선형대수학팀

3팀
윤지영
이지윤
이수린
임지훈
채희지

INDEX

1. 선형대수학의 개념과 필요성
2. 벡터와 행렬
3. 선형방정식과 선형결합
4. 선형변환
5. 역행렬과 행렬식
6. 아핀변환과 딥러닝

1

선형대수학의 개념과 필요성

1 선형대수학의 개념과 필요성

선형대수학의 개념

'선형대수학'이 뭐라고 생각하세요?

'선형대수학'는 말이죵~

선형성을 연구하는 대수학의 한 분야
기본단위인 행렬과 벡터를 이용하여 문제를
공간적으로 이해 및 처리

⋮

우리가 다루는 데이터가 2차원 이상의 공간일 때
이를 효율적으로 다루기 위한 언어

1 선형대수학의 개념과 필요성

선형대수학의 필요성

선형대수학

- 대수학의 한 분야 → 수많은 연립선형방정식을 행렬로 표현
  벡터를 통해 선형결합으로 표현
- 공간적 개념 → 데이터를 고차원 공간 내의 점으로 해석
  데이터프레임화, 행렬화, 차원 축소 등의 과정들이
  모두 선형대수학에 근간을 둠

1 선형대수학의 개념과 필요성

선형대수학의 필요성

선형대수학

머신러닝 모델의 알고리즘, 딥러닝 구조도 마찬가지

선형대수학은 통계의 기반이기 때문에

데이터 분석에 중요

2

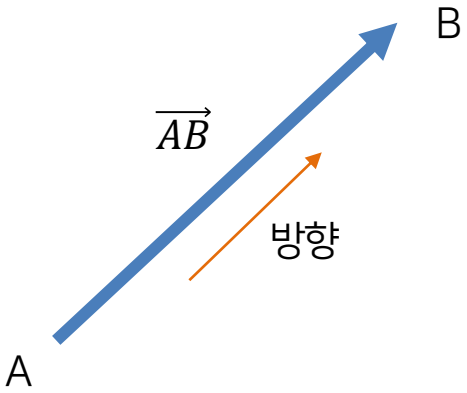
벡터와 행렬

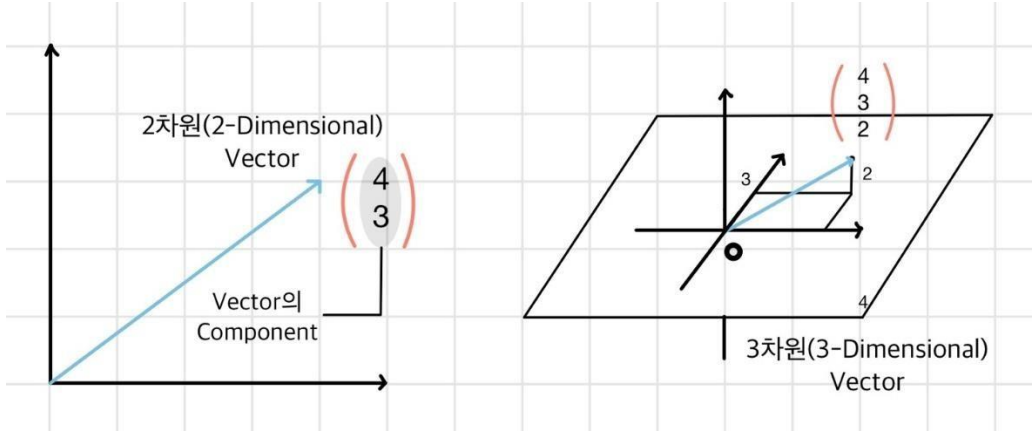
벡터의 의미

벡터 (Vector)

물리적으로 방향과 크기로 구성

공간상에서는 원점을 중심으로 뻗어가는 화살표와 같이 매칭





벡터의 의미

스칼라 (Scalar)

크기만 나타내는 단위로 일반적으로 상수라고 표현

2 벡터와 행렬

벡터의 의미

벡터

- 컴퓨터과학적 의미 →
  - 데이터 형태의 기본 단위
  - 데이터분석에서 다루는 대부분의 변수가 벡터로 표현
- 공간적 이해 →
  - 처리하고 싶은 수치들을 나열하여 모아 놓은 개념
  - 성분 개수에 따라 벡터의 **차원** 결정
    - **n개의 원소로 이루어진 벡터는 n차원 공간에 있음**

2 벡터와 행렬

벡터의 연산

벡터의 연산은 벡터의 성분(component)끼리 이루어짐

벡터의 연산 법칙

$v=\begin{bmatrix} a \\ b \end{bmatrix}$, $w=\begin{bmatrix} c \\ d \end{bmatrix}$

↓

상수배: $cv=\begin{bmatrix} ca \\ cb \end{bmatrix}$

덧셈: $v+w=\begin{bmatrix} a+c \\ b+d \end{bmatrix}$

뺄셈: $v-w=\begin{bmatrix} a-c \\ b-d \end{bmatrix}$

벡터의 기본 연산

R^n의 벡터 x, y, z와 스칼라 h, k에 대하여 다음이 성립한다.

$$x + y = y + x$$

$$(x + y) + z = x + (y + z)$$

$$x + 0 = x = 0 + x$$

$$x + (-x) = 0 = (-x) + x$$

$$k(x + y) = kx + ky$$

$$(h + k)x = hx + hy$$

$$(hk)x = h(kx)$$

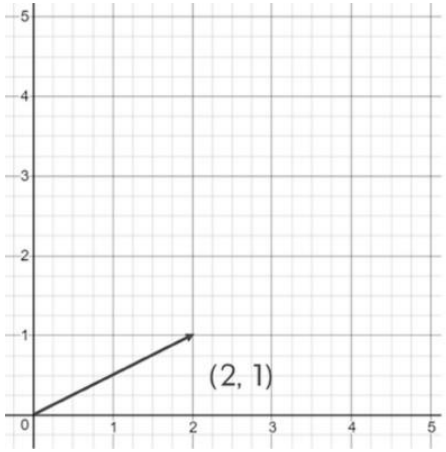
$$1x = x$$

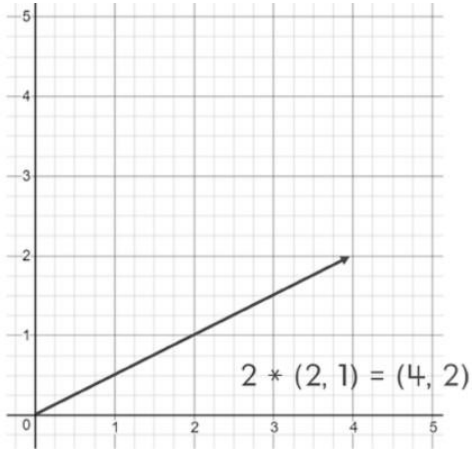
벡터 연산의 기하학적 이해

상수배

각각의 성분에 c를 곱해주는 형태

벡터의 길이를 c배 늘려주는 형태



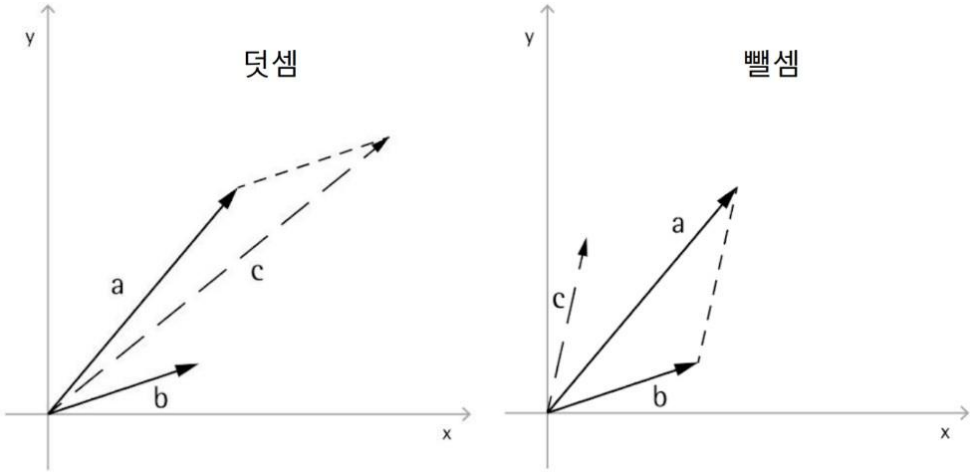


C가 양수인지, 음수인지에 따라 벡터의 방향이 바뀜

벡터 연산의 기하학적 이해

덧셈/뺄셈

원점에서부터 u벡터만큼 화살표 방향으로 이동 후
v만큼 이동하면서 생기는 **평행사변형의 대각선**이 u+v 즉, 벡터 w



뺄셈도 덧셈과 같은 방식으로 u벡터만큼 이동 후 **–v** 만큼 이동하면 벡터 u-v 구할 수 있음

2 벡터와 행렬

행렬의 개념

행렬 (Matrix)

수나 식을 직사각형 모양의 행과 열로 배열한 것

원소(element): 성분(entry)라고도 불리며 행과 열의 개수를 이용해 크기를 표현

원소(element)

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}$$

m x n 행렬

$m \neq n$

$n \; x \; n$ 일 때

정사각행렬 (Square Matrix)

$m \; x \; n$ 일 때

직사각행렬 (Rectangular Matrix)

⋮

정사각행렬 ⊂ 직사각행렬

행렬의 연산

상수배

모든 원소에 **같은 상수**를 곱함

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

$$3A = \begin{bmatrix} 1*3 & 3*3 & -2*3 \\ 3*3 & 1*3 & 2*3 \end{bmatrix} = \begin{bmatrix} 3 & 9 & -6 \\ 9 & 3 & 6 \end{bmatrix}$$

행렬의 합

크기가 같은 두 행렬에서
같은 위치에 있는 원소들의 합

행렬의 곱

모든 원소에 **같은 상수**를 곱함

행렬의 연산

상수배

모든 원소에 **같은 상수**를 곱함

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

$$B = \begin{bmatrix} 2 & 4 & -5 \\ -1 & 2 & 3 \end{bmatrix}$$

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

$$= \begin{bmatrix} 3 & 9 & -6 \\ 9 & 3 & 6 \end{bmatrix}$$

행렬의 합

크기가 같은 두 행렬에서

같은 위치에 있는 원소들의 합

행렬의 곱

모든 원소에 **같은 상수**를 곱함

$$A + B$$

$$= \begin{bmatrix} 1+2 & 3+4 & -2-5 \\ 3-1 & 1+2 & 2+3 \end{bmatrix}$$

$$= \begin{bmatrix} 3 & 7 & -7 \\ 2 & 3 & 5 \end{bmatrix}$$

$$AC = \begin{bmatrix} 9 & 11 \\ -3 & 10 \end{bmatrix}$$

행렬의 연산

$$A = \begin{bmatrix} 1 & 3 & -2 \\ 3 & 1 & 2 \end{bmatrix}, C = \begin{bmatrix} 2 & -1 \\ 3 & 2 \\ -1 & 3 \end{bmatrix}$$

AC

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

$$= \begin{bmatrix} 9 & 11 \\ -3 & 10 \end{bmatrix}$$

행렬의 곱

앞의 **행렬의 열**과
뒤 **행렬의 행**이 같을 때,
원소의 곱의 합

2 벡터와 행렬

행렬의 종류

한 번 알아보자 ~

행렬의 종류는 행렬의 형태 또는 구성 원소에 따라 종류가 달라짐

영행렬(zero matrix)

모든 원소가 0인 행렬

$$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}$$

대각행렬 (diagonal matrix)

대각 성분을 제외한 다른 성분이 모두 0인 행렬

$$\begin{bmatrix} a_{11} & 0 & 0 \\ 0 & a_{22} & 0 \\ 0 & 0 & a_{33} \end{bmatrix}$$

단위행렬 (identity matrix)

대각 성분(diagonal)이 모두 1이고 나머지 성분들은 0인 행렬

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

2 벡터와 행렬

행렬의 종류

한 번 알아보자 ~

행렬의 종류는 행렬의 형태 또는 구성 원소에 따라 종류가 달라짐

전치행렬 (Transpose matrix)

행과 열을 교환하여 얻는 행렬

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \qquad A^T = \begin{bmatrix} a_{11} & a_{21} & a_{31} \\ a_{12} & a_{22} & a_{32} \\ a_{13} & a_{23} & a_{33} \end{bmatrix}$$

대칭행렬 (Symmetric matrix)

A= A^T 인 행렬

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \\ 3 & 6 & 1 \end{bmatrix}$$

3

선형방정식과 선형결합

3 선형방정식과 선형결합

선형방정식(Linear Equation)

선형방정식 (Linear Equation)

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n = b$$

하나 이상의 선형방정식 집합은 **연립선형방정식**

해를 구할 때 문자와 식의 개수가 맞지않거나, 일반화된 해를 찾고 싶을 때 선형방정식 사용

EXAMPLE

$\begin{cases} 2x - y = 0 \\ -x + 2y = 3 \end{cases}$ 의 해 구하기: $x\begin{bmatrix}2\\1\end{bmatrix} + y\begin{bmatrix}-1\\2\end{bmatrix} = \begin{bmatrix}0\\3\end{bmatrix}$

$\rightarrow Ax = b$의 꼴: $\begin{bmatrix}2 & -1\\-1 & 2\end{bmatrix}\begin{bmatrix}x\\y\end{bmatrix} = \begin{bmatrix}0\\3\end{bmatrix}$

$\rightarrow$ 일반화: $x\begin{bmatrix}2\\1\end{bmatrix} + y\begin{bmatrix}-1\\2\end{bmatrix} = \begin{bmatrix}b_1\\b_2\end{bmatrix}$

3 선형방정식과 선형결합

선형방정식(Linear Equation)

미지수의 개수가 적고, 방정식의 개수가 적으면?

선형방정식 (Linear Equation)

단순 가감법 or 대입법 사용 가능

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n = b$$

하나 이상의 선형방정식의 집합은 **연립선형방정식**

해를 구할 때 문자와 식의 개수가 맞지않거나, 일반화된 해를 찾고 싶을 때 선형방정식 사용

미지수/방정식의 개수가 늘어나면?

EXAMPLE

$$a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1$$

$$\begin{cases} 2x - y = 0 \\ -x + 2y = 3 \end{cases} \quad \text{일반화하자} \quad x\begin{bmatrix}2\\1\end{bmatrix} + y\begin{bmatrix}-1\\2\end{bmatrix} = \begin{bmatrix}0\\3\end{bmatrix}$$

$$a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2$$

$$\vdots$$

$\rightarrow Ax = b$의 꼴: $\begin{bmatrix}2 & -1\\ -1 & 2\end{bmatrix}\begin{bmatrix}x\\y\end{bmatrix} = \begin{bmatrix}0\\3\end{bmatrix}$

$$a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n = b_m$$

$\rightarrow$ 일반화: $x\begin{bmatrix}2\\1\end{bmatrix} + y\begin{bmatrix}-1\\2\end{bmatrix} = \begin{bmatrix}b_1\\b_2\end{bmatrix}$

Elementary Row Operation (ERO)

Elementary Row Operation (ERO)

① 두 행을 교환한다

② 행에 대한 상수배

③ 행에 대한 상수배를 진행한 후 다른 행과 더한다

고차원 선형시스템의 해를 쉽게 구할 수 있음

⋮

두 행 교환

$$x + y + 2z = 2$$
$$2x + y - z = 1$$

행에 대한 상수배

$$x + y + 2z = 2$$
⋮ x 2
$$\mathbf{2x + 2y + 4z = 4}$$

상수배 후 더하기

$$2x + 2y + 4z = 4$$
$$-1(2x + y - z) = -1$$
⋮
$$\mathbf{y + 5z = 3}$$

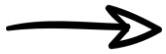
행렬사다리꼴 (Row Echelon Form, REF)

행렬사다리꼴 (Row Echelon Form)

ERO를 이용해 계단형으로 간단히 만든 선형시스템

확장행렬 (Argumented Matrix)

$$\begin{cases} x_1 + 0x_2 + 2x_3 + 5x_4 = 6 \\ 0x_1 + 4x_2 + 3x_3 + 5x_4 = 1 \\ 0x_1 + 0x_2 + 2x_3 + 3x_4 = 5 \\ 0x_1 + 0x_2 + 0x_3 + 2x_4 = 7 \end{cases}$$



Row Echelon Form, REF

$$\begin{bmatrix} 1 & 0 & 2 & 5 & 6 \\ 0 & 4 & 3 & 5 & 1 \\ 0 & 0 & 2 & 3 & 5 \\ 0 & 0 & 0 & 2 & 7 \end{bmatrix}$$

행렬사다리꼴 성립 조건

REF의 세 가지 조건

① 0으로만 구성된 행은 0이 아닌 값이 존재하는 행보다 아래에 위치

② 각 행의 leading entry는 해당 행의 위에 있는 leading entry보다 오른쪽에 위치

③ 열 내에서 leading entry보다 아래에 있는 값들은 모두 0

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 2 \\ 0 & 0 & 0 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 2 \\ 0 & 0 & 0 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 2 \\ 0 & 0 & 0 \end{bmatrix}$$

3 선형방정식과 선형결합

행렬사다리꼴 성립 조건

REF의 세 가지 조건

① 0으로만 구성된 행은 0이 아닌 값이 존재하는 행보다 아래에 위치

② 각 행의 leading entry는 해당 행의 위에 있는 leading entry보다 오른쪽에 위치

③ 열 내에서 leading entry보다 아래에 있는 값들은 모두 0

행이 모두 0으로 구성되지 않았다면,

그 행에서 첫 번째로 나타나게 되는 0이 아닌 수를 의미

1일 경우 모든 열에 최대 1개 존재

leading 1

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

Reduced Row Echelon Form, RREF

Reduced Row Echelon Form (RREF)

행렬사다리꼴 조건에 **2가지 조건 추가**

모든 행렬에 하나씩 존재

④ 0으로만 구성되지 않은 행의 leading entry가 1이다 (leading 1)

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

⑤ 각각의 **leading 1**은 해당 열에서 유일한 0이 아닌 값이다.

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

가우스 – 조던 소거법 (Gauss – Jordan Elimination)

가우스 – 조던 소거법

ERO를 통해 선형시스템을
RREF로 변환해 연립방정식의 해를 구하는 방법

① **성분이 모두 0인 행렬**을 가장 아래에 위치
② **첫 번째 열의 수가 0이 아닌 행**을 제일 위에 위치
③ 다른 행의 **첫 번째 열의 수가 0이 되도록** 첫 번째 행을 상수배하여 다른 행에 더함
④ 첫 번째 행과 열을 제외한 나머지 부분에 대해서 위의 **과정을 반복**
⑤ 한 개의 열에 **최대 1개의 leading 1만 존재**하도록 아래에서부터 거꾸로 ERO를 진행

3 선형방정식과 선형결합

가우스 – 조던 소거법 (Gauss – Jordan Elimination)

가우스 – 조던 소거법

ERO를 통해 선형시스템을

RREF로 변환해 연립방정식의 해를 구하는 방법

모든 행렬에서 항상 깔끔하게 해를

구할 수 있을까?

② 성분이 모두 0인 행렬을 가장 아래에 위치

째 열의 수가 0이 아닌 행을 제일 위에 위치

인 첫 번째 열의 수가 0이 되도록 첫 번째 행을 상수배하여 다른 행에 더함

행과 열을 제외한 나머지 부분에 대해서 위의 과정을 반복

열에 최대 1개의 leading 1만 존재하도록 아래에서부터 거꾸로 ERO를 진행

항상 성립하지는 않음

수정

3 선형방정식과 선형결합

선형시스템에서의 해집합

선형시스템에서의 세 가지 상황

1) 해집합이 단 하나만 존재 (Unique Solution)
2) 해집합이 무수히 많이 존재 (Infinitely many solution)

→ **해 존재** (consistent)

3) 해집합이 존재하지 않음 (No solution)

→ **해 존재 X** (inconsistent)

시각화

해집합이 단 하나만 존재

해집합이 무수히 많음

해집합 존재 x

3 선형방정식과 선형결합

선형시스템에서의 해집합

선형시스템에서의 세 가지 상황

1) 해집합이 단 하나만 존재 (Unique Solution)
2) 해집합이 무수히 많이 존재 (Infinitely many solution)

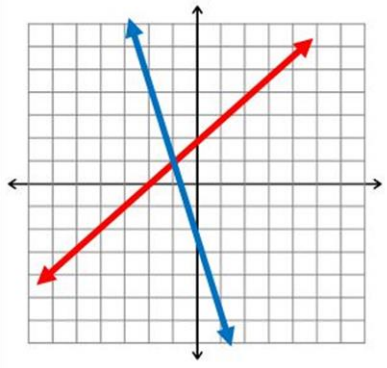
→ 해 존재 (consistent)

3) 해집합이 존재하지 않음 (No solution)

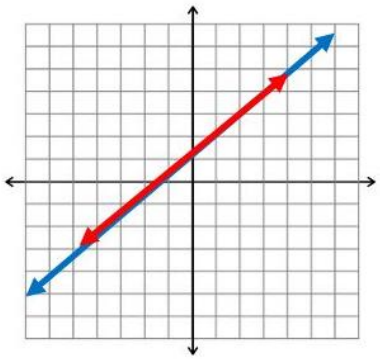
→ 해 존재 X (inconsistent)

시각화

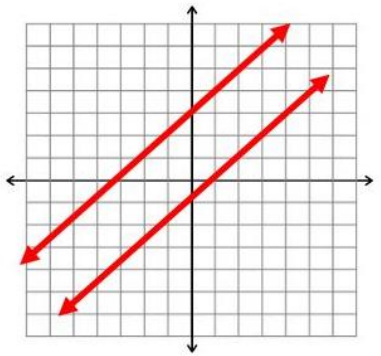
해집합이 단 하나만 존재



해집합이 무수히 많음



해집합 존재 x



3 선형방정식과 선형결합

선형시스템에서의 해집합

해를 깔끔하게 구할 수 있는 경우

$$\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 4 \end{bmatrix} \rightarrow \begin{cases} x_1 = 1 \\ x_2 = 2 \\ x_3 = 4 \end{cases}$$

Pivot column

leading 1을 포함하고 있는 열

각 행별로 leading 1을

얻을 수 있는 경우

해를 깔끔하게 구할 수 없는 경우

$$\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 0 & 4 \end{bmatrix} \rightarrow \begin{cases} x_1 = 1 \\ x_2 = 2 \\ 0 = 4 \end{cases}$$

해가 존재 X

(inconsistent)

Free column

Pivot을 포함하고 있지 않은 열

$$\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 0 & 0 \end{bmatrix} \rightarrow \begin{cases} x_1 = 1 \\ x_2 = 2 \\ x_3 \text{ is free} \end{cases}$$

해가 무수히 많음

(consistent)

3 선형방정식과 선형결합

선형시스템에서의 해집합

해를 깔끔하게 구할 수 있는 경우

$$\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 4 \end{bmatrix} \rightarrow \begin{cases} x_1 = 1 \\ x_2 = 2 \\ x_3 = 4 \end{cases}$$

Pivot column

leading 1을 포함하고 있는 열

각 행별로 leading 1을 얻을 수 있는 경우

해를 깔끔하게 구할 수 없는 경우

$$\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 0 & 4 \end{bmatrix} \rightarrow \begin{cases} x_1 = 1 \\ x_2 = 2 \\ 0 = 4 \end{cases}$$

해가 존재 X

(inconsistent)

Free column

Pivot을 포함하고 있지 않은 열

$$\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 0 & 0 \end{bmatrix} \rightarrow \begin{cases} x_1 = 1 \\ x_2 = 2 \\ x_3 \ is \ free \end{cases}$$

해가 무수히 많음

(consistent)

선형결합(Linear Combination)

선형결합(Linear Combination)

벡터들을 스칼라배와 벡터 덧셈을 통해 조합한 새로운 벡터를 얻는 연산

$$Ax = [a_1\ a_2 \cdots a_n]\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = a_1x_1 + a_2x_2 + \cdots + a_nx_n$$

span : a의 조합

2차원 벡터공간의 모든 벡터는 (0,1)과 (1,0)의 조합을 통해 표현 가능

$'Ax = b'$의 해가 존재한다 = a의 조합으로 b를 표현할 수 있다

Span에 관한 자세한 설명은 2주차에!

4

선형변환

선형변환(Linear Transformation)

선형변환 (Linear Transformation)

R^n에서 R^m 으로 변환하는 T가

① $T(u + v) = T(u) + T(v)$ / ② $T(ku) = k\,T(u)$를 만족할 때,

이를 **선형변환**이라고 함

정의역으로부터 하나의 화살표만 나가야 한다는 함수의 특징 또한 똑같이 적용

$$\begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ 3 \end{bmatrix}$$

$Ax = b$의 해를 찾는 것

A를 곱했을 때 b로 변환되는 벡터 x를 찾는 것

선형변환(Linear Transformation)

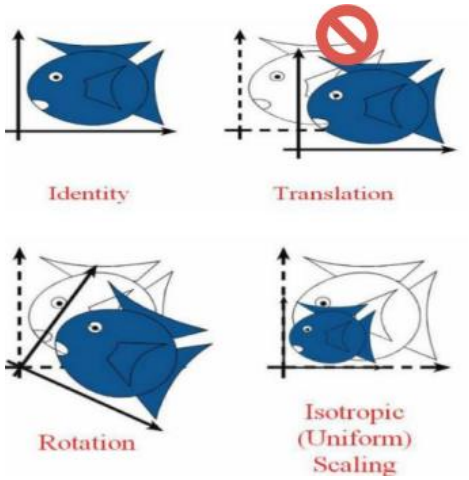
선형변환 (Linear Transformation)

R^n에서 R^m 으로 변환하는 T가

① $T(u + v) = T(u) + T(v)$ / ② $T(ku) = k\,T(u)$를 만족할 때,

이를 **선형변환**이라고 함

변환의 4가지 예시



Translation의 경우, **선형 변환이 아님**!

⋮

$f(x) = ax + b$와 같이 상수항 b를 포함하므로

선형의 조건을 만족하지 않음

4 선형변환

선형변환(Linear Transformation)

즉, 선형변환이란…

정의역에서 원소 두 개를 뽑아 선형결합해 나오는 함수(벡터)

= 선형 결합을 나중에 적용했을 때의 함수 값

↓

기하학적 관점

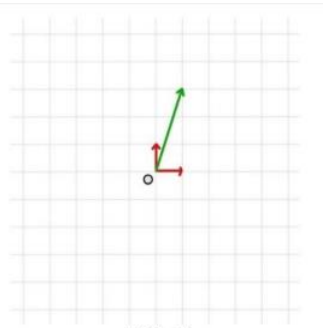
1) 공간 위의 모든 직선은 **변환 이후에도 직선**

2) 원점은 변환 이후에도 **그대로 원점**

4 선형변환

선형변환의 기하학적 의미

원래 좌표계를 오른쪽으로 회전 후 늘린 형태

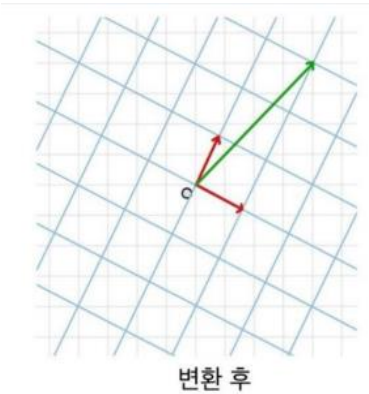


변환 전

기저 벡터(basis vector, **빨간색**): 좌표계의 기본 $\binom{1}{0}, \binom{0}{1}$

변환전

초록색 벡터를 기저 벡터의 선형결합으로 표현: $1\begin{bmatrix}1\\0\end{bmatrix} + 3\begin{bmatrix}0\\1\end{bmatrix}$



변환 후

변환된 기저 벡터: $\begin{bmatrix}2\\-1\end{bmatrix}, \begin{bmatrix}1\\2\end{bmatrix}$

변환된 초록색 벡터는 $1\begin{bmatrix}2\\-1\end{bmatrix} + 3\begin{bmatrix}1\\2\end{bmatrix} = \begin{bmatrix}5\\5\end{bmatrix}$

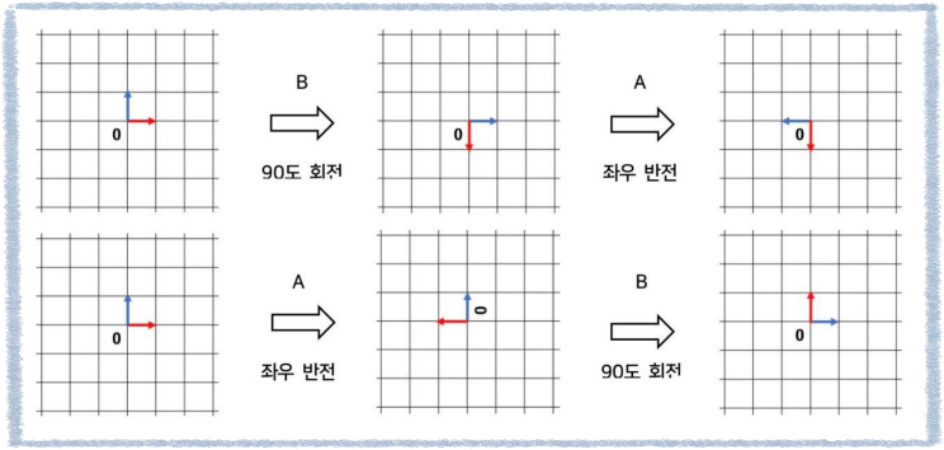
Basis에 관한 자세한 설명은 2주차에!

4 선형변환

선형변환의 기하학적 의미

하나의 행렬이 하나의 Transformation을 설명

→ 두 행렬의 곱은 두 번 변환을 적용



A: 공간 90도 회전 / B: 공간을 좌우로 뒤집음

AB = 공간을 좌우로 뒤집을 다음, 90도 회전시키는 선형 변환

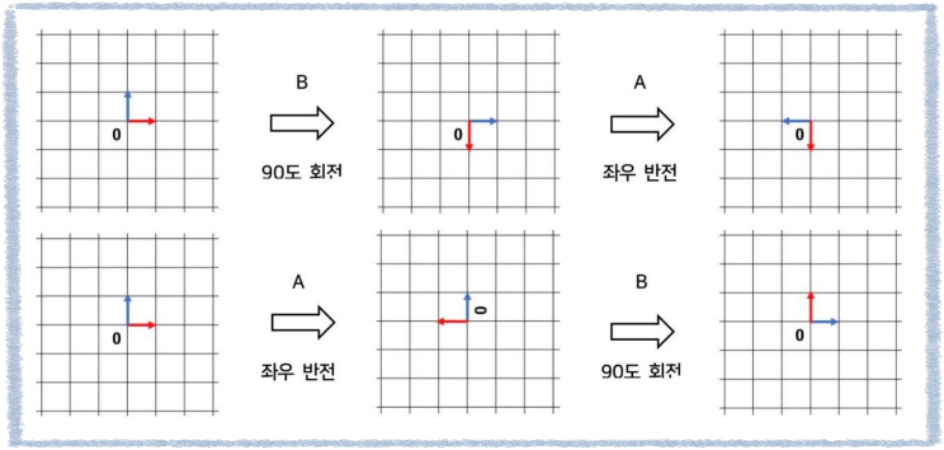
BA = 공간을 90도 회전시킨 다음, 좌우로 뒤집는 선형 변환

4 선형변환

선형변환의 기하학적 의미

하나의 행렬이 하나의 Transformation을 설명

→ 두 행렬의 곱은 두 번 변환을 적용



A: 공간 90도 회전 / B: 공간을 좌우로 뒤집음

AB = 공간을 좌우로 뒤집은 다음, 90도 회전시키는 선형변환

BA = 공간을 90도 회전시킨 다음, 좌우로 뒤집는 선형변환

행렬 연산에서 교환 법칙이 성립하지 않음!

5

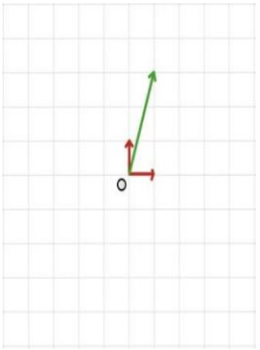
역행렬과 행렬식

5 역행렬과 행렬식

역행렬의 기하학적 의미

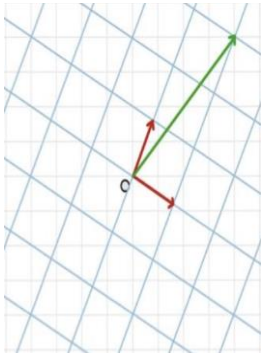
역행렬

A의 선형변환
(linear transformation)을
다시 되돌리는 행렬 A^{-1}



① 행렬 A

② 역행렬 A^{-1}



예시

① 벡터 X가 벡터 b로
선형변환

$$Ax = b$$

$$\begin{bmatrix} 2 & 1 \\ -1 & 2 \end{bmatrix}\begin{bmatrix} 1 \\ 3 \end{bmatrix} = \begin{bmatrix} 5 \\ 5 \end{bmatrix}$$

② 벡터 b가 벡터 X로
선형변환

$$A^{-1}\,b = x$$

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

5 역행렬과 행렬식

역행렬의 기하학적 의미

역행렬

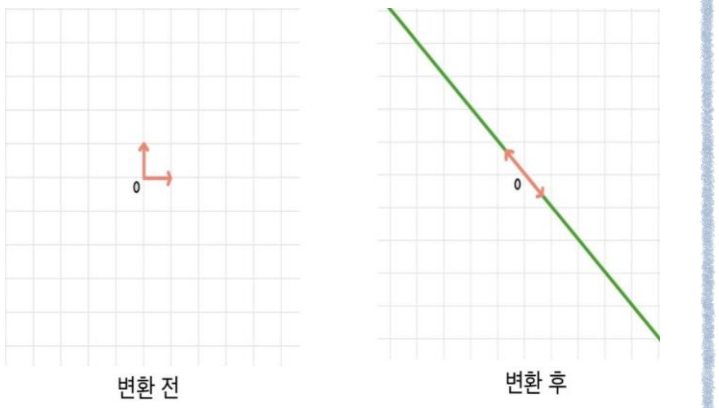
존재 → ① '$Ax = b$'가 유일한 해를 가짐
② 특정 x를 선형변환한 Ax가 유일
③ x와 Ax가 서로 일대일 대응

존재 x → 공간을 **압축시키는** 선형 변환을 가한 경우

5 역행렬과 행렬식

역행렬의 기하학적 의미

R^2 공간이 초록색 선으로 압축된 경우



공간을 **압축시키**는 선형 변환을 가한 경우

2차원에서 1차원으로 압축된 벡터가

어떤 2차원 벡터로 다시 변환되는지 알 수 없음

(함수: 하나의 x벡터는 특정 y벡터로만 반환되어야 함)

Ax = b의 해가 유일하지 않다

5 역행렬과 행렬식

역행렬의 기하학적 의미

역행렬

존재 →
① '$Ax = b$'가 유일한 해를 가짐
② 특정 x를 선형변환한 Ax가 유일
③ x와 Ax가 서로 일대일 대응

존재 x → 공간을 **압축시키는** 선형 변환을 가한 경우
x와 Ax가 서로 일대일 대응이 아님 ✓

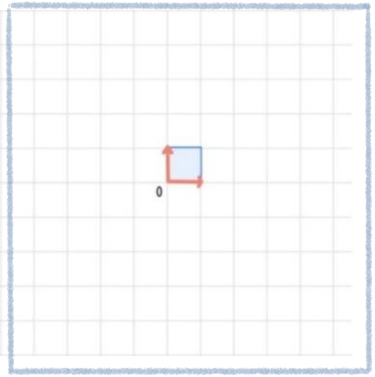
5 역행렬과 행렬식

행렬식의 기하학적 의미

행렬식 (Determinant)

선형 변환을 할 때 선형 변환이 공간을 얼마나 변환시키는지에 대한 정보 제공

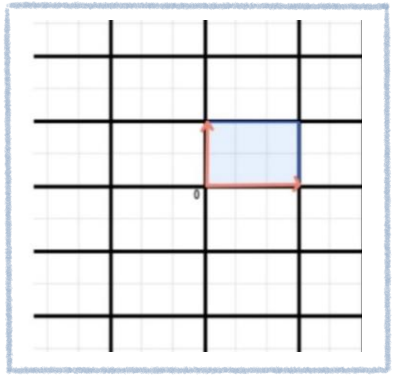
① 행렬식의 값 : 변환의 크기



$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ ⇢ $\begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix}$

넓이 : 1 넓이 : 6

② 행렬식의 부호 : 변환의 방향



$\begin{bmatrix} 2 & 0 \\ 0 & -3 \end{bmatrix}$ 음수 ⇢ 이 상황에서는 **좌우반전**

행렬식 : -6

5 역행렬과 행렬식

특수한 행렬의 행렬식

det I = 1

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

아무것도 하지 않는 행렬,
공간의 크기도 그대로 유지

det AB = det A * det B

B에서의 변화율에
A에서의 변화율 적용,
AB에서의 변화율과 동일

det A = 0

역행렬이 존재하지 않는 행렬,
즉 공간을 압축시키는 행렬

det A^{-1} = 1 / det A

행렬 A로 확대/축소된 공간을
반대로 축소/확대

6

아핀변환과 딥러닝

6 아핀변환과 딥러닝

이동변환과 아핀변환

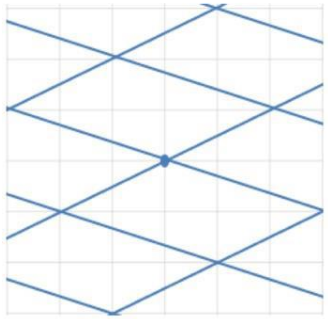
아핀 변환

선형 변환

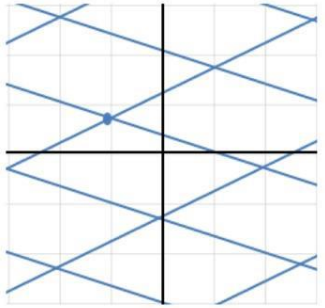
$$y = Ax + b$$

평행이동
(선형성 X)

변환 전



선형변환



아핀변환

예시

① 선형 변환에 $\begin{bmatrix} c \\ d \end{bmatrix}$ 를 추가

$$\begin{bmatrix} 2 & 1 \\ -1 & 2 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} + \begin{bmatrix} c \\ d \end{bmatrix}$$

② Linear combination 형태

$$a \begin{bmatrix} 2 \\ -1 \end{bmatrix} + b \begin{bmatrix} 1 \\ 2 \end{bmatrix} + 1 \begin{bmatrix} c \\ d \end{bmatrix}$$

③ matrix화

$$\begin{bmatrix} 2 & 1 & c \\ -1 & 2 & d \end{bmatrix} \begin{bmatrix} a \\ b \\ 1 \end{bmatrix}$$

6 아핀변환과 딥러닝

이동변환과 아핀변환

아핀 변환

평행이동으로 인해 원점이 변하기 때문에

선형성을 만족하지 못함!

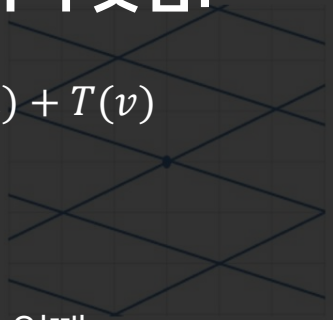
선형 변환

$$y = Ax + b$$

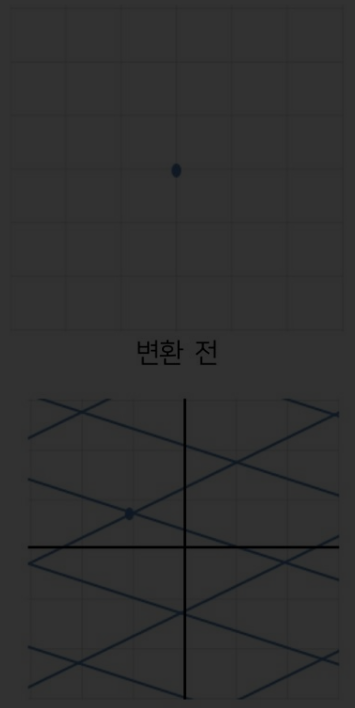
평행이동

(선형성 X)

$$T(u+v) = T(u) + T(v)$$



선형변환



아핀변환

$\boldsymbol{T(X) = AX + \mathbf{k}}$일때,

예시

① 선형 변환에 $\begin{bmatrix} c \\ d \end{bmatrix}$를 추가 ② linear combination 형태 (Matrix)가

$$\begin{bmatrix} 2 & 1 \\ -1 & 2 \end{bmatrix}\begin{bmatrix} a \\ b \end{bmatrix} + \begin{bmatrix} c \\ d \end{bmatrix}$$

$$T(au+bv) = A(au+bv) + k = Aau + Abv + \boldsymbol{k}$$

$$\neq$$

$$a\begin{bmatrix} 2 \\ -1 \end{bmatrix} + b\begin{bmatrix} 1 \\ 2 \end{bmatrix} + 1\begin{bmatrix} c \\ d \end{bmatrix}$$

$$T(au) + T(bv) = (Aau + k) + (Abv + k) = Aau + Abv + \boldsymbol{2k}$$

$$\begin{bmatrix} 2 & 1 & c \\ -1 & 2 & d \end{bmatrix}\begin{bmatrix} a \\ b \\ 1 \end{bmatrix}$$

아핀 변환

6 아핀변환과 딥러닝

이동변환과 아핀변환

Bias 항이 추가되면서 input이 2차원 → 3차원으로 변환

아핀 변환은 Non-Square Matrix를 통해
3차원 input을 2차원 output으로 바꿔 줌

예시

① 선형 변환에 $\begin{bmatrix} c \\ d \end{bmatrix}$ 를 추가

$$\begin{bmatrix} 2 & 1 \\ -1 & 2 \end{bmatrix}\begin{bmatrix} a \\ b \end{bmatrix} + \begin{bmatrix} c \\ d \end{bmatrix}$$

② Linear combination 형태

$$a\begin{bmatrix} 2 \\ -1 \end{bmatrix} + b\begin{bmatrix} 1 \\ 2 \end{bmatrix} + 1\begin{bmatrix} c \\ d \end{bmatrix}$$

③ matrix화

$$\begin{bmatrix} 2 & 1 & c \\ -1 & 2 & d \end{bmatrix}\begin{bmatrix} a \\ b \\ 1 \end{bmatrix}$$

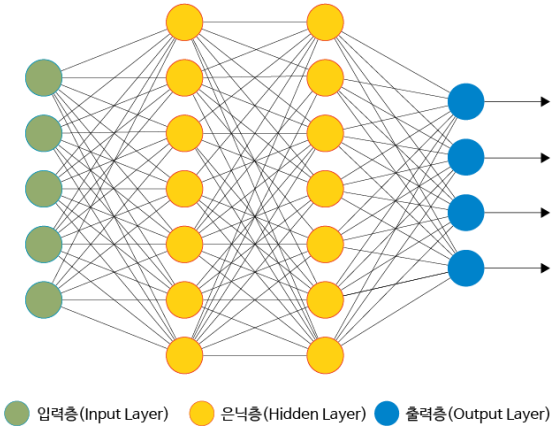
6 아핀변환과 딥러닝

딥러닝 개념

자세한 내용은 딥러닝팀 참고!

딥러닝

머신 러닝의 일종으로, 여러 개의 노드를 가진
층 (Layer)을 쌓아서 학습
Input layer와 hidden layer를 거침



활성화 함수로 입력 값 예측

손실 함수로 예측 값과 실제의 오차 측정

손실함수 바탕으로 가중치 업데이트

선형대수학의 활용

인풋 레이어에 있는 인풋 값 x 가중치 + bias

Wx + b는 아핀 변환의 Ax+ b 구조와 유사

6 아핀변환과 딥러닝

딥러닝 개념

자세한 내용은 딥러닝팀 참고!

딥러닝

머신 러닝의 일종으로, 여러 개의 노드를 가진
층 (Layer)을 쌓아서 학습
Input layer와 hidden layer를 거침

활성화 함수로 입력 값 예측

손실 함수로 예측 값과 실제의 오차 측정

손실함수 바탕으로 가중치 업데이트

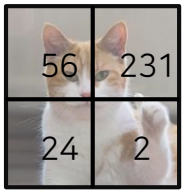
선형대수학의 활용

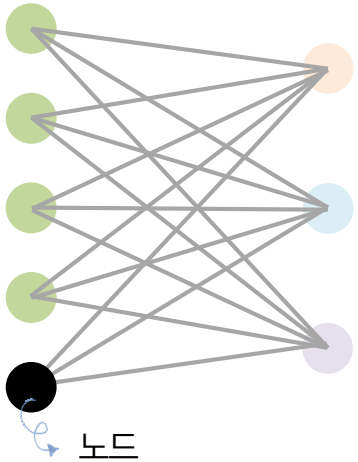
인풋 레이어에 있는 인풋 값 x 가중치 + bias

Wx + b는 아핀 변환의 Ax+ b 구조와 유사

아핀변환과 딥러닝

e.g) 고양이를 인식하는 방법





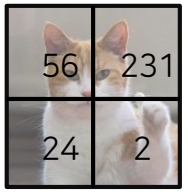
0.2	-0.5	0.1	2	56	1.1	- 96.8
1.5	1.3	2.1	1	231	+ 3.2	= 439.9
-0.2	0.3	0.7	-1.3	24	-1.2	71.1
				2		

= 56	0.2	+ 231	-0.5	+ 24	0.1	+2	2	+1	1.1
	1.5		1.3		2.1		1		3.2
	-0.2		0.7		0.7		-1.3		-1.2

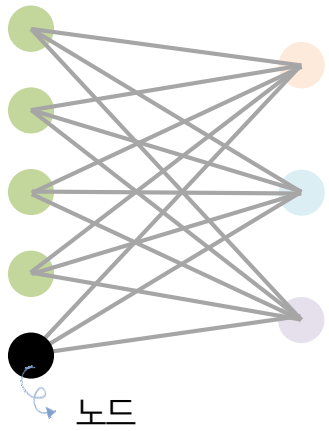
=	0.2	-0.5	0.1	2	1.1	56
	1.5	1.3	2.1	1	3.2	231
	-0.2	0.3	0.7	-1.3	-1.2	24
						2
						1

선형대수학 딥러닝 적용 예시

아핀변환과 딥러닝



고양이 인식



0.2	-0.5	0.1	2	56			
1.5	1.3	2.1	1	231	+	1.1	=
-0.2	0.3	0.7	-1.3	24		3.2	
				2		-1.2	

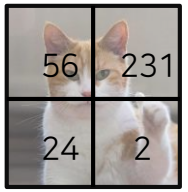
- 96.8
439.9
71.1

4개의 초록색 ● 노드가 벡터 성분 값을 받음

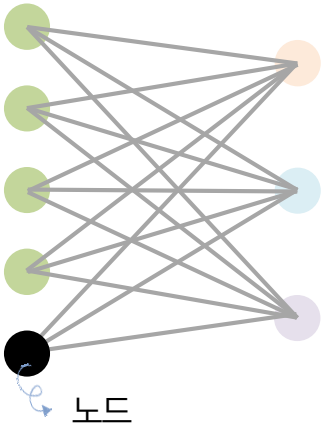
● 노드: Bias

■■■항:각 노드에서 뻗어나가는 가중치

아핀변환과 딥러닝



고양이 인식



0.2	-0.5	0.1	2	56		1.1	-96.8
1.5	1.3	2.1	1	231	+	3.2	= 439.9
-0.2	0.3	0.7	-1.3	24		-1.2	71.1
				2			

4개의 초록색 ● 노드가 벡터 성분 값을 받음

4개의 특성을 가진 데이터를 포함한 4차원 벡터가 들어옴

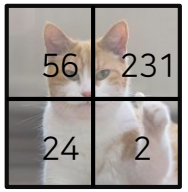
가중치 행렬 $\times$ $input$ 벡터 $+$ $bias$

$\rightarrow$ $Wx + b$ (아핀 변환의 형태)

● 노드: Bias

■■■항:각 노드에서 뻗어나가는 가중치

아핀변환과 딥러닝



고양이 인식

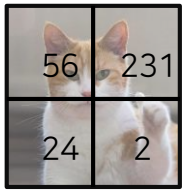
$$= 56 \begin{bmatrix} 0.2 \\ 1.5 \\ -0.2 \end{bmatrix} + 231 \begin{bmatrix} -0.5 \\ 1.3 \\ 0.7 \end{bmatrix} + 24 \begin{bmatrix} 0.1 \\ 2.1 \\ 0.7 \end{bmatrix} + 2 \begin{bmatrix} 2 \\ 1 \\ -1.3 \end{bmatrix} + 1 \begin{bmatrix} 1.1 \\ 3.2 \\ -1.2 \end{bmatrix}$$

$$= \begin{bmatrix} 0.2 & -0.5 & 0.1 & 2 & 1.1 \\ 1.5 & 1.3 & 2.1 & 1 & 3.2 \\ -0.2 & 0.3 & 0.7 & -1.3 & -1.2 \end{bmatrix} \begin{bmatrix} 56 \\ 231 \\ 24 \\ 2 \\ 1 \end{bmatrix}$$

증강시킨(Augmented) 행렬을 이용하여

linear transformation 형태로 나타낼 수도 있음

아핀변환과 딥러닝



고양이 인식

$$= 56 \begin{bmatrix} 0.2 \\ 1.5 \\ -0.2 \end{bmatrix} + 231 \begin{bmatrix} -0.5 \\ 1.3 \\ 0.7 \end{bmatrix} + 24 \begin{bmatrix} 0.1 \\ 2.1 \\ 0.7 \end{bmatrix} + 2 \begin{bmatrix} 2 \\ 1 \\ -1.3 \end{bmatrix} + 1 \begin{bmatrix} 1.1 \\ 3.2 \\ -1.2 \end{bmatrix}$$

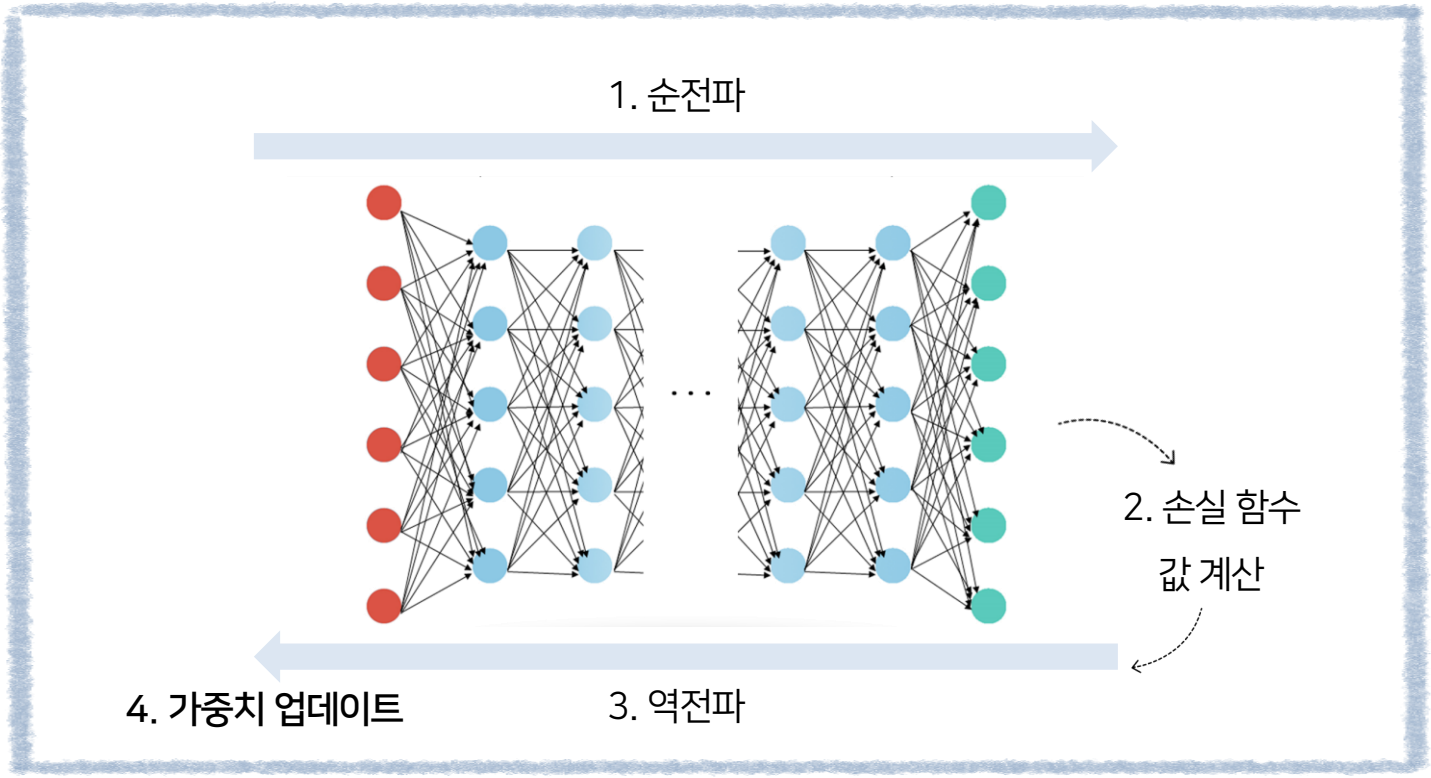
$$= \begin{bmatrix} 0.2 & -0.5 & 0.1 & 2 & 1.1 \\ 1.5 & 1.3 & 2.1 & 1 & 3.2 \\ -0.2 & 0.3 & 0.7 & -1.3 & -1.2 \end{bmatrix} \begin{bmatrix} 56 \\ 231 \\ 24 \\ 2 \\ 1 \end{bmatrix}$$

[56, 231, 24, 2, 1] 벡터에 가중치 행렬을 통한 Linear Transformation이 가해짐으로써 다음 층으로 전달

→ 가중치를 계속 업데이트 해 나가는 것이 딥러닝의 학습 과정

Again 자세한 내용은 딥러닝팀 참고!

아핀변환과 딥러닝



활성화함수를 이용하여 가중치를 업데이트

'비선형' 활성화 함수를 바탕으로 네트워크를 구성

아핀변환과 딥러닝

그래서 왜 비선형 함수를 바탕으로 네트워크를 구성하는 건데!

$f(x) = kx$를 선형 함수, $g(x) = ax + b$를 비선형함수라고 하면

$f(x)$로 n번 층을 쌓는다고 해도 $k^n x$임으로,

k^n을 한번 적용하는 것과 차이가 없음

⋮

→ **여러 hidden layer를 쌓고 가중치를 업데이트하는 이점이 없어짐**

sigmoid, ReLU와 같은 비선형 활성화 함수를 이용해,

층을 쌓는 과정에 의미를 부여

자세한 내용은 딥러닝팀 참고!

다음 주 예고

1. 공간(Space)
2. 노름과 내적
3. 직교성
4. 사상(Projection)
5. 회귀적용