순천향대학교
SOON CHUN HYANG UNIVERSITY

2024년 2학기

선형대수

9장

Chapter 09

- SECTION 09-1 연립방정식
- SECTION 09-2 행 축소
- SECTION 09-3 LU 분해

SCH

SECTION 09-1 연립방정식(1)

- LU 분해와 그 응용을 이해하려면 행 축소와 가우스 소거법의 이해가 필요
  - 방정식을 다루는 방법
  - 행렬 방정식으로 변환하는 방법
  - 행 축소를 사용해 행렬 방정식을 푸는 방법

- 연립방정식

$$\begin{aligned} x &= 4 - y \\ y &= x/2 + 2 \end{aligned} \quad \Rightarrow \quad \begin{aligned} x + (2y) &= 4 - y + (x + 4) \\ y - (x) &= x/2 + 2 - (4 - y) \end{aligned} \quad \Rightarrow \quad x = 4/3, \; y = 8/3$$

  - 각각의 방정식에서 x와 y를 소거하여 해를 구함

9.1.1 연립방정식을 행렬로 변환하기

- 방정식을 행렬로 변환하는 단계

1) 상수가 방정식의 오른쪽에 오도록 방정식을 정리
   - 상수(constan)t는 변수에 결합되지 않은 숫자
     - 절편(intercept) 또는 변위(offset)
   - 변수와 그 곱셈 계수는 순서대로 방정식의 왼쪽에 위치

$$\begin{aligned} x &= 4 - y \\ y &= x/2 + 2 \end{aligned} \quad \Rightarrow \quad \begin{aligned} x + y &= 4 \\ -x/2 + y &= 2 \end{aligned}$$

2) 계수(변수에 곱한 숫자, 방정식에서 누락된 변수는 계수가 0이 됨)를 행렬로 분리
   - 각 방정식의 계수는 행이 됨
   - 변수는 계수 행렬의 오른쪽에 곱하는 열벡터에 놓임
   - 상수는 방정식의 우변에 열벡터로 만듦

$$\begin{bmatrix} 1 & 1 \\ -1/2 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 4 \\ 2 \end{bmatrix}$$

SECTION 09-1 연립방정식(3)

9.1.2 행렬 방정식 다루기

- 행렬 방정식 조작과 유효성

$$Ax = b$$
$$v + Ax = v + b$$
$$(v + Ax)^T = (v + b)^T$$

유효함

$$AX = B$$
$$CAX = CB$$

유효함

$$AX = B$$
$$AXC = CB$$

유효하지 않음

SECTION 09-1 연립방정식(4)

- 파이썬 예제
  - **AX** = **B** 방정식에서 미지의 행렬 **X**를 구하고 난수 행렬 **A**와 **B**를 생성
  - 행렬 곱셈에서 교환이 가능하다면(순서는 중요하지 않다는 의미), res1과 res2가 모두 영행렬이 되는 지 확인

```
A = np.random.randn(4,4)
B = np.random.randn(4,4)

X1 = np.linalg.inv(A) @ B
X2 = B @ np.linalg.inv(A)

res1 = A@X1 - B
res2 = A@X2 - B

np.around(res1, 5), np.around(res2, 5)
```

```
(array([[-0., -0., -0.,  0.],
        [ 0.,  0., -0.,  0.],
        [ 0.,  0., -0.,  0.],
        [-0., -0.,  0., -0.]]),
 array([[  0.06693,   0.84074,   9.95004, -12.94844],
        [ -5.45991,  -1.66853,  -0.15106,   3.83862],
        [  0.86702,  -0.18981,   6.97818,  -9.08718],
        [  2.15482,   2.37426,   4.13814,  -5.37658]]))
```

SECTION 09-2 행 축소(1)

- 행 축소(row reduction)
  - 행렬의 행에 스칼라 곱셈과 덧셈이라는 두 가지 연산을 반복적으로 적용하는 작업
  - 행 축소는 하나의 연립방정식 내에서 하나의 방정식을 다른 방정식에 더하는 것과 동일한 원리를 사용
  - **행 축소의 목표는 밀집 행렬을 상삼각 행렬로 변환하는 것**
  - 밀집 행렬(dense matrix)에서 첫 번째 행을 두 번째 행에 더해 -1을 제거
  - 상삼각 행렬로 변환

$$\begin{bmatrix} 2 & 3 \\ -2 & 2 \end{bmatrix} \xrightarrow{R_1 + R_2} \begin{bmatrix} 2 & 3 \\ 0 & 5 \end{bmatrix}$$

- 행렬의 사다리꼴 형태 - 행 축소의 결과인 상삼각 행렬
  - (1) 각 행에서 가장 왼쪽에 있는 0이 아닌 숫자(기준 원소pivot)가 위행의 기준 원소 오른쪽에 있고,
  - (2) 모든 원소가 0인 행은 0이 아닌 원소를 포함한 행 아래에 있음
- 3 × 3 행렬을 사다리꼴 형태로 변환하려면 두 단계가 필요

$$\begin{bmatrix} 1 & 2 & 2 \\ -1 & 3 & 0 \\ 2 & 4 & -3 \end{bmatrix} \xrightarrow{-2R_1 + R_3} \begin{bmatrix} 1 & 2 & 2 \\ -1 & 3 & 0 \\ 0 & 0 & -7 \end{bmatrix} \xrightarrow{R_1 + R_2} \begin{bmatrix} 1 & 2 & 2 \\ 0 & 5 & 2 \\ 0 & 0 & -7 \end{bmatrix}$$

9.2.1 가우스 소거법

■ 가우스 소거법(Gaussian elimination)

1) 행렬 방정식으로 변환

$$\begin{aligned} x &= 4 - y \\ y &= x/2 + 2 \end{aligned} \Rightarrow \begin{bmatrix} 1 & 1 \\ -1/2 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 4 \\ 2 \end{bmatrix}$$

2) 계수 행렬을 상수벡터로 증강

$$\Downarrow$$

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

3) 행을 사다리꼴 형태로 축소

$$\Downarrow$$

$$\begin{bmatrix} 1 & 1 & 4 \\ -1/2 & 1 & 2 \end{bmatrix} \xrightarrow{1/2R_1 + R_2} \begin{bmatrix} 1 & 1 & 4 \\ 0 & 3/2 & 4 \end{bmatrix}$$

$$\Downarrow$$

4) 역치환(back substitution)을 사용하여 각 변수를 차례로 풀어냄

$$\begin{aligned} x + y &= 4 \\ 3/2y &= 4 \end{aligned}$$

SECTION 09-2 행 축소(3)

9.2.2 가우스-조던 소거법

■ 가우스-조던 소거법(Gaussian-Jordan Elimination)

- 연립방정식에서 서로 얽혀 있는 변수를 분리하고 각 변수에 대한 해를 바로 드러내므로 더 이상 역치환이나 기본적인 산술 조차도 필요하지 않음

1) 모든 기준 원소(각 행의 가장 왼쪽에 있는 0이 아닌 숫자)를 1로 바꾸기 위해 예제 행렬의 행을 계속 축소
2) 각 기준 원소 위의 모든 원소를 제거하기 위해 행을 위쪽으로 축소
   - RREF는 항상 원래 행렬의 왼쪽 상단에 단위 행렬을 부분 행렬로 생성
3) 행렬을 다시 연립방정식으로 변환하여 가우스 소거법을 계속 진행

$$\begin{bmatrix} 1 & 1 & 4 \\ 0 & 3/2 & 4 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 1 & 4 \\ 0 & 1 & 8/3 \end{bmatrix}$$

$$\Downarrow$$

$$\begin{bmatrix} 1 & 1 & 4 \\ 0 & 1 & 8/3 \end{bmatrix} \xrightarrow{-R_2 + R_1} \begin{bmatrix} 1 & 0 & 4/3 \\ 0 & 1 & 8/3 \end{bmatrix}$$

- **기약 행 사다리꼴 형태(Reduced Row Echelon Form, RREF)**

$$\Downarrow$$

$$x = 4/3$$
$$y = 8/3$$

SECTION 09-2 행 축소(3)

- 기약 행 사다리꼴 형태(Reduced Row Echelon Form, RREF)
  - 0이 아닌 원소를 갖는 행에서 맨 처음 나오는 0이 아닌 수는 1이어야 함. 이것을 선도 1(leading one)이라고 함
  - 모든 원소가 0인 행은 행렬의 맨 아래로 내려가야 함
  - 0이 아닌 원소를 갖는 연속된 두 행은 해당 행의 leading 1이 위 행의 leading 1보다 오른쪽에 있어야 함
  - leading 1이 있는 열의 나머지 원소들은 모두 0이어야 함

RREF: $\begin{bmatrix} 1 & 0 & 0 & 5 \\ 0 & 1 & 0 & 7 \\ 0 & 0 & 1 & 8 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 1 & 2 & 0 & 3 \\ 0 & 0 & 0 & 1 & -8 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$

REF: $\begin{bmatrix} 1 & 5 & 8 & -6 \\ 0 & 1 & 4 & 2 \\ 0 & 0 & 1 & 5 \end{bmatrix}, \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 1 & 6 & 3 & 0 \\ 0 & 0 & 1 & -9 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$

SECTION 09-2 행 축소(4)

- RREF는 고유하므로 행렬마다 정확히 하나의 RREF만 존재
- 행렬의 RREF를 계산하는 함수는 NumPy에 없고 sympy 라이브러리에 있음

```
import sympy as sym

# sympy로 행렬 변환
M = np.array([ [1,1,4],[-1/2,1,2] ])
symMat = sym.Matrix(M)

# RREF
symMat.rref()[0]
```

```
>>
  [[1, 0, 1.33333333333333],
   [0, 1, 2.66666666666667]]
```

실습 문제

- 다음 연립방정식의 해를 RREF를 사용해 구하세요.
  - $3x + 4y - 5z = 11$
  - $x + 2y - 10z = 1$
  - $2x + 9y - 5z = 13$

SECTION 09-3 LU 분해(1)

LU 분해

- LU는 하삼각, 상삼각에서와 같이 '아래(lower) 위(upper)'를 의미
- 행렬을 두 개의 삼각 행렬의 곱으로 분해하는 역행렬 계산

$$A = LU$$

- L의 대각성분이 1이라는 전제하에 LU분해가 고유한 것을 보장(full rank)

$$\begin{bmatrix} 2 & 2 & 4 \\ 1 & 0 & 3 \\ 2 & 1 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1/2 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 2 & 2 & 4 \\ 0 & -1 & 1 \\ 0 & 0 & -3 \end{bmatrix}$$

```
import scipy.linalg # LU in scipy library
A = np.array([ [2,2,4], [1,0,3], [2,1,2] ])
_,L,U = scipy.linalg.lu(A)
# 각각 출력합니다
print('L: '), print(L)
print('U: '), print(U)
```

```
L:
[[1.  0.  0. ]
 [0.5 1.  0. ]
 [1.  1.  1. ]]

U:
[[ 2.  2.  4.]
 [ 0. -1.  1.]
 [ 0.  0. -3.]]
```

SECTION 09-3 LU 분해(2)

9.3.1 치환 행렬을 통한 행 교환

- 일부 행렬은 상삼각형태로 변환되지 않음
- 사다리꼴 형태 아닌 행렬의 두 번째 줄과 세 번째 줄을 바꾸어(행 교환) 치환 행렬을 통해 구현

$$\begin{bmatrix} 3 & 2 & 1 \\ 0 & 0 & 5 \\ 0 & 7 & 2 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} 3 & 2 & 1 \\ 0 & 0 & 5 \\ 0 & 7 & 2 \end{bmatrix} = \begin{bmatrix} 3 & 2 & 1 \\ 0 & 7 & 2 \\ 0 & 0 & 5 \end{bmatrix}$$

- 치환 행렬은 **P**로 표기
  - 치환 행렬은 직교 행렬이므로 $\mathbf{P}^{-1} = \mathbf{P}^T$
  - 임의의 두 열 사이의 내적은 0이고 열 자체의 내적은 1입니다, 즉, $\mathbf{P}^T\mathbf{P} = \mathbf{I}$
    - □ 위에서 작성한 공식은 LU 분해를 수학적으로 설명한 것
    - □ SciPy는 실제로 **A** = **PLU**를 반환하며, 이를 $\mathbf{P}^T\mathbf{A} = \mathbf{LU}$ 로 쓸 수 있음

실습 문제

- 임의의 4x4 행렬 A를 생성해 LU분해를 증명하세요.
  - $P^T A = LU$ 임을 증명하세요.
  - $A = PLU$ 임을 증명하세요.
    - SciPy는 **A** = **PLU**를 반환하며, 이를 **P**^{T}**A** = **LU** 로 쓸 수 있음
  - P가 직교행렬인지를 증명하세요.
    - 직교행렬 증명
      - ☐ 행렬의 모든 열이 서로 직교
      - ☐ 각 열의 노름은 정확히 1
      - ☐ 직교 행렬의 역행렬은 그 행렬의 전치