순천향대학교
SOON CHUN HYANG UNIVERSITY

2024년 2학기

선형대수

7장

Chapter 07

- SECTION 07-1 역행렬
- SECTION 07-2 역행렬의 유형과 가역성의 조건
- SECTION 07-3 역행렬 계산
- SECTION 07-4 역행렬의 유일성
- SECTION 07-5 무어-펜로즈 의사역행렬
- SECTION 07-6 역행렬의 수치적 안정성
- SECTION 07-7 역행렬의 기하학적 해석

SCH

SECTION 07-1 역행렬

■ 역행렬

- 선형 변환을 포함하며 행렬 곱셈은 이 변환을 적용하는 매커니즘
- 행렬 **A**의 역행렬
  - **A**와 곱해서 단위 행렬을 만드는 행렬 $\mathbf{A}^{-1}$ ($\mathbf{A}\mathbf{A}^{-1} = I$)
  - 행렬을 '취소(cancel)'
  - 행렬을 단위 행렬로 선형 변환

$$
\begin{aligned}
Ax &= b \\
A^{-1}Ax &= A^{-1}b \\
Ix &= A^{-1}b \\
x &= A^{-1}b
\end{aligned}
$$

SECTION 07-2 역행렬의 유형과 가역성의 조건

완전 역행렬

- $\mathbf{A}^{-1}\mathbf{A} = \mathbf{A}\mathbf{A}^{-1} = \mathbf{I}$
- 행렬이 완전 역행렬을 가지려면 (1)정방이면서 (2)최대계수이어야 함
- 모든 정방 최대계수 행렬은 역행렬을 가지고, 완전 역행렬을 갖는 모든 행렬은 정방이며 최대계수

단방향 역행렬

- 단방향 역행렬은 정방이 아닌 행렬을 단위 행렬로 변환할 수 있지만 하나의 곱셈 방향에 대해서만 동작
- 높은 행렬 **T** 는 왼쪽 역행렬을 가질 수 있음
  - **LT** = **I**이지만 **TL** ≠ **I**
- 넓은 행렬 W는 오른쪽 역행렬을 가질 수 있음
  - **WR** = **I**이지만, **RW** ≠ **I**
- 정방이 아닌 행렬은 최대로 가능한 계수일 때만 단방향 역행렬을 가짐

의사역행렬

- 모든 행렬에는 모양과 계수에 관계없이 의사역행렬(pseudo inverse)이 존재
- 완전 역행렬 또는 단방향 역행렬이 존재하지 않는 행렬을 특이 또는 비가역 행렬이라고 함
  - 이는 행렬에 축소계수(reduced-rank) 또는 계수부족(rank-deficient)이라고 지칭하는 것과 같음

SECTION 07-3 역행렬 계산(1)

7.3.1 2×2 행렬의 역행렬

- 2 × 2 행렬의 역 계산
  - 대각 원소를 교환하고, 대각이 아닌 원소에 -1을 곱한 다음, 행렬식으로 나눠줌
  - 이 알고리즘으로 원래 행렬을 단위 행렬로 변환하는 역행렬을 생성

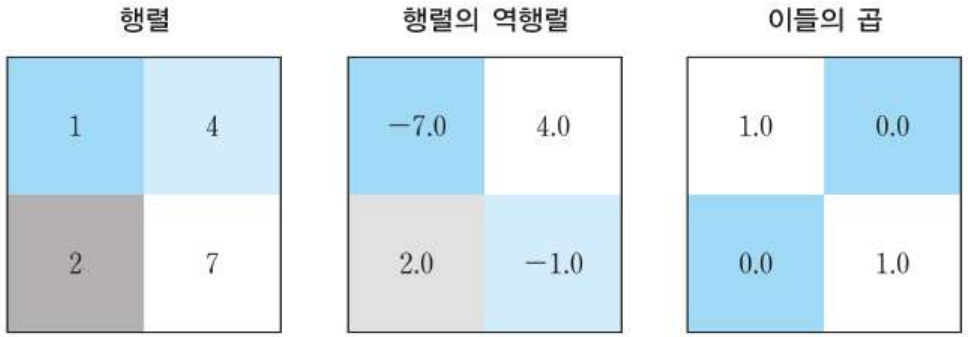
$$
\begin{aligned}
\mathrm{A} &= \begin{bmatrix} a & b \\ c & d \end{bmatrix} \\
\mathrm{A}^{-1} &= \frac{1}{ad-bc}\begin{bmatrix} d & -b \\ -c & a \end{bmatrix} \\
\mathrm{A}\mathrm{A}^{-1} &= \begin{bmatrix} a & b \\ c & d \end{bmatrix}\frac{1}{ad-bc}\begin{bmatrix} d & -b \\ -c & a \end{bmatrix} \\
&= \frac{1}{ad-bc}\begin{bmatrix} ad-bc & 0 \\ 0 & ad-bc \end{bmatrix} \\
&= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}
\end{aligned}
$$

$$
\begin{bmatrix} 1 & 4 \\ 2 & 7 \end{bmatrix}\begin{bmatrix} 7 & -4 \\ -2 & 1 \end{bmatrix}\frac{1}{7-8} = \begin{bmatrix} (7-8) & (-4+4) \\ (14-14) & (-8+7) \end{bmatrix}\frac{1}{-1} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}
$$

SECTION 07-3 역행렬 계산(2)

- 파이썬으로 행렬의 역 계산
  - A@Ainv를 통해 단위 행렬이 생성되는지 확인
    - Ainv@A도 마찬가지
    - A*Ainv로는 단위 행렬이 생성되지 않음 (*가 아다마르(원소별) 곱셈이기 때문임)

```
A = np.array([ [1,4],[2,7] ])
Ainv = np.linalg.inv(A)
A@Ainv
```



2×2 행렬의 역행렬

SECTION 07-3 역행렬 계산(3)

- 축소계수 행렬은 역행렬을 가지지 않음

$$\begin{bmatrix}1 & 4\\ 2 & 8\end{bmatrix}\begin{bmatrix}8 & -4\\ -2 & 1\end{bmatrix}\frac{1}{0}=\begin{bmatrix}(8-8) & (-4+4)\\ (16-16) & (-8+8)\end{bmatrix}\frac{1}{0}=???$$

  - 행렬 곱셈의 결과가 ΔI 가 아닌 0
  - 행렬식이 0
  - 이것은 축소계수 행렬(계수 = 1)
    - 즉 축소계수 행렬은 비가역적

  - 파이썬 코드와 에러가 발생한 결과

```
A = np.array([ [1,4],[2,8] ])
Ainv = np.linalg.inv(A)
A@Ainv
```

⇨ `LinAlgError: 특이 행렬`

SECTION 07-3 역행렬 계산(4)

7.3.2 대각 행렬의 역행렬

- 정방 대각 행렬의 역행렬을 계산하는 쉬운 방법
  - 핵심은 두 대각행렬의 곱이 대응하는 대각 원소의 스칼라 곱셈이라는 점
  - 대각선에 0이 있는 대각 행렬이라면 어떻게 될까?
    - 1/0이 되므로 역을 구할 수 없음
    - 대각선에 0이 하나 이상 있는 대각 행렬은 역행렬이 존재하지 않음

$$\begin{bmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 4 \end{bmatrix} \begin{bmatrix} b & 0 & 0 \\ 0 & c & 0 \\ 0 & 0 & d \end{bmatrix} = \begin{bmatrix} 2b & 0 & 0 \\ 0 & 3c & 0 \\ 0 & 0 & 4d \end{bmatrix}$$

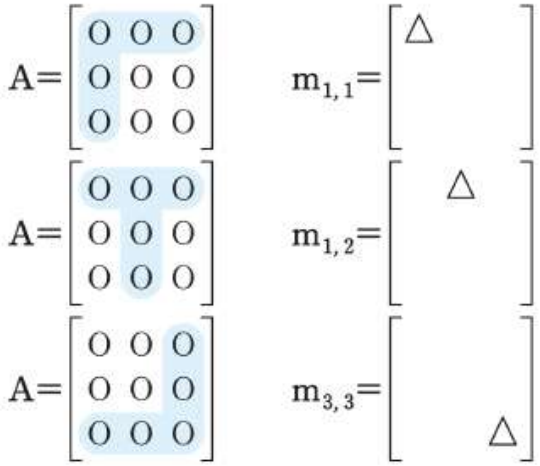
SECTION 07-3 역행렬 계산(5)

7.3.3 임의의 정방 최대계수 행렬의 역행렬

■ 역행렬을 계산하는 알고리즘

소행렬(minors)

- 부분행렬의 행렬식으로 구성
- 소행렬의 각 원소 $m_{i,j}$는 i번째 행과 j번째 열을 제외하고 만든 부분행렬의 행렬식



소행렬 계산(강조색 음영 영역을 제거해서 각 부분행렬을 생성)

SECTION 07-3 역행렬 계산(5)

7.3.3 임의의 정방 최대계수 행렬의 역행렬

- 역행렬을 계산하는 알고리즘

  소행렬(minors)

```
def matrix_minor(arr, i, j):
    return np.delete(np.delete(arr,i,axis=0), j, axis=1)

# test
A = np.arange(1,10).reshape(3,3)
print(A)
A0 = matrix_minor(A, 1, 1)
A1 = np.delete(np.delete(A, 1, axis=0), 1, axis=1)
A0, A1
```

SECTION 07-3 역행렬 계산(6)

격자행렬(grid)

- +1과 -1를 교대로 사용하는 체스판
- 파이썬에서 이 공식을 구현할 때 색인화와 지수화를 주의

$$g_{i,j} = -1^{i+j}$$

여인수행렬(cofactors)

- 소행렬과 격자행렬의 아다마르곱의 결과로 생성

수반행렬(adjugate)

- 여인수행렬의 전치
- 원래 행렬(여인수행렬이 아니라 역행렬을 구하려는 행렬)의 행렬식의 역수를 스칼라 곱셈

SECTION 07-3 역행렬 계산(8)

7.3.4 단방향 역행렬

- 높은 행렬은 완전 역행렬을 가지지 않음
  - 즉 크기가 $M > N$인 행렬 **T** 에 대해서 $\mathbf{TT}^{-1} = \mathbf{T}^{-1}\mathbf{T} = \mathbf{I}$ 를 만족하는 높은 행렬 $\mathbf{T}^{-1}$은 존재하지 않음
- 하지만 LT = **I** 를 만족하는 행렬 **L**은 존재
  - 왼쪽의 **L** 행렬을 찾기
    - 먼저 행렬 **T** 를 정방으로 만듦 - 전치를 곱
  - $\mathbf{T}^{T}\mathbf{T}$ 와 $\mathbf{TT}^{T}$ 둘 다 정방이 되지만 만약 **T**가 최대열계수라면 $\mathbf{T}^{T}\mathbf{T}$ 는 최대계수가 됨
- 파이썬으로 높은 행렬과 그 행렬의 전치를 곱한 행렬은 완전 역행렬을 가진다는 사실을 구현

```
T = np.random.randint(-10,11,size=(40,4))
TtT = T.T @ T
TtT_inv = np.linalg.inv(TtT)
TtT_inv @ TtT
```

$$(\mathrm{T}^{\mathrm{T}}\mathrm{T})^{-1}(\mathrm{T}^{\mathrm{T}}\mathrm{T}) = \mathrm{I}$$

- **코드와 공식을 통해 $\mathbf{T}^{T}\mathbf{T}$ 가 T 와 같은 행렬이 아니기 때문에 $(\mathbf{T}^{T}\mathbf{T})^{-1}$은 T 의 역행렬이 아님을 알 수 있음**

SECTION 07-3 역행렬 계산(9)

- 행렬 L 은 행렬 T 의 왼쪽 역행렬

$$(T^T T)^{-1}(T^T T) = I \qquad L = (T^T T)^{-1} T^T$$
$$LT = I$$

- 파이썬 코드로 왼쪽 역행렬 계산

```
L = TtT_inv @ T.T # 왼쪽 역행렬
L@T # 단위 행렬을 생성
```

  - 원래의 높은 행렬의 왼쪽으로 곱해서 단위 행렬을 생성
  - 파이썬에서 **TL** (즉 왼쪽 역행렬을 오른쪽에서 곱함)이 단위 행렬이 아님을 확인
  - 그러므로 왼쪽 역행렬은 단방향

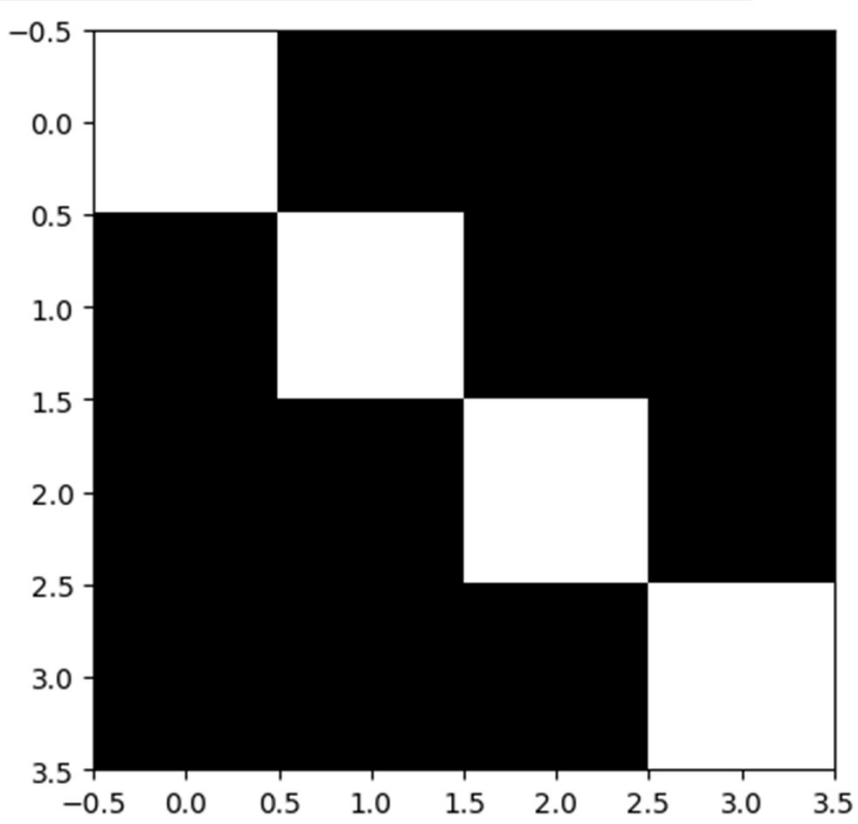
SECTION 07-3 역행렬 계산(9)

```
import matplotlib.pyplot as plt
import matplotlib.cm as cm

T = np.random.randint(-10,11,size=(40,4))
TtT = T.T @ T
TtT_inv = np.linalg.inv(TtT)

LT = TtT_inv @ T.T @ T

plt.imshow(LT, cmap = cm.gray)
plt.show()
```



SECTION 07-3 역행렬 계산(10)

- 왼쪽 역행렬은 최대열계수를 갖는 높은 행렬에 대해서만 정의
  - 크기가 $M > N$ 이고 행렬의 계수가 $r < N$ 인 행렬에는 왼쪽 역행렬이 없음
    - $\mathbf{T}^T\mathbf{T}$는 축소 행렬이므로 비가역

SECTION 07-4 역행렬의 유일성

- 역행렬은 유일함
  - 즉 역행렬이 있다면 정확히 하나만 존재
  - **AB** = **I** 와 **AC** = **I** 일 때 **B** ≠ **C** 인 두 개의 행렬 **B**와 **C**는 존재할 수 없음

- 부정에 의한 증명(proof by negation)

[가정]

1) 행렬 **A**는 가역적
2) 행렬 **B** 와 **C** 는 **A**의 역행렬
3) 행렬 **B** 와 **C** 는 서로 다름 (즉 **B** ≠ **C**)

[가정에 따라 계산]

$$C = CI = CAB = IB = B$$

[결론]

- 모든 식이 동일하므로 첫 번째와 마지막 식은 동일하며 이는 **B** ≠ **C** 에 대한 가정이 거짓임을 의미
- 동일한 행렬의 역행렬은 고유함

SECTION 07-5 무어-펜로즈 의사역행렬

무어-펜로즈(Moore-Penrose, MP) 의사역행렬

- 의사역행렬은 위첨자 안에 단검, 더하기 기호 또는 별표를 사용하여 표시
  - $\mathbf{A}^{\dagger}$, $\mathbf{A}^{+}$, $\mathbf{A}^{*}$

- 파이썬에서 의사역행렬을 구하는 함수 np.linalg.pinv
  - 다음 코드는 np.linalg.inv가 오류를 출력한 특이 행렬의 의사역행렬을 계산

```
A   = np.array([[1,4],[2,8]])
A_pinv = np.linalg.pinv(A)
A_pinv
#1 AA*A = A
print(A, A @ A_pinv @ A)
#2 A*AA* = A*
print(A_pinv, A_pinv @ A @ A_pinv)
```

SECTION 07-6 역행렬의 수치적 안정성(1)

- 행렬의 역행렬을 계산하려면 많은 행렬식을 포함해 상당한 FLOP(부동 소수점 연산)이 수반
  - 많은 행렬식을 계산하면 수치적 부정확성이 발생할 수 있음
  - 부정확성이 누적되면 큰 행렬을 연산할 때 심각한 문제 발생
- 수치 연산을 구현하는 저수준 라이브러리(예를 들어 LAPACK)
  - 가능하면 행렬의 역행렬을 명확하게 구하려고 애쓰지 않고 수치적으로 더 안정적인 다른 행렬의 곱으로 분해
- 수치값이 대체로 같은 범위에 있는 행렬이 더 안정적인 경향
  - '수차값의 범위'는 행렬의 조건수를 이용해 표현
    - 조건수는 가장 큰 값과 가장 작은 특이값의 비율

SECTION 07-6 역행렬의 수치적 안정성(2)

■ 수치적으로 불안정한 행렬의 예 - 힐버트 행렬

힐버트 행렬을 생성하는 공식. i와 j는 행과 열 인덱스 $h_{i,j} = \frac{1}{i+j-1}$

▪ 3 × 3 힐버트 행렬의 예

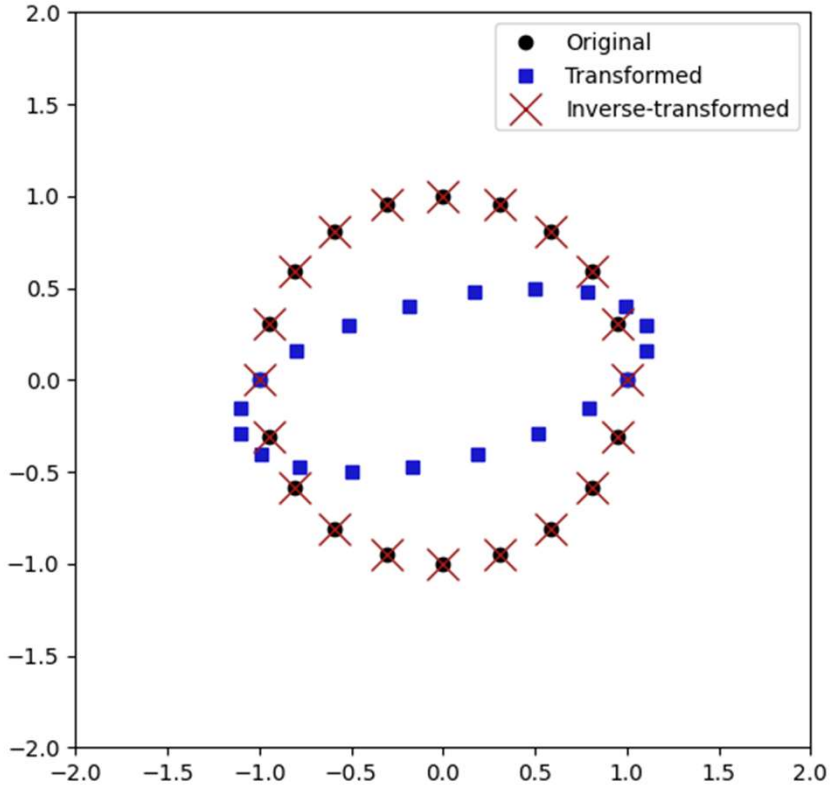
$$\begin{bmatrix} 1 & 1/2 & 1/3 \\ 1/2 & 1/3 & 1/4 \\ 1/3 & 1/4 & 1/5 \end{bmatrix}$$

- 행렬이 커질수록 수치값의 범위가 늘어남
- 결과적으로 컴퓨터로 계산된 힐버트 행렬의 계수는 급격히 낮아짐
- 최대계수의 힐버트 행렬은 전혀 다른 수치 범위에서 역행렬을 가짐
- 곱 행렬이 보기에는 단위 행렬처럼 보이지만, 행렬의 크기에 따라 반올림 오차가 급격히 증가

힐버트 행렬과 역행렬 그리고 그 곱

SECTION 07-7 역행렬의 기하학적 해석

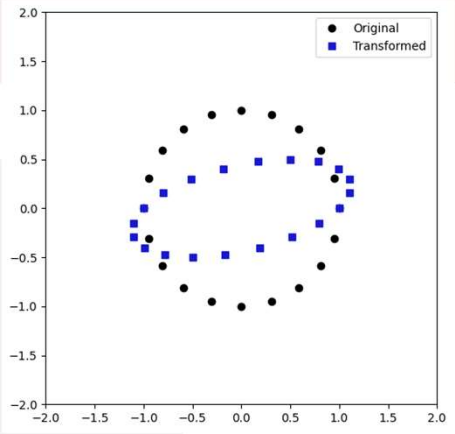
- 역행렬은 행렬 곱셈으로 인한 기하학적 변환을 되돌리는 것
  - 변환된 기하학적 좌표에 변환 행렬의 역행렬을 곱하기

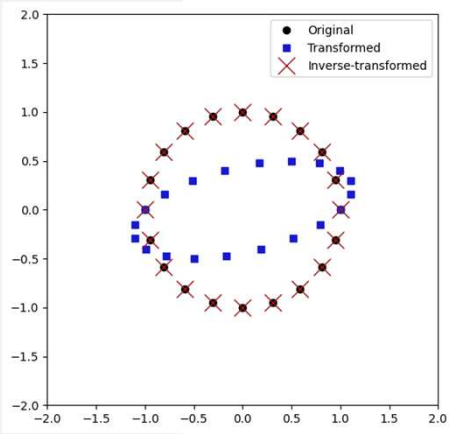


$$Q = TP$$
$$U = T^{-1}Q$$
$$U = T^{-1}TP$$

- P - 최초 기하학적 좌표의 2 x N 행렬
- T - 변환 행렬
- Q - 변환된 좌표의 행렬,
- U - 역변환 좌표의 행렬

실습 문제






```
T = np.array([[1,.5],[0,.5]]) # Transformation matrix

# define the set of points (a circle)
theta = np.linspace(0,2*np.pi-2*np.pi/20,20)
origPoints = np.vstack( (np.cos(theta),np.sin(theta)) )

# apply transformation
transformedPoints = T @ origPoints

# plot the points
plt.figure(figsize=(6,6))
plt.plot(origPoints[0,:],origPoints[1,:],'ko',label='Original')
plt.plot(transformedPoints[0,:],transformedPoints[1,:],'s', color=[.1, .1,
.8],label='Transformed')
# plt.plot(backTransformed[0,:],backTransformed[1,:],'rx',markersize=15,
color=[.6,.1,.1],label='Inverse-transformed')

plt.axis('square')
plt.xlim([-2,2])
plt.ylim([-2,2])
plt.legend()
plt.savefig('Figure_07_06.png',dpi=300)
plt.show()
```