - '단 하나의 행렬 노름'은 없음
  - 각 행렬 노름은 행렬을 특징짓는 하나의 숫자라는 점에서 벡터 노름과 유사
  - 행렬 **A** 의 노름은 ||**A**||와 같이 이중 수직선을 사용해서 나타냄
  - 각 행렬 노름은 서로 다른 의미를 가짐
  - 원소별(또는 항목별) 계열
    - 행렬의 개별 원소를 기반으로 계산
    - 행렬의 원소의 크기를 반영해서 해석
  - 유도(induced) 계열
    - 벡터의 변환으로 인해 벡터의 크기가 얼마나 조정되는지에(늘리거나 줄이거나) 대한 측정치

SECTION 05-1 행렬 노름(2)

원소별 노름

- 유클리드 노름
  - 벡터 노름을 그대로 행렬에 확장한 것
  - 유클리드 노름은 프로베니우스 노름(Frobenius Norm)이라고도 하며 모든 행렬 원소의 제곱합의 제곱근으로 계산

$$\| \mathrm{A} \|_{\mathrm{F}} = \sqrt{\Sigma_{i=1}^{M} \Sigma_{j=1}^{N} \mathrm{a}_{ij}^{2}}$$

  - 인덱스 i 와 j 는 각각 M 행과 N 열에 해당
  - 프로베니우스 노름을 나타내는 아래 첨자 $_{\mathrm{F}}$
  - 프로베니우스 노름은 L2 노름이라고도 함
  - L2 노름은 원소별 p - 노름에 대한 일반 공식(p=2 일 때 프로베니우스 노름)에서 이름을 딴 것

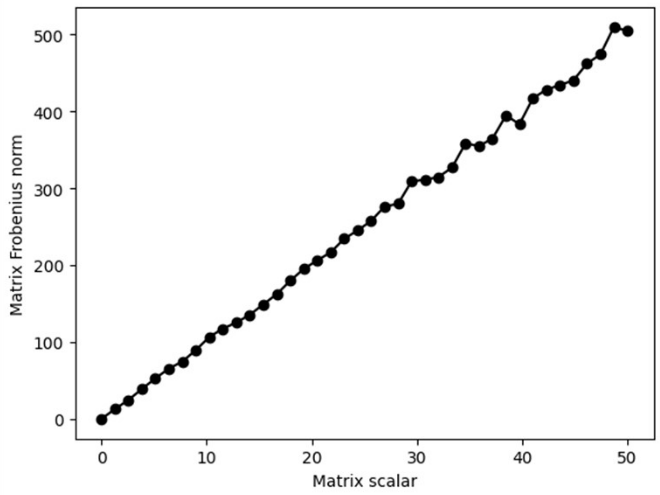
$$\| \mathrm{A} \|_{p} = \left(\sum_{i=1}^{M} \sum_{j=1}^{N} |\mathrm{a}_{ij}|^{p} \right)^{1/p}$$

  - '행렬 거리'를 계산할 때 응용
    - 동일한 행렬 사이의 거리는 0이며 서로 다른 행렬 사이의 거리는 행렬 안의 숫자의 차이가 클수록 증가

실습 – 행렬의 노름

- 행렬의 노름은 행렬이 가진 숫자의 크기와 관련이 있음을 증명

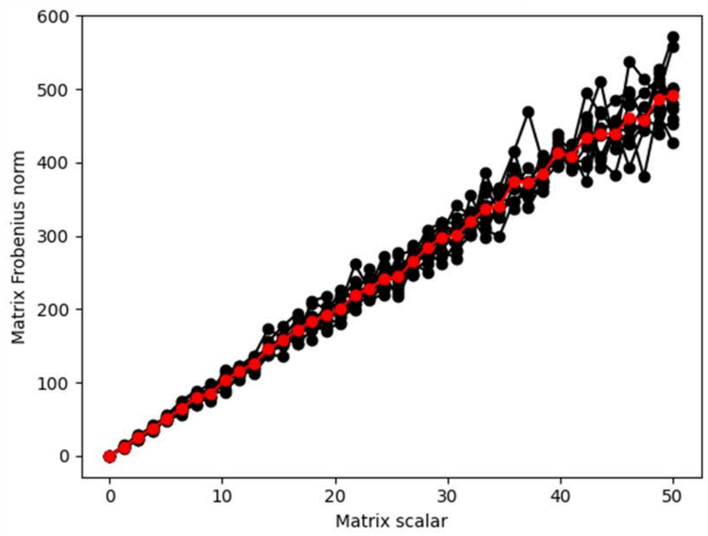
```
import numpy as np
import matplotlib.pyplot as plt

scalingVals = np.linspace(0,50,40) # range of scaling parameters (0 to 50 in 40 steps)
nExperiments = 10

# initialize output
matrixNorms = np.zeros((len(scalingVals),nExperiments))

# run experiment!
for si in range(len(scalingVals)):
  for expi in range(nExperiments):

    # generate a random scaled matrix, 표준정규분포
    R = np.random.randn(10,10) * scalingVals[si]
    matrixNorms[si,expi] = np.linalg.norm(R,'fro') #default:'fro'

# plot the results!
plt.plot(scalingVals,np.mean(matrixNorms,axis=1),'ko-')
plt.xlabel('Matrix scalar')
plt.ylabel('Matrix Frobenius norm')
plt.savefig('Figure_05_07.png',dpi=300)
plt.show()

# check that norm=0 for zeros matrix
print(matrixNorms[0,:])
```



실습 – 행렬의 노름

- 행렬의 노름은 행렬이 가진 숫자의 크기와 관련이 있음을 증명

```
# experiment simulations
scalingVals = np.linspace(0,50,40) # range of scaling parameters (0 to 50 in 40 steps)
nExperiments = 10

# initialize output
matrixNorms = np.zeros((len(scalingVals),nExperiments))

# run experiment!
for si in range(len(scalingVals)):
  for expi in range(nExperiments):

    # generate a random scaled matrix, 표준정규분포
    R = np.random.randn(10,10) * scalingVals[si]

    # store its norm
    matrixNorms[si,expi] = np.linalg.norm(R,'fro') #default:'fro'

# plot the results!
plt.plot(scalingVals,matrixNorms,'ko-')
plt.plot(scalingVals,np.mean(matrixNorms,axis=1),'ko-', color="red")
plt.xlabel('Matrix scalar')
plt.ylabel('Matrix Frobenius norm')
plt.savefig('Figure_05_07.png',dpi=300)
plt.show()

# check that norm=0 for zeros matrix
print(matrixNorms[0,:])
```



SECTION 05-1 행렬 노름(3)

- 행렬 노름
  - 머신러닝과 통계 분석의 여러 곳에서 응용
  - 정규화(regularization)
    - 정규화의 목표는 모델 적합성을 개선하고 발견되지 않은 데이터에 대한 모델의 일반화 성능을 높이는 것
    - 행렬 노름을 최소화 알고리즘에 비용 함수로 추가
    - L2 정규화, 릿지 회귀(ridge regression) - 모델 매개변수가 너무 커지는 것을 방지
    - L2 정규화, 라쏘 회귀(lasso regression) – 희소 결과가 나오는 것을 방지

5.1.1 행렬의 대각합과 프로베니우스 노름

- 행렬의 대각합(trace)
  - 대각 원소의 합이며 tr(A)로 나타냄
  - 정방 행렬에 대해서만 존재

$$\begin{bmatrix} 4 & 5 & 6 \\ 0 & 1 & 4 \\ 9 & 9 & 9 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 0 & 8 & 0 \\ 1 & 2 & 6 \end{bmatrix}$$

- 행렬은 모두 대각합이 14로 동일

  - 행렬의 대각합은 행렬의 고유값의 합과 같고 결국 행렬의 고유공간 (eigenspace)의 '부피'에 대한 측정치가 됨
  - 프로베니우스 노름은 어떤 행렬의 전치와 그 행렬을 곱한 결과의 대각합의 제곱근으로 계산할 수 있음
    - 행렬 $\mathbf{A}^T\mathbf{A}$ 의 각 대각 원소는 동일한 행에 대한 내적이기 때문임

$$\| \mathrm{A} \|_{\mathrm{F}} = \sqrt{\sum_{i=1}^{M} \sum_{j=1}^{N} \mathrm{a}_{ij}^2} = \sqrt{tr(\mathrm{A}^{\mathrm{T}}\mathrm{A})}$$

실습 문제

- 증명
  - 프로베니우스 노름은 어떤 행렬의 전치와 그 행렬을 곱한 결과의 대각합의 제곱근으로 계산됨
    - 행렬 $\mathbf{A}^T\mathbf{A}$ 의 각 대각 원소는 동일한 행에 대한 내적이기 때문임
    - np.trace()

SECTION 05-2 행렬 공간(열, 행, 영)(1)

5.2.1 열공간

- 행렬의 열공간
  - 무한 개의 스칼라로 벡터 집합을 무한히 결합한 결과로 생성된 무한 벡터 집합
  - 열이 하나만 있는 행렬(열벡터와 동일)에서 해당 열의 모든 가능한 선형 가중 결합인 열공간 $C\left(\begin{bmatrix}1\\3\end{bmatrix}\right)=\lambda\begin{bmatrix}1\\3\end{bmatrix},\quad \lambda\in\mathbb{R}$

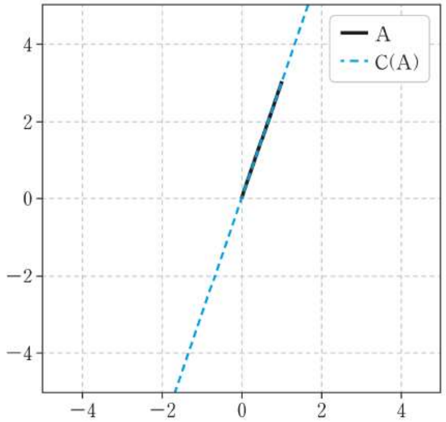


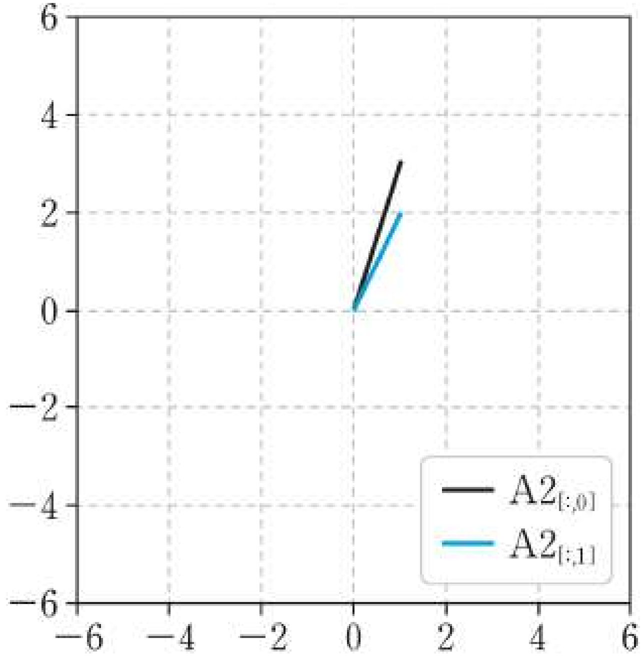
그림 **5-1** 하나의 열을 가진 행렬의 열공간을 시각화. 이 열공간은 **1**차원 부분공간입니다

SECTION 05-2 행렬 공간(열, 행, 영)(2)

- 더 많은 열을 가진 행렬과 열공간
  - 2차원 그래프로 시각화할 수 있도록 열차원의 수를 2로 고정

$$C\left(\begin{bmatrix} 1 & 1 \\ 3 & 2 \end{bmatrix}\right) = \lambda_1 \begin{bmatrix} 1 \\ 3 \end{bmatrix} + \lambda_2 \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \quad \lambda \in \mathbb{R}$$

  - 두 열벡터의 선형 결합으로 생성할 수 있는 모든 벡터들의 집합은 $\mathbb{R}^2$의 모든 벡터

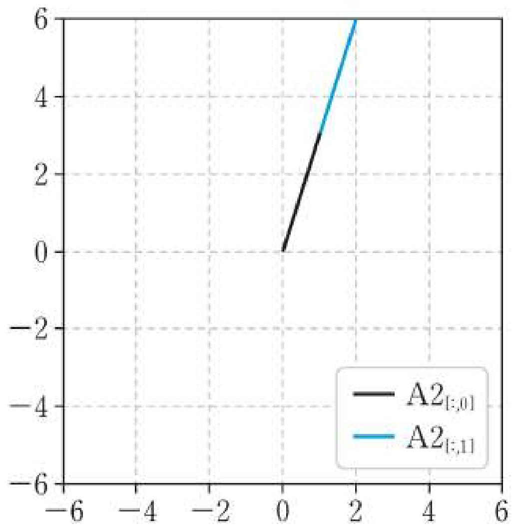


SECTION 05-2 행렬 공간(열, 행, 영)(3)

- $\mathbb{R}^2$의 다른 예시

$$C\left(\begin{bmatrix} 1 & 2 \\ 3 & 6 \end{bmatrix}\right) = \lambda_1 \begin{bmatrix} 1 \\ 3 \end{bmatrix} + \lambda_2 \begin{bmatrix} 2 \\ 6 \end{bmatrix}, \quad \lambda \in \mathbb{R}$$

- 두 열의 선형 가중 결합으로 $\mathbb{R}^2$의 아무 점이나 도달할 수 없음
  - 사실 두 개의 열은 같은 직선 상에 존재함. 하나의 열은 다른 열과 크기만 다를 뿐임
  - 즉 이 2x2 행렬의 열공간은 여전히 선(1차원 부분공간)
- 행렬에 N 개의 열이 있다고 해서 열공간이 N 차원이 된다는 것을 보장하지 않음
  - 열공간의 차원 수는 열들이 선형 독립 집합을 형성하는 경우에만 열 수와 같음

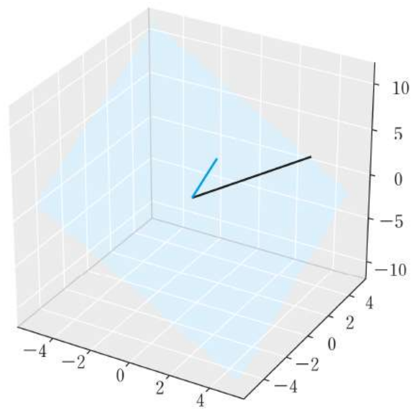


SECTION 05-2 행렬 공간(열, 행, 영)(4)

- 3차원 열공간

$$C\left(\begin{bmatrix} 3 & 0 \\ 5 & 2 \\ 1 & 2 \end{bmatrix}\right) = \lambda_1 \begin{bmatrix} 3 \\ 5 \\ 1 \end{bmatrix} + \lambda_2 \begin{bmatrix} 0 \\ 2 \\ 2 \end{bmatrix}, \quad \lambda \in \mathbb{R}$$

  - $\mathbb{R}^3$에 두 개의 열이 있으며 이 두 열은 선형적으로 독립
    - 즉 하나의 열을 다른 열의 크기를 변경해서 표현할 수 없음
    - 따라서 이 행렬의 열공간은 2차원이지만, $\mathbb{R}^3$에 포함된 2차원 평면(그림 5-3).



3차원에 포함된 행렬의 2차원 열공간
두 개의 두꺼운 선은 행렬의 두 열을 표시함

SECTION 05-2 행렬 공간(열, 행, 영)(5)

5.2.2 행공간

- 행렬의 행공간은 열공간과 완전히 동일한 개념
  - 열 대신 행으로 가능한 모든 가중 결합을 다룸
  - 행공간은 $R(\mathbf{A})$로 나타냄
  - 전치 연산은 행과 열을 바꾸는 것이므로 행렬의 행공간은 전치된 행렬의 열공간, 즉 $R(\mathbf{A}) = C(\mathbf{A}^T)$로 표기
  - 행공간(열공간은 아님)은 행 축소 연산의 변형
  - 행공간이 전치된 행렬의 열공간과 같기 때문에 이 두 행렬 공간은 대칭 행렬일 때 동일

SECTION 05-2 행렬 공간(열, 행, 영)(6)

5.2.3 영공간

- 영공간
  - **A**의 열의 가중 결합이 영벡터가 되는 모든 값이 0이 아닌 가중치 집합
  - 행렬 **A** 에 대해 **Ay** = 0을 만족하는 벡터 **y**는 무한히 존재

$$\mathrm{A}\mathbf{y} = 0$$

$$\begin{bmatrix} 1 & -1 \\ -2 & 2 \end{bmatrix}$$

Ay=0를 만족하는 y 벡터는 [7.34,7.34]. [1,1], [-1,-1] ………무수히 존재

  - 이 모든 벡터는 선택한 벡터의 크기를 조정한 어떠한 형태로 표현할 수 있음

$$N(\mathrm{A}) = \lambda \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \lambda \in \mathbb{R}$$

  - 행렬의 열이 선형 독립 집합이라면 영공간이 비어 있음(빈 집합)

$$\begin{bmatrix} 1 & -1 \\ -2 & 3 \end{bmatrix}$$

Ay=0를 만족하는 y 벡터가 존재하지 않음 N(A) = { }

  - 최대계수와 최대열계수 행렬은 빈 영공간을 가지지만 축소계수 행렬의 영공간은 비어 있지 않음

SECTION 05-2 행렬 공간(열, 행, 영)(7)

- 파이썬 SciPy 라이브러리의 영공간을 계산하는 함수
  - 무한히 가능한 벡터들 중에서 파이썬은 단위(unit) 벡터를 반환
    - import scipy

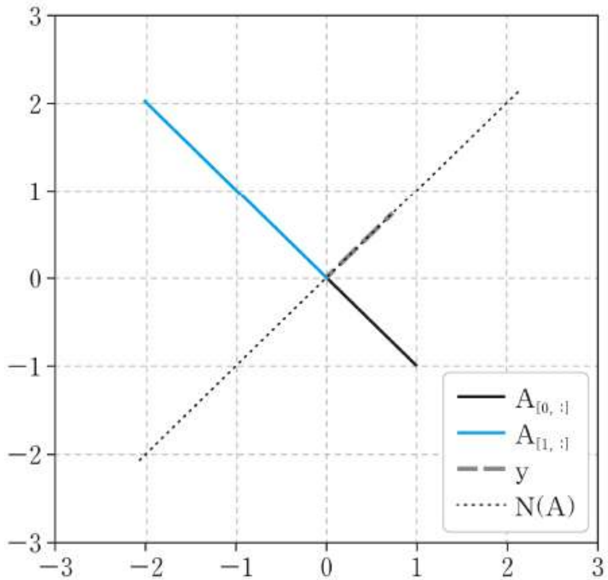
```
A = np.array([ [1,-1],[-2,2] ])
B = np.array([ [1,-1],[-2,3] ])
print( scipy.linalg.null_space(A) )
print( scipy.linalg.null_space(B) )
```

➡

```
[[0.70710678]
 [0.70710678]]

[]
```

SECTION 05-2 행렬 공간(열, 행, 영)(8)

- 행렬 A의 행벡터와 영공간
  - 행렬의 각 행(a_i)에 대해 식을 다시 쓰면 $a_i\mathbf{y} = \mathbf{0}$
  - 즉, 영공간 벡터와 각 행 사이의 내적은 0
- 모든 행렬에는 4개의 연관된 부분공간이 존재
  - 열, 행, 영, 행의 영공간 - 행렬 전치의 영공간

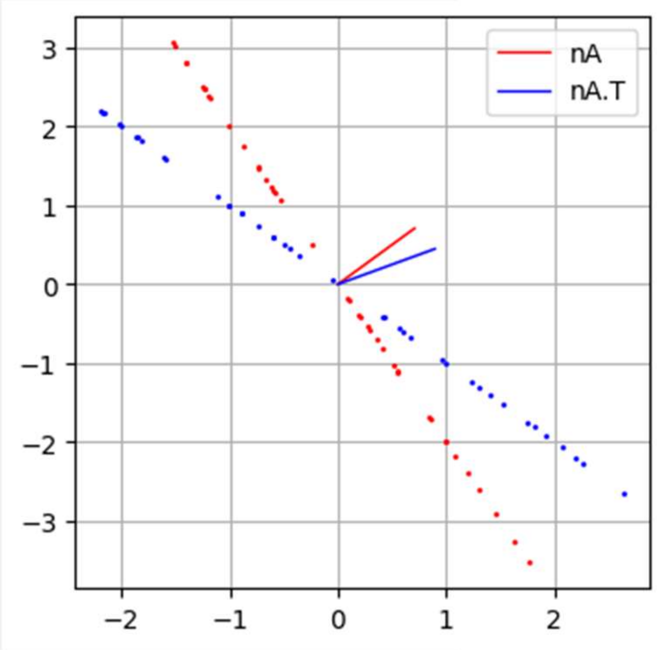


행렬의 영공간 시각화

실습 문제

- 행렬 A의 열, 행, 영, 행의 영공간을 그리기
  - 아래 코드 (수정한 2장 코드)를 참고해 4개의 그래프를 그리기

```
import numpy as np
import scipy
import matplotlib.pyplot as plt

A = np.array([[1,-1],[-2,2]])

# random scalars in that range
scalars = np.random.uniform(low=-1,high=1,size=(40,2))
plt.figure(figsize=(4,4)) # create a figure and size

# 1. 열공간과 행공간을 선형가중결합을 통해 계산
for s in scalars:
  p = A*s # create point p
  plt.plot(p[0], p[1], 'o', color = "red", markersize = 3)

# 2. 열과 행의 영공간을 선으로 그리기
# 선그리기 샘플 코드
# plt.plot([0,A[0,0]],[0,A[1,0]],'k', linewidth=1, color="red", label='A')

plt.legend() #범례 표기
plt.grid()
plt.show()
```



SECTION 05-3 계수(1)

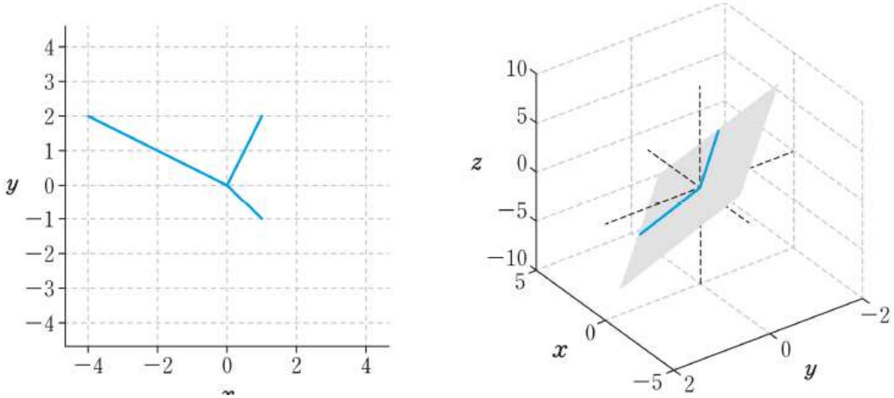
- 계수(rank)
  - 하나의 행렬과 연관된 고유한 숫자
  - 행렬 부분공간의 차원의 수와 관련이 있으며 역행렬이나 방정식의 정답의 수를 결정
  - 계수의 속성
    - 계수는 음이 아닌 정수이므로 0, 1, 2, ..., 이 될 수 있지만 -2 또는 3.14는 될 수 없음
    - 모든 행렬은 하나의 고유한 계수를 가짐
      - 즉 여러 개의 서로 다른 계수를 가질 수 없음(이는 계수가 행이나 열의 특징이 아닌 행렬의 특징임을 나타냄)
    - 행렬의 계수는 r(**A**) 또는 rank(**A**)로 나타내며, '**A** 는 계수-r 행렬'로 읽음
    - 행렬의 최대로 가능한 계수는 행 또는 열의 개수 중에서 더 작은 값, 즉 $\min\{M,N\}$
    - 최대로 가능한 계수를 갖는 행렬을 '최대계수(full rank)' 혹은 전계수
    - 계수가 $r < \min\{M,N\}$인 행렬은 '축소계수', '계수부족' 또는 '특이' 등이라 칭함
    - 스칼라 곱셈은 행렬 계수에 영향을 미치지 않음
      - 0은 예외. 0은 행렬을 계수가 0인 영행렬로 변환

SECTION 05-3 계수(2)

- 행렬 계수에 대한 몇 가지 동일한 해석과 정의
  - 선형 독립 집합을 형성하는 최대 열(또는 행)의 수
  - 열공간의 차원의 수(행공간의 차원의 수와 동일)
  - 행렬 안의 정보를 포함하는 차원 수. 선형 종속적일 가능성이 있으므로 행렬의 전체 열 또는 행 수와 같지 않음
  - 행렬에서 0이 아닌 특이값의 수

SECTION 05-3 계수(3)

- 열공간과 행공간의 차원의 수가 동일한 지의 증명(예시)
  - 행렬의 열공간은 $\mathbb{R}^2$이고 행공간은 $\mathbb{R}^3$
  - 세 열의 집합은 선형 독립은 아니지만 $\mathbb{R}^2$전체를 생성
  - 행렬의 열공간은 2차원 $\begin{bmatrix} 1 & 1 & -4 \\ 2 & -2 & 2 \end{bmatrix}$
  - 두 행은 선형적으로 독립이며 생성되는 부분공간은 $\mathbb{R}^2$의 2차원 평
  - 행렬의 열공간과 행공간이 다르지만 행렬 공간의 차원의 수는 동일
    - 그 차원의 수는 행렬의 계수입니다. 즉 이 행렬의 계수는 2



열공간과 행공간의 생성은 다르지만 차원의 수는 동일함

SECTION 05-3 계수(4)

- [연습] 다음 행렬의 계수를 추정하기

$$A = \begin{bmatrix} 1 \\ 2 \\ 4 \end{bmatrix}, \quad B\begin{bmatrix} 1 & 3 \\ 2 & 6 \\ 4 & 12 \end{bmatrix}, \quad C\begin{bmatrix} 1 & 3.1 \\ 2 & 6 \\ 4 & 12 \end{bmatrix}, \quad D\begin{bmatrix} 1 & 3 & 2 \\ 6 & 6 & 1 \\ 4 & 2 & 0 \end{bmatrix}, \quad E\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \quad F\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

SECTION 05-3 계수(4)

- [연습] 다음 행렬의 계수를 추정하기

$$A = \begin{bmatrix} 1 \\ 2 \\ 4 \end{bmatrix},\quad B\begin{bmatrix} 1 & 3 \\ 2 & 6 \\ 4 & 12 \end{bmatrix},\quad C\begin{bmatrix} 1 & 3.1 \\ 2 & 6 \\ 4 & 12 \end{bmatrix},\quad D\begin{bmatrix} 1 & 3 & 2 \\ 6 & 6 & 1 \\ 4 & 2 & 0 \end{bmatrix},\quad E\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix},\quad F\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

r(A)=1, r(B)=1, r(C)=2, r(D)=3, r(E)=1, r(F)=0

SECTION 05-3 계수(5)

5.3.1 특수 행렬의 계수

- 벡터
  - 모든 벡터의 계수는 1
  - 유일한 예외는 영벡터
- 영행렬
  - 어떤 크기든 영행렬(영벡터 포함)의 계수는 0
- 단위 행렬
  - 단위 행렬의 계수는 행의 수(열의 수)와 같음 - 즉 $r(\mathbf{I}_N)=N$
- 대각 행렬
  - 대각 행렬의 계수는 0이 아닌 대각 원소의 수와 같음
  - 이 속성은 방정식을 풀거나 특이값 분해를 해석할 때 유용하게 사용
- 삼각 행렬
  - 삼각 행렬은 모든 대각선 원소에 0이 아닌 값이 있는 경우에만 최대계수
  - 대각선에 0이 하나 이상 있는 삼각 행렬은 축소계수(정확한 계수는 행렬의 숫자 값에 따라 달라짐)

SECTION 05-3 계수(6)

- 무작위 행렬
  - 무작위 행렬의 계수는 선험적(경험에 앞선 주관적 인식)으로 알 수 없음
    - 행렬의 원소를 도출한 수의 분포와 각 숫자의 도출 확률에 따라 달라지기 때문임
  - 최대로 가능한 계수가 보장되도록 무작위 행렬을 만드는 방법
    - 가우스 또는 균일한 분포 등에서 임의로 부동 소수점을 도출
- 계수-1 행렬
  - 계수-1 행렬의 계수는 당연히 1
  - 즉, 행렬에는 실제로 한 열만 의미 있는 정보를 가지고(또는 한 행의 정보만) 다른 모든 열(또는 행)은 단순히 이 열의 선형 배수임
  - 계수-1 행렬은 정방이거나 높거나 넓을 수 있음
    - 크기에 상관없이 각 열은 첫 번째 열의 크기를 조정한 것 또는 각 행은 첫 번째 행의 크기를 조정한 것

$$\begin{bmatrix} -2 & -4 & -4 \\ -1 & -2 & -2 \\ 0 & 0 & 0 \end{bmatrix}, \quad \begin{bmatrix} 2 & 1 \\ 0 & 0 \\ 2 & 1 \\ 4 & 2 \end{bmatrix}, \quad \begin{bmatrix} 12 & 4 & 4 & 12 & 4 \\ 6 & 2 & 2 & 6 & 2 \\ 9 & 3 & 3 & 9 & 3 \end{bmatrix}$$

SECTION 05-3 계수(7)

5.3.2 덧셈 및 곱셈 행렬의 계수

- 행렬 **A** 와 **B**의 계수를 알면 자동으로 **A**+**B** 또는 **AB**의 계수를 알 수 없지만, 두 개별 행렬의 계수로 **A**+**B** 또는 **AB**가 가질 수 있는 최대로 가능한 계수를 구할 수 있음
  - 덧셈 행렬의 계수는 개별 행렬의 계수보다 클 수 있음
  - 곱셈 행렬의 계수는 개별 행렬의 가장 큰 계수보다 클 수 없음

SECTION 05-3 계수(8)

5.3.3 이동된 행렬의 계수

- 행렬을 이동시키면 보통 최대계수가 됨
  - 정방 행렬을 이동하는 주된 목표는 $r<M$ 에서 $r=M$ 으로 계수를 늘리는 것
  - 영행렬을 단위 행렬로 이동하면 **0**+**I**는 최대계수 행렬
- 행렬 이동 예제

$$\begin{bmatrix} 1 & 3 & 2 \\ 5 & 7 & 2 \\ 2 & 2 & 0 \end{bmatrix} + .01 \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1.01 & 3 & 2 \\ 5 & 7.01 & 2 \\ 2 & 2 & .01 \end{bmatrix}$$

- 가장 왼쪽 행렬의 계수는 2
- 세 번째 열은 두 번째 열에서 첫 번째 열을 뺀 값
- 단위 행렬을 더한 결과 행렬의 계수는 3
- 세 번째 열은 처음 두 열의 선형 결합으로 생성할 수 없음
  - 그러나 행렬이 가진 정보는 거의 변하지 않음
- 기존 계수 -2 행렬은 가역이 아니었지만 이동된 행렬은 가역
- 역행렬(Inverse matrix)의 조건은 정방행렬이면서 full rank여야함

실습 문제

- 행렬 **A** , **A**^{T} , **A**^{T}**A** 및 **AA**^{T} 의 계수는 모두 동일함을 증명
  - A = np.random.randn(15,4) @ np.random.randn(4,8)와 같이 특정 크기의 표준정규분포 난수행렬 생성
  - np.linalg.matrix_rank(A)

SECTION 05-4 계수 응용(1)

5.4.1 열공간에 존재하나요?

- 행렬 확장(augmenting)
  - 행렬의 오른쪽에 열을 추가
  - ‘기저’ $M \times N$ 행렬과 ‘추가’ $M \times K$ 행렬이 있다고 가정
    - 확장된 행렬의 크기는 $M \times (N+K)$가 됨
  - 기본적으로 두 행렬의 행 수가 동일할 때(열 수는 달라도 됨) 확장할 수 있음

$$\begin{bmatrix} 4 & 5 & 6 \\ 0 & 1 & 2 \\ 9 & 9 & 4 \end{bmatrix} \sqcup \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 4 & 5 & 6 & 1 \\ 0 & 1 & 2 & 2 \\ 9 & 9 & 4 & 3 \end{bmatrix}$$

SECTION 05-4 계수 응용(2)

- 벡터가 행렬의 열 공간에 있는지 여부를 확인하는 알고리즘
  - 벡터로 행렬을 확장. 원래 행렬을 벡터 **A**로 확장한 행렬은 $\tilde{\boldsymbol{A}}$
  - 두 행렬의 계수를 계산
  - 두 계수를 비교 - 두 가지 가능한 결과

  a. rank(**A**) = rank($\tilde{\boldsymbol{A}}$) 벡터는 행렬 A 의 열공간에 있음

  b. rank(**A**) < rank($\tilde{\boldsymbol{A}}$) 벡터는 행렬 A 의 열공간에 없음

- 알고리즘 해석
  - **v**∈C(**A**)라면 **v** 는 **A** 의 열들의 선형 가중 결합으로 나타낼 수 있다는 의미
    - 즉 확장 행렬 $\tilde{\boldsymbol{A}}$ 의 열들은 선형 종속 집합
    - 생성 측면에서 벡터 **v** 는 $\tilde{\boldsymbol{A}}$ 에서 중복이며, 따라서 계수는 그대로 유지
  - 반대로 **v** ∉ C(**A**)라면 **v** 는 **A** 의 열들의 선형 가중 결합으로 나타낼 수 없으며 이는 **v**가 $\tilde{\boldsymbol{A}}$ 에 새로운 정보를 추가했으며, 그 계수가 1 증가할 것이라는 것을 의미
- 설계 행렬(design matrix)
  - 세상이 어떻게 돌아가는지에 대한 모델을 개발하고, 그 모델을 행렬로 변환

SECTION 05-5 행렬식(1)

■ 행렬식(determinant)의 특성

1) 정방 행렬에 대해서만 정의
2) 특이(축소계수) 행렬에 대해서는 0
   - 행렬식은 *det*(**A**) 또는|**A**|로 표기(행렬 노름을 나타내는 이중 수직선이 아닌 단일 수직선)
   - 그리스어 대문자 델타 Δ는 특정 행렬을 언급하는 것이 아닐 때 주로 사용

SECTION 05-5 행렬식(2)

5.5.1 행렬식 계산

- 2 × 2 행렬식의 계산

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

- 3 × 3 행렬식의 계산

$$\begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} = aei + bfg + cdh - ceg - bdi - afh$$

- 4 × 4 행렬식의 계산

$$\begin{vmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{vmatrix} = \begin{array}{c} afkp - aflo - agjp + agln + ahjo - ahkn - bekp + belo \\ + bgip - bglm - bhio + bhkm + cejp - celn - cfip + cflm \\ + chin - chjm - dejo + dekn + dfio - dfkm - dgin + dgjm \end{array}$$

파이썬의 행렬식 계산 함수 np.linalg.det() 또는 scipy.linalg.det()

5.5.2 선형 종속성과 행렬식

- 모든 축소계수 행렬의 행렬식은 0
  - 따라서 r<*M* 이라면 Δ=0
  - 이와 반대로 모든 최대계수 행렬의 행렬식은 0이 아님

$$\begin{vmatrix} a & \lambda a \\ c & \lambda c \end{vmatrix} = ac\lambda - a\lambda c = 0$$

  - 3 × 3 특이 행렬의 행렬식

$$\begin{vmatrix} a & b & \lambda a \\ d & e & \lambda d \\ g & h & \lambda g \end{vmatrix} = ae\lambda g + b\lambda dg + \lambda adh - \lambda aeg - bd\lambda g - a\lambda dh = 0$$

SECTION 05-5 행렬식(4)

5.5.3 특성 다항식

- 2 × 2 행렬의 행렬식을 계산하는 방정식에는 네 가지 원소와 행렬식, 총 다섯 개의 수가 존재
  - ad - bc = Δ

행렬식을 사용하여 미지의 행렬 원소를 계산

$$\begin{vmatrix} a & b \\ c & \lambda \end{vmatrix} \Rightarrow a\lambda - bc = \Delta$$

$$\begin{vmatrix} 2 & 7 \\ 4 & \lambda \end{vmatrix} = 4 \Rightarrow 2\lambda - 28 = 4 \Rightarrow 2\lambda = 32 \Rightarrow \lambda = 16$$

- 2차 다항식의 계산

$$\begin{vmatrix} \lambda & 1 \\ 3 & \lambda \end{vmatrix} = 1 \Rightarrow \lambda^2 - 3 = 1 \Rightarrow \lambda^2 = 4 \Rightarrow \lambda = \pm 2$$

SECTION 05-5 행렬식(5)

- 특성 다항식(characteristics polynomial)
  - 행렬 이동과 행렬식을 결합

    행렬의 특성 다항식 $det(\mathrm{A} - \lambda \mathrm{I}) = \Delta$

  - 이동된 M×M 행렬은 λ^M 항을 가지며 따라서 M 개의 해가 존재

2x2 행렬 $$\begin{vmatrix} a-\lambda & b \\ c & d-\lambda \end{vmatrix} = 0 \Rightarrow \lambda^2 - (a+d)\lambda + (ad-bc) = 0$$

3x3 행렬 $$\begin{vmatrix} a-\lambda & b & c \\ d & e-\lambda & f \\ g & h & i-\lambda \end{vmatrix} = 0 \Rightarrow \begin{array}{c} -\lambda^3 + (a+e+i)\lambda^2 \\ + (-ae+bd-ai+cg-ei+fh)\lambda \\ + aei - afh - bdi + bfg + cdh - ceg = 0 \end{array}$$

SECTION 05-5 행렬식(6)

- 특성 다항식 사용
  - 아래 행렬은 최대계수이므로 행렬식은 0이 아님(Δ = -8)
  - 어떤 스칼라만큼 이동된 후에 행렬식이 0이라고 가정
  - 행렬식을 축소계수로 만드는 값 찾기

$$\det\left(\begin{bmatrix} 1 & 3 \\ 3 & 1 \end{bmatrix} - \lambda \mathrm{I}\right) = 0$$

$$\begin{vmatrix} 1-\lambda & 3 \\ 3 & 1-\lambda \end{vmatrix} = 0 \quad \Rightarrow \quad (1-\lambda)^2 - 9 = 0$$

  - 두 해는 λ = -2, 4

$$\lambda = -2 \quad \Rightarrow \begin{bmatrix} 3 & 3 \\ 3 & 3 \end{bmatrix} \qquad\qquad \lambda = 4 \quad \Rightarrow \begin{bmatrix} -3 & 3 \\ 3 & -3 \end{bmatrix}$$

- ❑ 두 행렬 모두 계수가 1
- ❑ 또한 두 행렬 모두 단순하지 않은 영공간을 가짐
- ❑ 즉, (**A** – λ**I**)**y** = 0를 만족하는 0이 아닌 벡터 **y**를 찾을 수 있음

실습

■ 축소계수 행렬의 행렬식은 이론적으로 0임을 실험

```
ns = np.arange(3,31) # matrix sizes

iters = 100 # iteration

# initialize
dets = np.zeros((len(ns),iters))

# loop over matrix sizes
for ni in range(len(ns)):
  for i in range(iters):
    A = np.random.randn(ns[ni],ns[ni]) #난수 행렬 생성
    A[:,0] = A[:,1] # 계수 1 감소
    dets[ni,i]=np.abs(np.linalg.det(A)) # 행렬식 계산->절대값

plt.figure(figsize=(6,4))
plt.plot(ns**2,np.log(np.mean(dets,axis=1)),'ks-',linewidth=3)
plt.xlabel('Number of elements in the matrix')
plt.ylabel('Log determinant')
plt.title('Empirical determinants of singular matrices')
plt.show()
```