인공지능수학

(선형대수학)

개 요

벡터, 행렬		
특징	**연산**	**특수형태**
노름, 거리, 차원, 내적, 행렬식	벡터합, 스칼라곱, 행렬합, 행렬곱	영행렬, 음행렬, 단위행렬, 역행렬, 대각행렬, 전치행렬 등

선형문제

- 선형결합
- 기본행변환
- 가우스-조르단 소거법

알고리즘

- LU분해
- QR분해
- 최소제곱법
- 그람-슈미트 방법

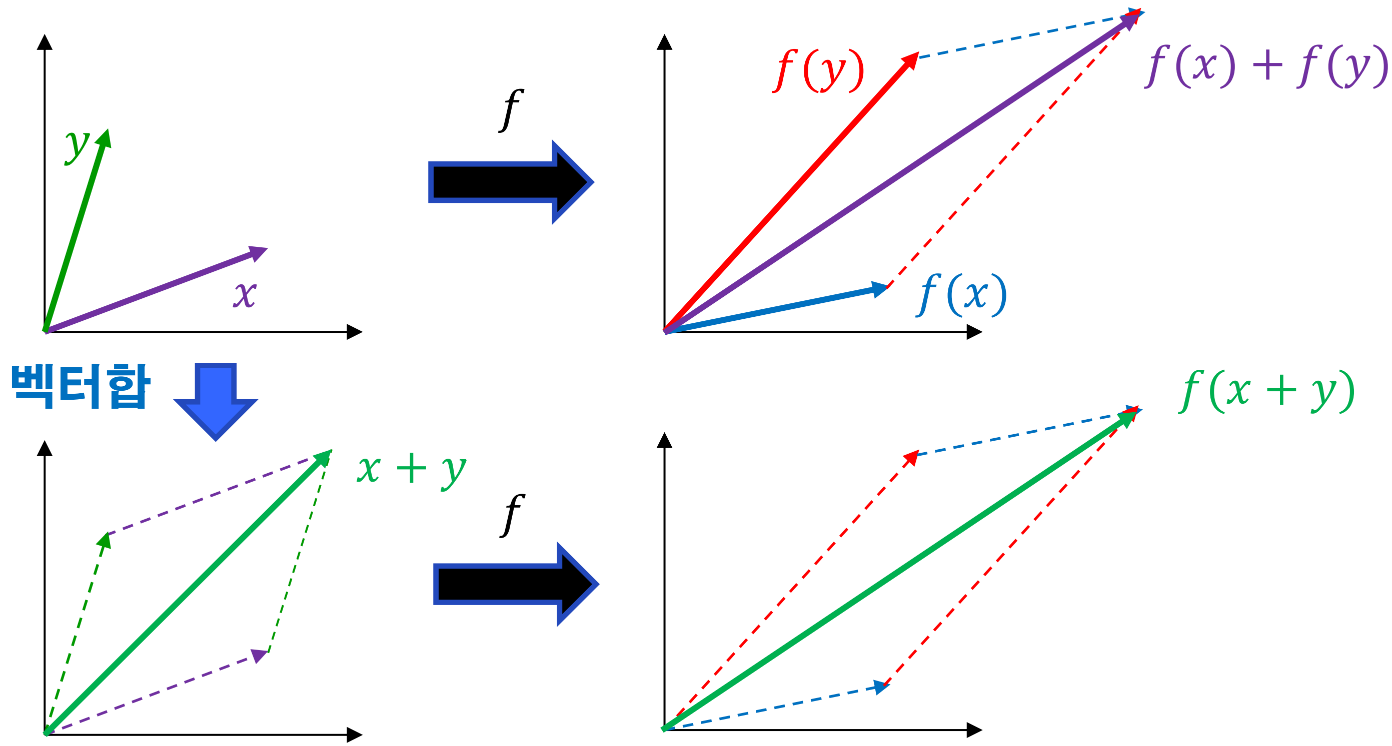
행렬의 특성값

- 고윳값&고유벡터
- 특잇값
  - 특잇값분해 (SVD)

선형성

■ "선형" 이란?

$$\boxed{\alpha \boldsymbol{f}(\boldsymbol{x}) + \beta \boldsymbol{f}(\boldsymbol{y}) = \boldsymbol{f}(\alpha \boldsymbol{x} + \beta \boldsymbol{y})}$$



선형다항식

■ "선형다항식" 이란?

$$\boxed{\boldsymbol{Ax = b}}$$

선형방정식 $3x = 2$

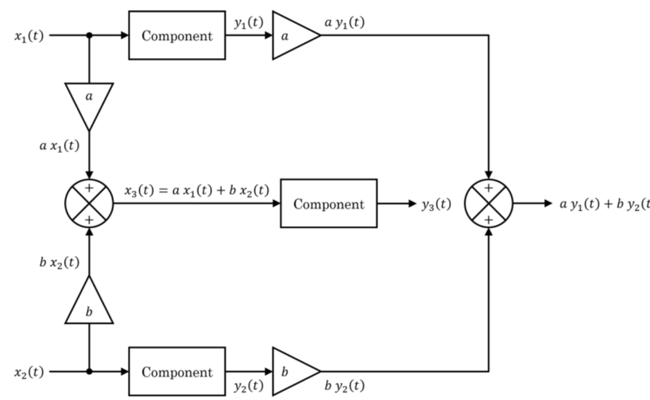
선형연립방정식 $\begin{cases} 2x + 3y = 4 \\ 3x + 6y = 9 \end{cases}$

행렬방정식 $\begin{pmatrix} 1 & 1 \\ 2 & 3 \end{pmatrix} \mathrm{x} = \begin{pmatrix} 4 \\ 9 \end{pmatrix}$

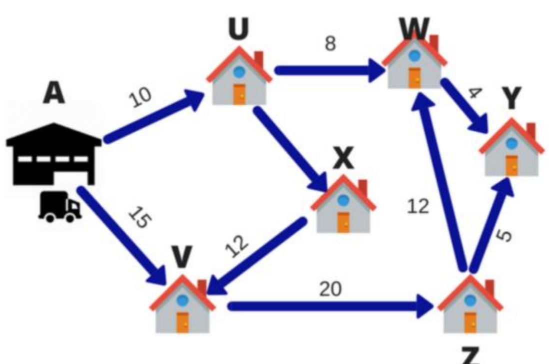
선형대수가 응용되는 분야

	Benefit Contribution per Unit of Each Activity				
Benefit	1	2	3	4	Minimum Acceptable Level
P	2	−1	4	3	80
Q	1	4	−1	2	60
R	3	5	4	−1	110
Unit cost	$400	$600	$500	$300	

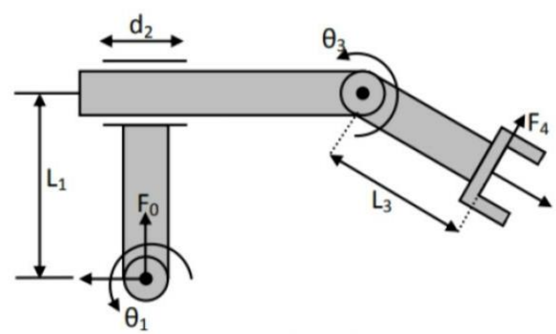
경제경영 : 선형계획법



선형회로 위 전력계산



경로탐색

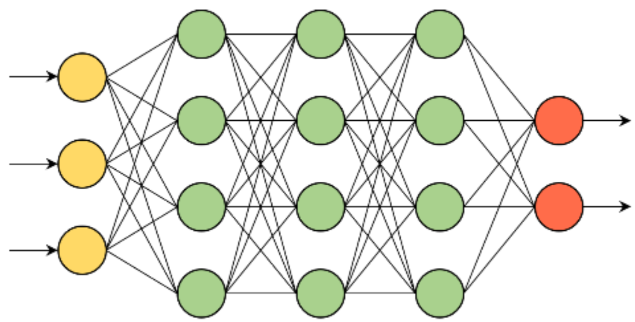


로봇공학 등...

선형대수가 응용되는 분야

딥러닝에서의 선형다항식

- 딥러닝이란 선형다항식을 이용하여 만들어지는 AI 구조이다.



$$\begin{pmatrix} w_{11}^{(4)} & w_{12}^{(4)} & w_{13}^{(4)} & w_{14}^{(4)} \\ w_{21}^{(4)} & w_{22}^{(4)} & w_{23}^{(4)} & w_{24}^{(4)} \end{pmatrix} \left(\cdots f \left(\begin{pmatrix} w_{11}^{(1)} & w_{12}^{(1)} & w_{13}^{(1)} \\ w_{21}^{(1)} & w_{22}^{(1)} & w_{23}^{(1)} \\ w_{31}^{(1)} & w_{32}^{(1)} & w_{33}^{(1)} \\ w_{41}^{(1)} & w_{42}^{(1)} & w_{43}^{(1)} \end{pmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \right) \right)$$

선형결합

Remind : 그래프로 표현하는 행렬연산

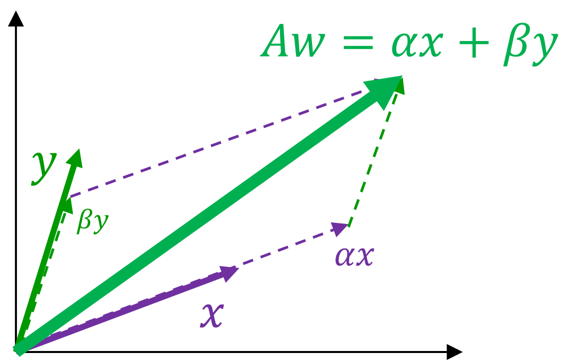
예시)

$$A = \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \end{bmatrix} = [x \quad y]$$

$$w = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$$

$$B = \begin{bmatrix} \alpha_1 & \alpha_2 \\ \beta_1 & \beta_2 \end{bmatrix} = [u \quad v]$$

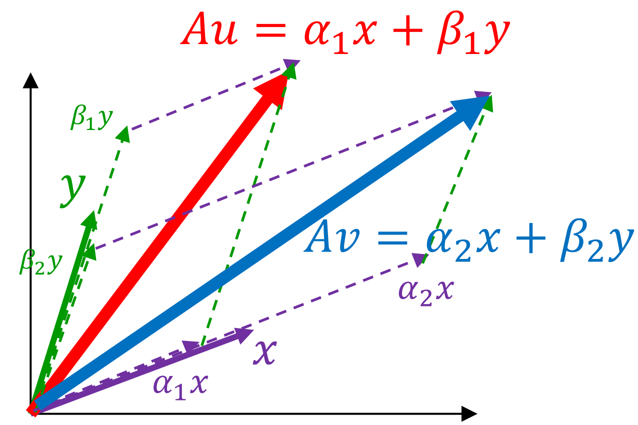
행렬-벡터곱



*** It makes a new vector, using column vectors of matrix**

행렬-행렬곱

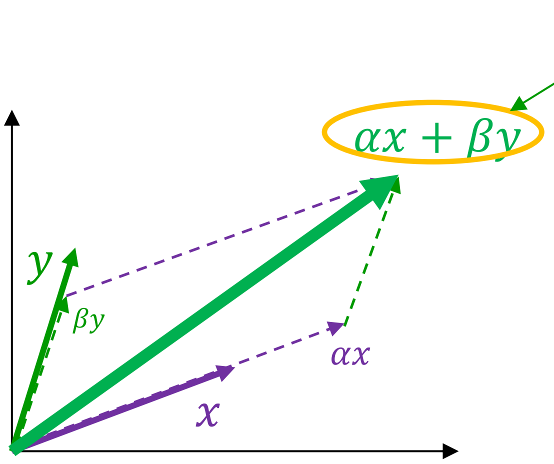
$$AB = [Au \quad Av]$$



*** It's changed column vectors of the matrix.**

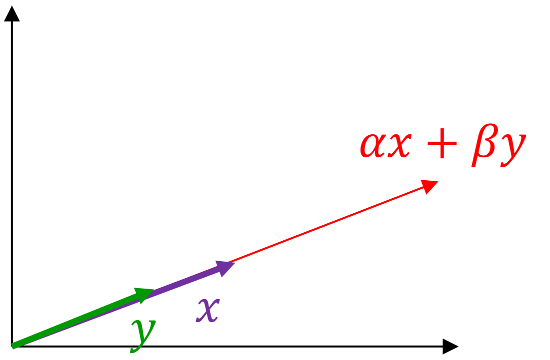
선형결합(linear combination)

■ 선형결합(일차결합);벡터들의 스칼라배의 합인 벡터



Aw = 0 라고 할 때
α 와 β는 무엇인가?

정답 : $\alpha = 0,\ \beta = 0$
(값이 유일하게 존재함)

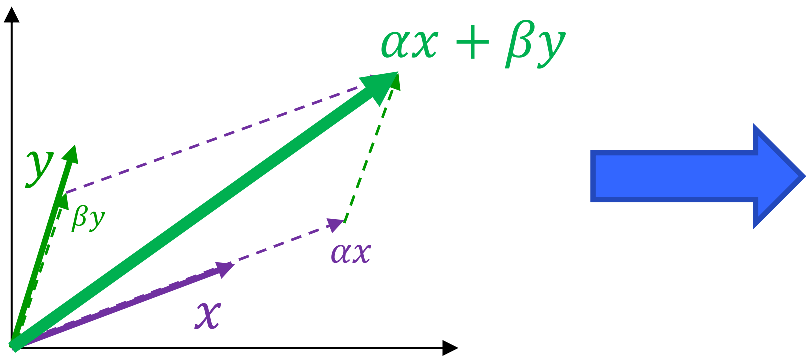


Aw = 0 라고 할 때
α 와 β는 무엇인가?

정답 : $\alpha = n,\ \beta = -n$
(값이 무한히 많음)

선형결합

■ 일차독립(linearly independent)



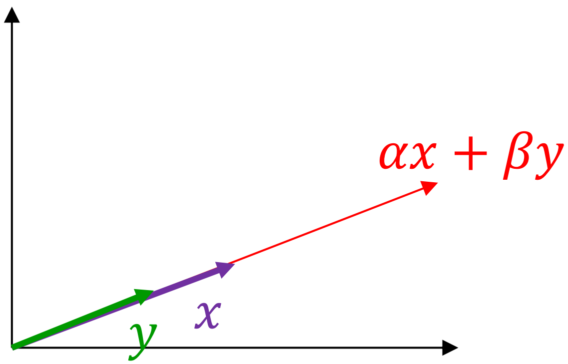
Aw = 0 라고 할 때
α 와 β는 무엇인가?

정답 : $\alpha = 0,\ \beta = 0$
(값이 유일하게 존재함)

값이 유일한 경우, $x \,\&\, y$를 일차독립이라 한다.

선형결합

■ 일차종속(linearly dependent)



Aw = 0 라고 할 때
α 와 β는 무엇인가?

정답 : $\alpha = n,\ \beta = -n$
(값이 무한히 많음)

값이 무한히 많은 경우, x & y를 일차종속이라 한다.

선형결합

■ 일차종속의 예시

예시)

$$x = \begin{bmatrix} 1 \\ 2 \end{bmatrix},\ y = \begin{bmatrix} 2 \\ 4 \end{bmatrix}$$

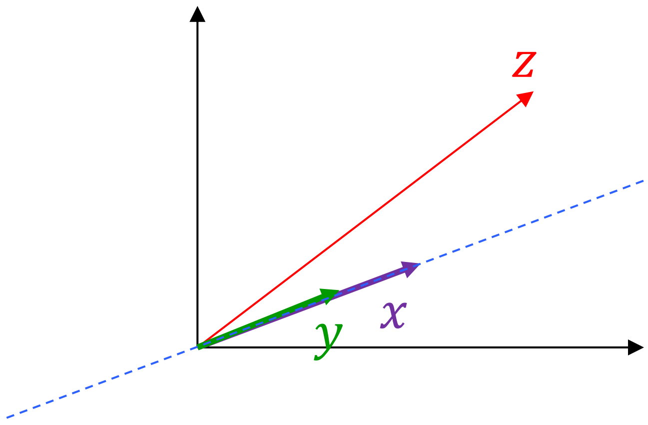
$$-2x + y = -2\begin{bmatrix} 1 \\ 2 \end{bmatrix} + \begin{bmatrix} 2 \\ 4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

위의 경우, x & y는 **일차종속**이다.

선형결합과 생성(span)

■ 일차종속과 생성

이 경우, x & y 의 선형결합으로
z를 만들 수 있는가?

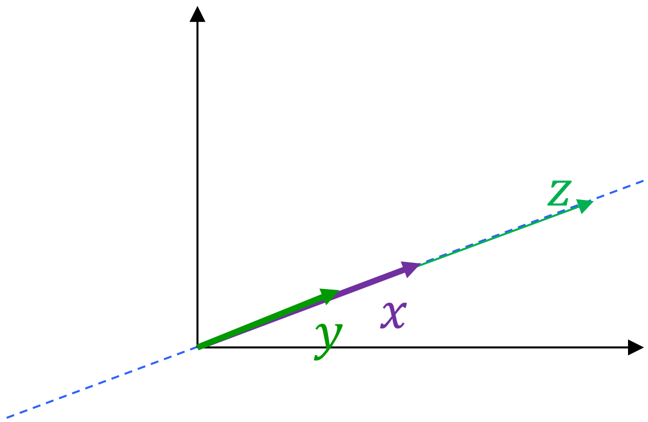


NO!!

선형결합과 생성(span)

■ 일차종속과 생성

이 경우, x & y 의 선형결합으로
z를 만들 수 있는가?



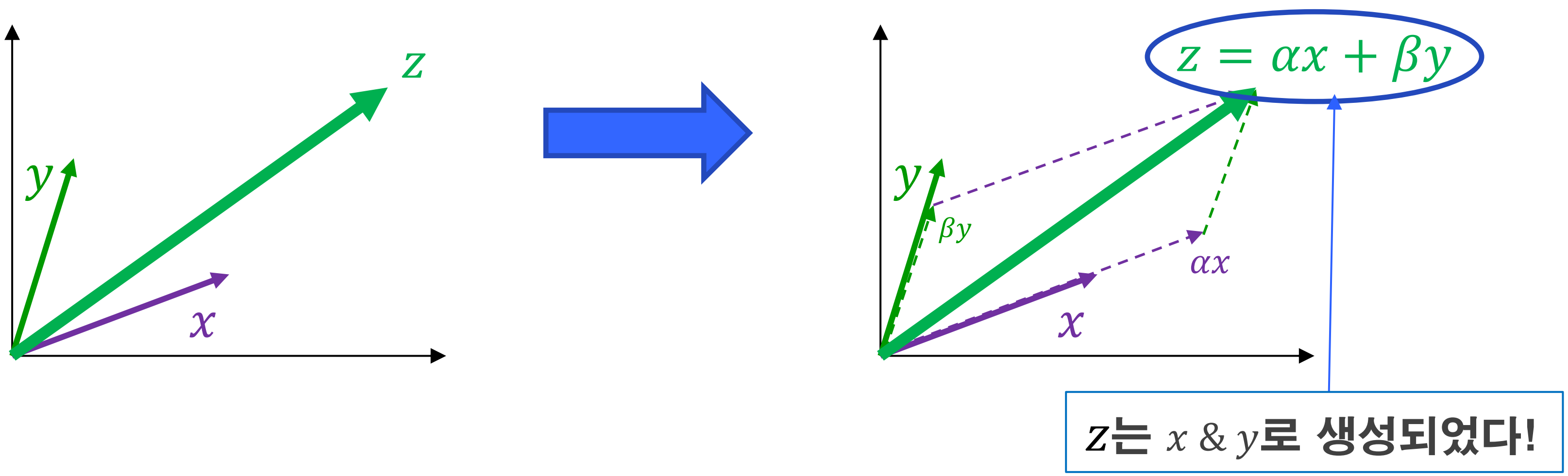
Yes!!

무한히 많은 z를 만들 수 있다!

선형결합과 생성(span)

■ 일차독립과 생성

이 경우, x & y 의 선형결합으로
z를 만들 수 있는가?

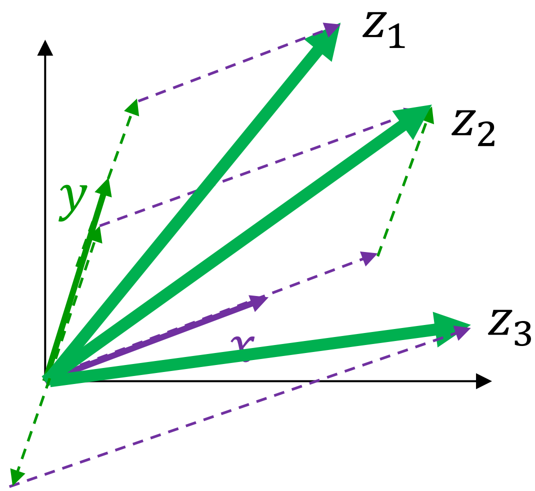


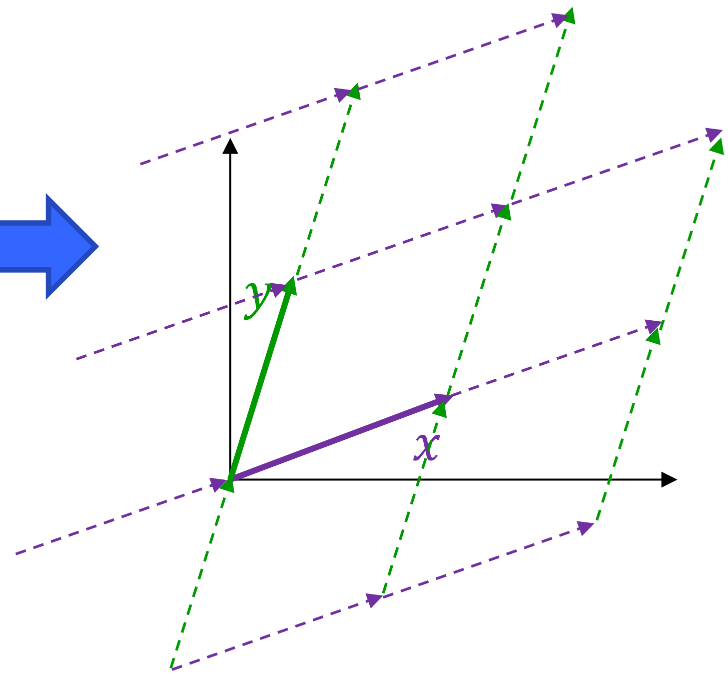
선형결합과 생성(span)

■ 일차독립과 생성

일차독립인 x & y를 이용하면 xy평면 위의 모든 벡터를 표현할 수 있다!

x-y 평면 = span(x, y)





선형결합과 생성(span)

■ 일차독립과 생성

예시)

$$x = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$$

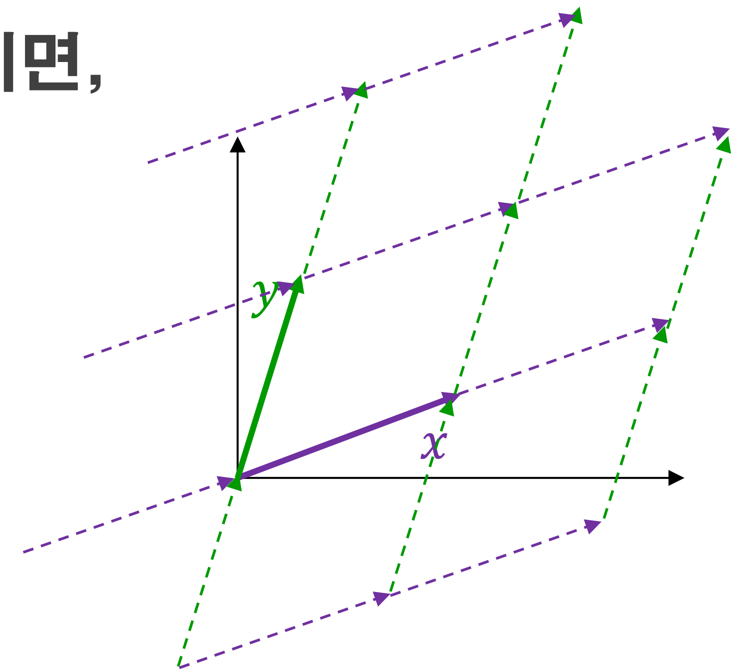
$$y = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$$

$S = \{x, y\}$**일 때,**

$$Span(S) = \{\alpha x + \beta y \mid x, y \in S\}$$

x, y **가 일차독립이면,**

$Span(S) = R^2$



선형결합과 생성(span)

■ 생성, 기저, 차원

예시)

$$x = \begin{bmatrix} 1 \\ 1 \\ 3 \end{bmatrix}$$

$$y = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

$$z = \begin{bmatrix} 2 \\ 3 \\ 8 \end{bmatrix}$$

$S = \{x, y, z\}$**이면** $Span(S) = R^3$

왜냐하면 x, y, z **는 선형독립이다.**

이 때, S**를** R^3**의 기저(basis)라 한다.**

기저 : 공간을 생성하기 위해 필요한 일차독립인 벡터로 구성된 집합

$|S| = 3$**은** R^3**의 차원(dimension)이라 한다**

차원 : 기저의 크기

선형방정식의 풀이

행렬-벡터곱의 응용

선형연립방정식

$$\begin{cases} 2x + 3y = 4 \\ 4x + 7y = 9 \end{cases}$$

행렬방정식

$$\underbrace{\begin{pmatrix} 2 & 3 \\ 4 & 7 \end{pmatrix}}_{\text{계수행렬}} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 4 \\ 9 \end{pmatrix}$$

선형방정식의 풀이

방정식을 푸는 방법

연립방정식

$$\begin{cases} 2x + 3y = 4 \cdots\cdots (1) \\ 4x + 7y = 9 \cdots\cdots (2) \end{cases}$$

(식2) – 2×(식1)

$$\begin{cases} 2x + 3y = 4 \\ \qquad\quad y = 1 \end{cases}$$

(식1) – 3×(식2)

$$\begin{cases} 2x \qquad = 1 \\ \qquad\quad y = 1 \end{cases}$$

정답 : $x = \frac{1}{2},\ y = 1$

행렬

계수행렬

$$\left(\begin{array}{cc:c} 2 & 3 & 4 \\ 4 & 7 & 9 \end{array}\right)$$

(행 2) – 2×(행 1)

$$\left(\begin{array}{cc:c} 2 & 3 & 4 \\ 0 & 1 & 1 \end{array}\right)$$

(행 1) – 3×(행 2)

$$\left(\begin{array}{cc:c} 2 & 0 & 1 \\ 0 & 1 & 1 \end{array}\right)$$

(행 1) ÷ 2

$$\left(\begin{array}{cc:c} 1 & 0 & \frac{1}{2} \\ 0 & 1 & 1 \end{array}\right)$$

선형방정식의 풀이

■ 기초행변환

Case 1.

(행 n)× α

$$\left(\begin{array}{cc:c} 1 & 1 & 1 \\ 2 & 1 & 3 \end{array}\right)$$

(행 1)× 3

$$\left(\begin{array}{cc:c} 3 & 3 & 3 \\ 2 & 1 & 3 \end{array}\right)$$

Case 2.

(행 n) ↔ (행 m)

$$\left(\begin{array}{cc:c} 2 & 3 & 4 \\ 4 & 7 & 9 \end{array}\right)$$

(행 1) ↔ (행 2)

$$\left(\begin{array}{cc:c} 4 & 7 & 9 \\ 2 & 3 & 4 \end{array}\right)$$

Case 3.

(행 n) – α ×(행 m)

$$\left(\begin{array}{cc:c} 2 & 3 & 4 \\ 4 & 7 & 9 \end{array}\right)$$

(행 2) – 2×(행 1)

$$\left(\begin{array}{cc:c} 2 & 3 & 4 \\ 0 & 1 & 1 \end{array}\right)$$

선형방정식의 풀이

■ 행사다리꼴 행렬(Row Echelon Form matrix)

대각행렬

$$X = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

행사다리꼴 행렬(REF)

선행요소

$$X = \begin{bmatrix} 1 & 3 & 1 & 5 \\ 0 & 2 & 1 & 0 \\ 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 4 \end{bmatrix} \quad X = \begin{bmatrix} 2 & 2 & 0 & 3 & 1 \\ 0 & 1 & 1 & 2 & 2 \\ 0 & 0 & 5 & 1 & 3 \\ 0 & 0 & 0 & 0 & 4 \end{bmatrix}$$

기약행사다리꼴 행렬(Reduced REF;RREF)

$$X = \begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 3 \end{bmatrix} \quad X = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 2 \\ 0 & 0 & 1 & 0 & 3 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

*선행요소가 전부 "1"

선형방정식의 풀이

■ 가우스-조르단 소거법

- 가우스-조르단 소거법은 **선형연립방정식을 풀기위해 사용**되는 대표적인 알고리즘이다.

- 알고리즘에서는 **오로지 기본행변환만을 이용**하며, 다음과 같이 두 단계의 과정으로 나뉜다.

1) 첫 번째 : forward elimination
   - 주어진 행렬을 행사다리꼴 행렬로 변환한다.

2) 두 번째 : back substitution
   - 행사다리꼴 행렬을 기약행사다리꼴 행렬로 변환한다.

계수행렬 → 가우스-조르단 소거법 → RREF

$$\left(\begin{array}{ccc:c} 2 & 1 & -1 & 8 \\ -3 & -1 & 2 & -11 \\ -2 & 1 & 2 & -3 \end{array}\right) \xrightarrow{\text{가우스-조르단 소거법}} \left(\begin{array}{ccc:c} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & -1 \end{array}\right)$$

선형방정식의 풀이

가우스-조르단 소거법

예시)

$$\left(\begin{array}{ccc:c} 2 & 1 & -1 & 8 \\ -3 & -1 & 2 & -11 \\ -2 & 1 & 2 & -3 \end{array}\right)$$

(행 2) + $\frac{3}{2}$×(행 1)
(행 3) + (행 1)

$$\left(\begin{array}{ccc:c} 2 & 1 & -1 & 8 \\ 0 & \frac{1}{2} & \frac{1}{2} & 1 \\ 0 & 2 & 1 & 5 \end{array}\right)$$

(행 3) − 4×(행 2)

$$\left(\begin{array}{ccc:c} 2 & 1 & -1 & 8 \\ 0 & \frac{1}{2} & \frac{1}{2} & 1 \\ 0 & 0 & -1 & 1 \end{array}\right)$$

(행 2) + $\frac{1}{2}$×(행 3)
(행 1) − (행 3)

$$\left(\begin{array}{ccc:c} 2 & 1 & 0 & 7 \\ 0 & \frac{1}{2} & 0 & \frac{3}{2} \\ 0 & 0 & -1 & 1 \end{array}\right)$$

(행 2) × 2
(행 3) × (−1)

$$\left(\begin{array}{ccc:c} 2 & 1 & 0 & 7 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & -1 \end{array}\right)$$

(행 1) − (행 2)
(행 1) × $\frac{1}{2}$

$$\left(\begin{array}{ccc:c} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & -1 \end{array}\right)$$

개 요

벡터, 행렬

특징	연산	특수형태
노름, 거리, 차원, 내적, 행렬식	벡터합, 스칼라곱, 행렬합, 행렬곱	영행렬, 음행렬, 단위행렬, 역행렬, 대각행렬, 전치행렬 등

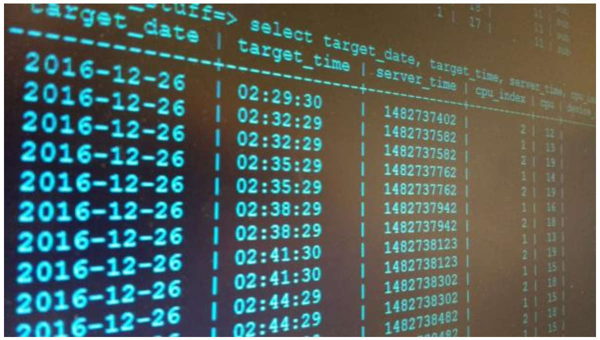
선형문제
- 선형결합
- 기본행변환
- 가우스-조르단 소거법

알고리즘
- LU분해
- QR분해
- 최소제곱법
- 그람-슈미트 방법

행렬의 특성값
- 고윳값&고유벡터
- 특잇값
  - 특잇값분해 (SVD)

Review : IT기술과 선형대수

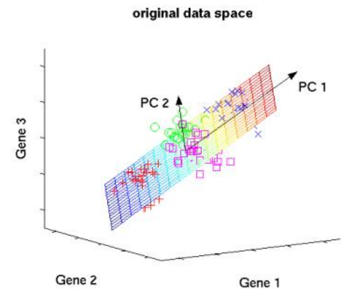
선형 데이터



행렬

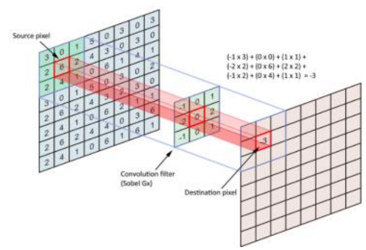
$$
\begin{pmatrix}
A_{11} & A_{12} & A_{13} & \cdots & A_{1n} \\
A_{21} & A_{22} & A_{23} & \cdots & A_{2n} \\
A_{31} & A_{32} & A_{33} & \cdots & A_{3n} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
A_{m1} & A_{m2} & A_{m3} & \cdots & A_{mn}
\end{pmatrix}
$$

데이터 분석



데이터 분석을 위해서
데이터 처리기술이 필요!

데이터 처리



데이터 처리를 위해서
고속연산 기술이 필요!

고속연산

고속연산?

카를 프리드리히 가우스
(1777~1855)

초등학생이었던 가우스는 1부터 100까지의 합을 구하는 문제를 $\frac{100(100+1)}{2}$ 라는 식으로 빠르게 계산하였다!

$$\begin{array}{r} 1 + 2 + \cdots + 99 + 100 \\ 100 + 99 + \cdots + 2 + 1 \\ \hline 101 + 101 + \cdots + 101 + 101 \end{array}$$

실제로 1부터 n까지의 합을 구하려면 몇 번 계산해야 할까?

1) $1 + 2 + \cdots + (n-1) + n$: 덧셈 $n-1$ 번

2) $\frac{n(n+1)}{2}$: 덧셈 1번, 곱셈 1번, 나눗셈 1번 → 1보다 빠른연산

계산량

■ 계산량이란?

- 실제로 값을 구할때까지 필요한 계산횟수 또는 계산 소요시간

예시)

$$x = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}, y = \begin{bmatrix} 2 \\ 4 \\ 1 \end{bmatrix}$$

Q1) $x \cdot y$을 구하시오.

$$x \cdot y = 1 \times 2 + 2 \times 4 + 3 \times 1 = 13$$

Q2) $x \cdot y$의 계산량은 얼마인가?

곱셈 : 3 번
덧셈 : 2 번
총 소요시간 : $3t_1+2t_2$

t_1 : 곱셈 1회에 걸리는 시간
t_2 : 덧셈 1회에 걸리는 시간

계산량

예시)

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

Q1) $x \cdot y$**을 구하시오.**

$$x \cdot y = x_1 \times y_1 + x_2 \times y_2 + \cdots + x_n \times y_n$$

Q2) $x \cdot y$**의 계산량은 얼마인가?**

곱셈 : $\boldsymbol{n}$ **번**
덧셈 : $\boldsymbol{n-1}$ **번**
총 소요시간 : $n\mathrm{t}_1 + (\boldsymbol{n}-\mathbf{1})\mathrm{t}_2$

t_1 **: 곱셈 1회에 걸리는 시간**
t_2 **: 덧셈 1회에 걸리는 시간**

계산복잡도

■ 계산복잡도 : 빅오(Big-O) 표기법

- 소요시간에 미치는 영향력이 가장 큰 변수(주로 최대차수인 변수)를 이용하여 복잡도를 나타내는 방법.

예시)

연산 소요시간	계산복잡도
$nt_1 + (n-1)t_2$	$O(n)$
$(n+1)t_1 + 2n^2t_2 + t_3$	$O(n^2)$
$(2^n + n + 1)t_1$	$O(2^n)$
$(n^5 + n^2 + 1)t_1 + n!\,t_2$	$O(n!)$

다항식 시간(P)

비다항식 시간(NP)

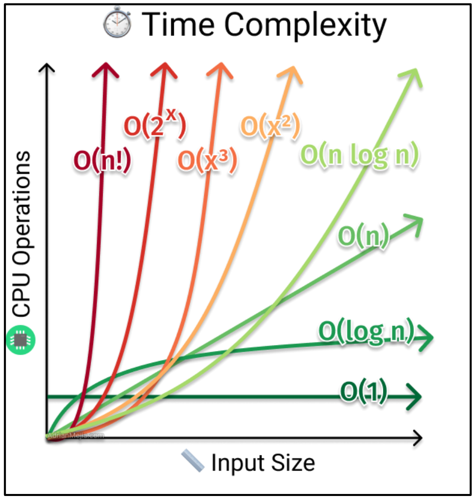
계산복잡도

■ 효율적인 계산이란?

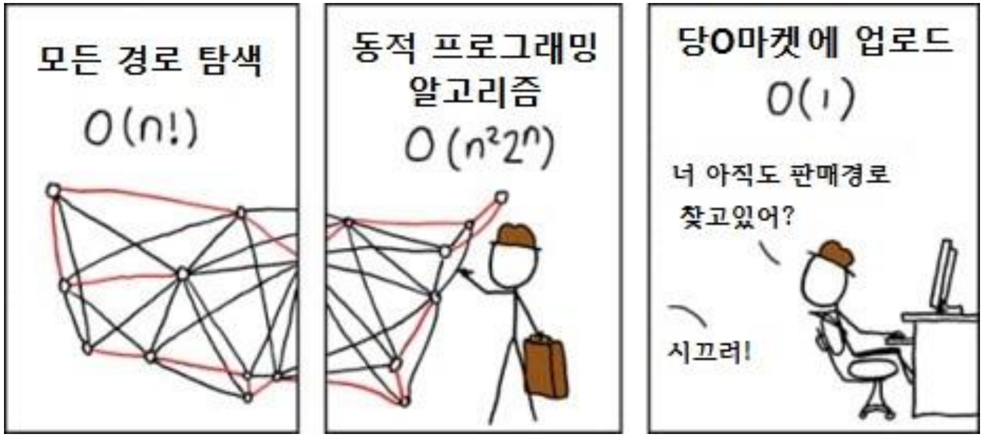
- 낮은 차수의 다항식 시간을 가진 계산방법

■ 고속연산이란?

- 수학적 / 공학적 기법을 전부 동원하여 구현한 효율적인 연산방법



계산복잡도 그래프



효율적인 계산? (외판원 문제)

수학이 많은 문제를 해결해주지만,
가끔 수학이 아닌 방법으로도 문제를 해결할 수 있다!

역행렬

■ 역행렬과 행렬방정식의 해

Q) A는 행렬, b는 벡터일 때,
다음 행렬방정식의 해는 어떻게 구할 수 있는가?

$$A\mathrm{x} = b$$

A) A^{-1}를 찾은 뒤 $x = A^{-1}b$를 계산한다.

하지만 어떻게 A^{-1}를 찾을 수 있을까?

■ 행렬식과 역행렬

- 행렬식이란 주어진 정방행렬로 인한 변화량을 나타내는 스칼라값.
- 행렬식이 0이 아니라면 주어진 정방행렬은 역행렬이 존재한다.

2×2행렬의 경우

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

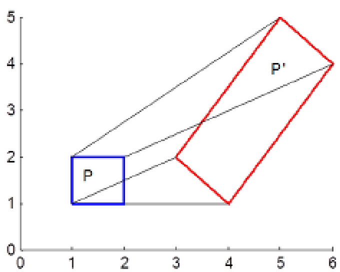
$$\det(A) = \begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$$

$$A^{-1} = \frac{1}{\det(A)} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

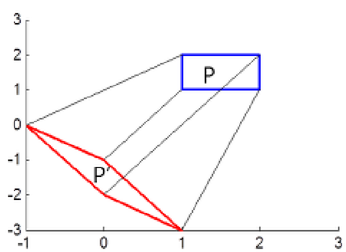
행렬식(Determinant)

■ 행렬식의 특징

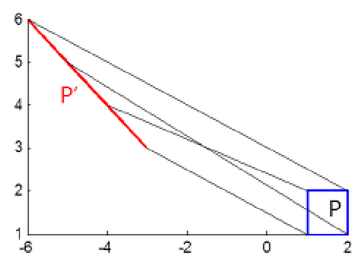
- 행렬방정식이 주어졌을 때 정방행렬의 행렬식이 0 이 아니라면, 주어진 행렬방정식은 유일한 해를 갖는다.
- 행렬식은 선형변환 시 단위 면적의 변화량(선형변환의 scale 성분)을 나타낸다.



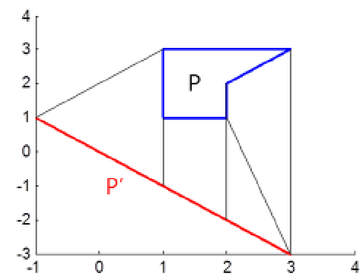
$A = \begin{pmatrix} 1 & 2 \\ -1 & 3 \end{pmatrix}$, det($A$)=5



$A = \begin{pmatrix} 1 & -1 \\ -2 & 1 \end{pmatrix}$, det($A$)=-1



$A = \begin{pmatrix} -2 & -1 \\ 2 & 1 \end{pmatrix}$, det($A$)=0



$A = \begin{pmatrix} 2 & -1 \\ -2 & 1 \end{pmatrix}$, det($A$)=0

행렬식과 역행렬

3×3 행렬의 경우

$$A = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

**일반적으로, a_{ij}의 여인자

$$C_{ij} = (-1)^{i+j} det\, A_{ij}$$

a_{ij} 의 여인자 (Cofactor of a) $= C_{11} = \begin{vmatrix} e & f \\ h & i \end{vmatrix}$

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

a_{ij} 의 여인자 (Cofactor of i) $= C_{33} = \begin{vmatrix} a & b \\ d & e \end{vmatrix}$

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

a_{ij} 의 여인자 (Cofactor of b) $= C_{12} = \begin{vmatrix} d & f \\ g & i \end{vmatrix}$

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

행렬식과 역행렬

3×3 **행렬의 경우** $A = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$

라플라스 전개
(여인자 전개)

$$\det(A) = a\begin{vmatrix} e & f \\ h & i \end{vmatrix} + b\begin{vmatrix} d & f \\ g & i \end{vmatrix} + c\begin{vmatrix} d & e \\ g & h \end{vmatrix} = aC_{11} + bC_{12} + cC_{13}$$

$$A^{-1} = \frac{1}{\det(A)}\begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix}^T$$

행렬식과 역행렬

$n \times n$ **행렬 :** $A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix}$

$$\det(A) = a_{11}C_{11} + \cdots + a_{1n}C_{1n} = \sum_{j=1}^{n} a_{1j}C_{1j}$$

임의의 i행, j열에 대한 라플라스 전개,

$\det(A) = a_{i1}C_{i1} + \cdots + a_{in}C_{in} = \sum_{j=1}^{n} a_{ij}C_{ij}$ **(i 행 확장)**

$\det(A) = a_{1j}C_{1j} + \cdots + a_{nj}C_{nj} = \sum_{i=1}^{n} a_{ij}C_{ij}$ **(j 열 확장)**

$$A^{-1} = \frac{1}{\det(A)} \begin{bmatrix} C_{11} & \cdots & C_{1n} \\ \vdots & \ddots & \vdots \\ C_{n1} & \cdots & C_{nn} \end{bmatrix}^{T}$$

역행렬을 풀기 위한 계산량

$n \times n$ 행렬의 경우 $A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix}$

- $\det(A)$를 계산하자. A 가 4×4 행렬이라면

$$C_{11} = \begin{vmatrix} a_{22} & a_{23} & a_{24} \\ a_{32} & a_{33} & a_{34} \\ a_{42} & a_{43} & a_{44} \end{vmatrix} = a_{22} \begin{vmatrix} a_{33} & a_{34} \\ a_{43} & a_{44} \end{vmatrix} + a_{23} \begin{vmatrix} a_{32} & a_{34} \\ a_{42} & a_{44} \end{vmatrix} + a_{24} \begin{vmatrix} a_{32} & a_{33} \\ a_{42} & a_{43} \end{vmatrix}$$

하나의 C_{ij} 값을 계산하려면 2×2 행렬을 3번 계산하는 것이 필요하다.

따라서, $\det(A)$ 값을 구하려면 C_{ij} 를 4 번 계산하는 것이 필요함!

- $n \times n$의 경우, **행렬식 $\det(A)$ 은 $O(n!)$의 시간 복잡도를 갖는다.**

가우스-조단 소거법의 응용

확장행렬 → 가우스-조단 소거법 → RREF

$$\left(\begin{array}{ccc:c} 2 & 1 & -1 & 8 \\ -3 & -1 & 2 & -11 \\ -2 & 1 & 2 & -3 \end{array}\right) \xrightarrow{\text{가우스-조단 소거법}} \left(\begin{array}{ccc:c} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & -1 \end{array}\right)$$

A^{-1} ***가 존재하는 경우에***

확장행렬 → 가우스-조단 소거법 → RREF

$$\left(\begin{array}{c:c} A & b \end{array}\right) \xrightarrow{\text{가우스-조단 소거법}} \left(\begin{array}{c:c} I & A^{-1}b \end{array}\right)$$

가우스-조던 소거법의 응용

확장행렬 → 가우스-조단 소거법 → RREF

$$\left(\begin{array}{c|c} A & b \end{array}\right) \xrightarrow{\text{가우스-조단 소거법}} \left(\begin{array}{c|c} I & A^{-1}b \end{array}\right)$$

$$\Downarrow$$

확장행렬 → 가우스-조단 소거법 → RREF

$$\left(\begin{array}{c|c} A & I \end{array}\right) \xrightarrow{\text{가우스-조단 소거법}} \left(\begin{array}{c|c} I & A^{-1} \end{array}\right)$$

가우스-조단 소거법의 응용

예제.

$$\left(\begin{array}{ccc|ccc} 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 2 & 2 & 0 & 1 & 0 \\ 1 & 2 & 3 & 0 & 0 & 1 \end{array}\right)$$

$$\left(\begin{array}{ccc|ccc} 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 2 & 2 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & -1 & 1 \end{array}\right)$$

$$\left(\begin{array}{ccc|ccc} 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & -1 & 1 & 0 \\ 0 & 0 & 1 & 0 & -1 & 1 \end{array}\right)$$

$$\left(\begin{array}{ccc|ccc} 1 & 0 & 0 & 2 & -1 & 0 \\ 0 & 1 & 0 & -1 & 2 & -1 \\ 0 & 0 & 1 & 0 & -1 & 1 \end{array}\right)$$

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 2 \\ 1 & 2 & 3 \end{pmatrix}\begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 2 \\ 1 & 2 & 3 \end{pmatrix}^{-1} = \begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix}$$

인공지능과 선형대수

❖ 역행렬을 구하기 위한 계산량

$n \times n$ 행렬의 경우 $A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix}$

- **가우스-조단 소거법을 사용한다.**

  -각 행에 대해서 n 번의 실행시간이 필요하다.

  - 하삼각부분을 삭제하기 위해서는 (n-1)+(n-2)+$\cdots + 1 = \frac{n(n-1)}{2}$ 번의 계산이 필요함.

  - 우삼각 부분을 삭제하는 데도 (n-1)+(n-2)+$\cdots + 1 = \frac{n(n-1)}{2}$ 번의 계산 필요함.

❖ $n \times n$ 행렬을 가우스-조단 소거법으로 역행렬을 구할 경우,
A^{-1} 를 구하는데 필요한 시간은 $O(n^3)$의 시간 복잡도를 갖는다

✓ 가우스-조단 소거법을 이용하여 선형 방정식을 계산하기 위한 시간 복잡도

■ 다음과 같이 n개의 선형방정식이 주어졌다고 가정하자

$$
\begin{aligned}
&(1)\ a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\
&(2)\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\
&\qquad\qquad\qquad\vdots \\
&(n)\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n
\end{aligned}
$$

인공지능과 선형대수

✓ 가우스-조단 소거법을 이용하여 선형 방정식을 계산하기 위한 시간 복잡도

■ 모든 행에서 첫 번째 변수 x_1 를 소거해 보자

$$-\frac{a_{21}}{a_{11}}\cdot(1)+(2), -\frac{a_{31}}{a_{11}}\cdot(1)+(3), \cdots, -\frac{a_{n1}}{a_{11}}\cdot(1)+(n)$$

$$\begin{array}{ll}
(1) & a_{11}^{(1)}x_1 + a_{12}^{(1)}x_2 + \cdots + a_{1n}^{(1)}x_n = b_1^{(1)} \\
(2) & \qquad\qquad a_{22}^{(1)}x_2 + \cdots + a_{2n}^{(1)}x_n = b_2^{(1)} \\
 & \qquad\qquad\qquad \vdots \\
(n) & \qquad\qquad a_{n2}^{(1)}x_2 + \cdots + a_{nn}^{(1)}x_n = b_n^{(1)}
\end{array}$$

$$a_{ij}^{(1)} = a_{ij}^{(0)},\ b_i^{(1)} = b_i^{(0)}\ (i=1)$$

$$a_{ij}^{(1)} = 0\ \ (i \neq 1, j = 1)$$

$$a_{ij}^{(1)} = a_{ij}^{(0)} - \frac{a_{i1}^{(0)}}{a_{11}^{(0)}} a_{1j}^{(0)}\ , b_i^{(1)} = b_i^{(0)} - \frac{a_{i1}^{(0)}}{a_{11}^{(0)}} b_1^{(0)}\ \ (i \neq 1, j \geq 2)$$

인공지능과 선형대수

✓ 가우스-조단 소거법을 이용하여 선형 방정식을 계산하기 위한 시간 복잡도

■ 모든 행에서 두번째 변수 x_2 를 소거해 보자

$$-\frac{a_{12}^{(1)}}{a_{22}^{(1)}}\cdot(2)+(1),-\frac{a_{32}^{(1)}}{a_{22}^{(1)}}\cdot(2)+(3),\cdots,-\frac{a_{n2}^{(1)}}{a_{22}^{(1)}}\cdot(2)+(n)$$

$$\begin{array}{llcl}
(1) & a_{11}^{(2)}x_1 & +\cdots+a_{1n}^{(2)}x_n & =b_1^{(2)} \\
(2) & \qquad a_{22}^{(2)}x_2 & +\cdots+a_{2n}^{(2)}x_n & =b_2^{(2)} \\
 & & \vdots & \\
(n) & & +\cdots+a_{nn}^{(2)}x_n & =b_n^{(2)}
\end{array}$$

$$a_{ij}^{(2)}=a_{ij}^{(1)},\ b_i^{(2)}=b_i^{(1)}\ \left(i=2, 2\le j\le n\right)$$

$$a_{ij}^{(2)}=0\ \left(i\ne 2, j=2\right)$$

$$a_{ij}^{(2)}=a_{ij}^{(1)}-\frac{a_{i2}^{(1)}}{a_{22}^{(1)}}a_{2j}^{(1)}\ ,b_i^{(2)}=b_i^{(1)}-\frac{a_{i2}^{(1)}}{a_{22}^{(1)}}b_2^{(1)}\ \left(i\ne 2, j\ge 3\right)$$

인공지능과 선형대수

✓ 가우스-조단 소거법을 이용하여 선형 방정식을 계산하기 위한 시간 복잡도

■ 앞서의 과정을 모든 변수 $x_3 \ldots, x_n$ 에 대해 반복해보면

$$
\begin{array}{lr}
(1)\ a_{11}^{(n-1)}x_1 & = b_1^{(n-1)} \\
(2)\ \qquad a_{22}^{(n-1)}x_2 & = b_2^{(n-1)} \\
\qquad\qquad \vdots & \\
(n)\ \qquad\qquad a_{nn}^{(n-1)}x_n & = b_n^{(n-1)}
\end{array}
$$

■ 위 방정식에서 다음과 같은 결과를 얻을 수 있다.

$$x_1 = \frac{b_1^{(n-1)}}{a_{11}^{(n-1)}}, x_2 = \frac{b_2^{(n-1)}}{a_{22}^{(n-1)}}, \ldots, x_n = \frac{b_n^{(n-1)}}{a_{nn}^{(n-1)}}$$

인공지능과 선형대수

❖ 역행렬을 위한 시간복잡도 비교

계산 방법	Big-O 표기법
라플라스 전개	$O(n!)$
가우스조단 소거법	$O(n^3)$
Strassen algorithm [1]	$O(n^{2.807})$
Coppersmith-Winograd [2]	$O(n^{2.376})$

참고 : (Wikipedia : Computational complexity of mathematical operations)
https://en.wikipedia.org/wiki/Computational_complexity_of_mathematical_operations

[1] Volker Strassen (Aug 1969). "Gaussian elimination is not optimal". Numerische Mathematik. 13 (4): 354–356.
[2] D. Coppersmith; S. Winograd (1981). "On the asymptotic complexity of matrix multiplication". Proc. 22nd Annual Symposium on Foundations of Computer Science (FOCS). pp. 82–90

인공지능과 선형대수

❖ LU 분해? 왜 분해를 하는가?

예제) 다항식 분해 (인수분해)

질문1) 다음 방정식을 풀어라 $x^5 - 15x^3 - 10x^2 + 24x$.

정답) 여러가지 이유로 매우 어려운 문제이다.

※ 위 방정식을 해결할 수있는 형식화된 식을 찾을 수 있는가?

인공지능과 선형대수

❖ 왜 분해하는가?

질문2) 다음 방정식을 풀어라.

$$x^5 - 15x^3 - 10x^2 + 24x.$$

위 방정식에 대해 우리는 다음과 같은 사실을 알고 있다:

$$x^5 - 15x^3 - 10x^2 + 24x = x(x-1)(x+2)(x+3)(x-4)$$

정답) $x = 0,\ 1, -2, -3,\ 4$

답을 구하는 과정은 매우 쉽다. 이미 방정식이 인수 분해가 되어 있기 때문이다!

때때로, 분해는 해를 찾는데 많은 도움을 준다.

행렬 분해도 또한 행렬 계산에 유용하게 활용된다.

인공지능과 선형대수

❖ LU 분해

하삼각행렬

$$L = \begin{bmatrix} \bullet & 0 & 0 \\ \bullet & \bullet & 0 \\ \bullet & \bullet & \bullet \end{bmatrix}$$

상삼각행렬

$$U = \begin{bmatrix} \bullet & \bullet & \bullet \\ 0 & \bullet & \bullet \\ 0 & 0 & \bullet \end{bmatrix}$$

$$A \xrightarrow{\text{LU decomposition}} LU$$

$$\begin{bmatrix} \star & \star & \star \\ \star & \star & \star \\ \star & \star & \star \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ \blacksquare & 1 & 0 \\ \blacksquare & \blacksquare & 1 \end{bmatrix} \begin{bmatrix} \blacklozenge & \blacklozenge & \blacklozenge \\ 0 & \blacklozenge & \blacklozenge \\ 0 & 0 & \blacklozenge \end{bmatrix}$$

대각 요소는 1이 되어야 한다.

인공지능과 선형대수

❖ LU 분해

문제

주어진 연립 방정식

$$\begin{cases} a_{11}x_1 + \cdots + a_{1n}x_n = b_1 \\ \vdots \\ a_{n1}x_1 + \cdots + a_{nn}x_n = b_n \end{cases}$$

주어진 조건

$a_{11}, \cdots, a_{nn}$ **은 결정됨.**

$b_1, \cdots, b_n$ **값은 가변적**

❖ 방정식 **Ax=b** 에 대해

주어진 **b=**$(b_1, \cdots, b_n)$ 를 갖는 해 **x** 를 쉽고 효율적으로 구할 수 있다.

❖ LU 분해

Pivot

예시) A

$$\begin{pmatrix} 2 & 1 & -1 \\ -3 & -1 & 2 \\ -2 & 1 & 2 \end{pmatrix} \qquad \begin{pmatrix} 2 \\ -3 \\ -2 \end{pmatrix} \div 2 \longrightarrow L \; \begin{pmatrix} 1 & \square & \square \\ -\frac{3}{2} & \square & \square \\ -1 & \square & \square \end{pmatrix}$$

가우스-조단 소거법

$$\begin{pmatrix} 2 & 1 & -1 \\ 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 2 & 1 \end{pmatrix} \qquad \begin{pmatrix} 0 \\ \frac{1}{2} \\ 2 \end{pmatrix} \div \frac{1}{2} \longrightarrow \begin{pmatrix} 1 & 0 & \square \\ -\frac{3}{2} & 1 & \square \\ -1 & 4 & \square \end{pmatrix}$$

$$U \begin{pmatrix} 2 & 1 & -1 \\ 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & -1 \end{pmatrix} \qquad \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix} \div -1 \longrightarrow \begin{pmatrix} 1 & 0 & 0 \\ -\frac{3}{2} & 1 & 0 \\ -1 & 4 & 1 \end{pmatrix}$$

$$A = LU$$

$$\begin{pmatrix} 2 & 1 & -1 \\ -3 & -1 & 2 \\ -2 & 1 & 2 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ -\frac{3}{2} & 1 & 0 \\ -1 & 4 & 1 \end{pmatrix} \begin{pmatrix} 2 & 1 & -1 \\ 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & -1 \end{pmatrix}$$

인공지능과 선형대수

❖ LU 분해법을 이용한 $Ax = LUx = b$ 의 해 구하기

예시) $Ax = LUx = b$

$$\begin{pmatrix} 2 & 1 & -1 \\ -3 & -1 & 2 \\ -2 & 1 & 2 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ -\frac{3}{2} & 1 & 0 \\ -1 & 4 & 1 \end{pmatrix}\begin{pmatrix} 2 & 1 & -1 \\ 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & -1 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

단계 1. $Ly = b$ 인 y 구하기

$$\begin{pmatrix} 1 & 0 & 0 \\ -\frac{3}{2} & 1 & 0 \\ -1 & 4 & 1 \end{pmatrix}\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} 1 \\ \frac{7}{2} \\ -10 \end{pmatrix}$$

단계 2. $Ux = y$ 인 x 구하기

$$\begin{pmatrix} 2 & 1 & -1 \\ 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & -1 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ \frac{7}{2} \\ -10 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}\boxed{\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 7 \\ -3 \\ 10 \end{pmatrix}}$$

인공지능과 선형대수

❖ LU 분해 : 공간 복잡도 (Space Complexity)

✓ **LU 분해 에서의 공간에 대한 효율성**

- **L U 행렬을 하나의 행렬로 표현하여 공간 절약**

컴퓨터 기억장치

$$A = LU$$

$$\begin{bmatrix} ☆ & ☆ & ☆ \\ ☆ & ☆ & ☆ \\ ☆ & ☆ & ☆ \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ \blacksquare & 1 & 0 \\ \blacksquare & \blacksquare & 1 \end{bmatrix} \begin{bmatrix} \blacklozenge & \blacklozenge & \blacklozenge \\ 0 & \blacklozenge & \blacklozenge \\ 0 & 0 & \blacklozenge \end{bmatrix} \longleftrightarrow \begin{bmatrix} \blacklozenge & \blacklozenge & \blacklozenge \\ \blacksquare & \blacklozenge & \blacklozenge \\ \blacksquare & \blacksquare & \blacklozenge \end{bmatrix}$$

$$\begin{pmatrix} 2 & 1 & -1 \\ -3 & -1 & 2 \\ -2 & 1 & 2 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ -\frac{3}{2} & 1 & 0 \\ -1 & 4 & 1 \end{pmatrix} \begin{pmatrix} 2 & 1 & -1 \\ 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & -1 \end{pmatrix} \longleftrightarrow \begin{pmatrix} 2 & 1 & -1 \\ -\frac{3}{2} & \frac{1}{2} & \frac{1}{2} \\ -1 & 4 & -1 \end{pmatrix}$$

인공지능과 선형대수

❖ LU 분해 : 시간 복잡도 (Time Complexity)

✓ 문제해결($Ax = y$ 의 해 구하기)을 위한 복잡도 비교

계산 방법	Big-O 표기법
가우스-조단 소거법	$O(n^3)$
LU 분해	$O(n^3)$
LU 분해 후의 단계 1	$O(\frac{n^2}{2})$
LU 분해 후의 단계 2	$O(\frac{n^2}{2})$

단지 1 번 필요

LU 분해후 2 단계가 필요함

정 리

벡터, 행렬

특징	연산	특수형태
노름, 거리, 차원, 내적, 행렬식	벡터합, 스칼라곱, 행렬합, 행렬곱	영행렬, 음행렬, 단위행렬, 역행렬, 대각행렬, 전치행렬 등

선형문제
- 선형결합
- 기본행변환
- 가우스-조르단 소거법

알고리즘
- LU분해
- QR분해
- 최소제곱법
- 그람-슈미트 방법

행렬의 특성값
- 고윳값&고유벡터
- 특잇값
  - 특잇값분해 (SVD)