

# 데이터 과학

## L02: Basic Linear Algebra

Kookmin University

# 목차

- 벡터
- 행렬
- NumPy

# 벡터 (vector)

- **정의1: 방향과 크기의 의미를 모두 포함하는 표현 도구**
  - 변위, 속도, 힘, 자기장, 전기장 등 물리적 개념 설명 도구
  - 라틴어 어원은 “운반하다, vehere”
- 반면, 스칼라(Scalar)는 방향성 없이 하나의 **크기**를 나타냄
  - 키, 질량, 온도, 시간 등
  - 라틴어 어원은 “저울, scala”
- 속도(velocity)는 벡터, 속력(speed)은 스칼라
- 무게(weight)는 벡터, 질량(mass)은 스칼라

# 벡터 (vector)

- 정의2: 유한한 차원의 공간에 존재하는 점
- 예)
  - 사람을 (키, 몸무게, 나이)의 3차원 벡터로 표현
  - 데이터과학 수업 학생을 (과제1 점수, 과제2 점수, 과제3 점수, 과제4 점수)의 4차원 벡터로 표현
- Python의 List나 Tuple을 사용하여 벡터 표현 가능

$$\vec{x} = \mathbf{x} = (x_1, x_2, \dots, x_n) = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix}$$

```
person = (180, 62, 22)
grades = (95, 80, 75, 62)
or
person = [180, 62, 22]
grades = [95, 80, 75, 62]
```

# 리스트는 벡터가 아니다

- 벡터는 다차원 공간상의 점으로 정의되므로, 리스트를 벡터처럼 사용하기 위해서는 각 점들끼리의 합, 차, 곱 등의 연산을 정의해야 함

# 벡터의 연산

n 차원의 벡터  $\mathbf{x}$ ,  $\mathbf{y}$ 가 주어질 때, 두 벡터의 합과 스칼라곱은 다음과 같이 정의함

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

$$\mathbf{x} + \mathbf{y} = \begin{bmatrix} x_1 + y_1 \\ x_2 + y_2 \\ \dots \\ x_n + y_n \end{bmatrix}$$

$$k\mathbf{x} = \begin{bmatrix} kx_1 \\ kx_2 \\ \dots \\ kx_n \end{bmatrix}$$

# 벡터의 연산

- 두 벡터의 합/차는 같은 차원의 값끼리 더한/뺀 것 (element-wise sum)
  - 예) 두 학생의 각 과제 점수의 합은?  
 $(1, 2, 3) + (2, 0, 4) = (3, 2, 7)$   
 $(1, 2, 3) - (2, 0, 4) = (-1, 2, -1)$
- 벡터  $\mathbf{x}$ 와 스칼라  $k$ 의 곱은 벡터  $\mathbf{x}$ 의 각 차원의 값을  $k$  배 한 것
  - 예) 가구 크기 단위 변경: cm (센티미터)  $\rightarrow$  m (미터)  
 $(100, 200, 500) \cdot 0.01 = (1, 2, 5)$

# 벡터의 내적 (dot product)

- 두 벡터의 내적 = 각 성분간 곱의 합
  - 예) 학생 점수 벡터와 반영비율 벡터를 곱하여 최종 점수 구하기
  - $(100, 90, 40) \cdot (0.2, 0.2, 0.4) = 20 + 18 + 16 = 54$
- 각 성분의 제곱 값의 합을 구하려면?
- 벡터의 크기를 구하려면?
- 두 벡터 사이의 거리를 구하려면?



# 목차

- 벡터
- 행렬
- NumPy

# 행렬

- 수를 다음과 같이 직사각형 모양의 행과 열로 배열한 것을 행렬(matrix) 이라 하며, 각각의 수를 행렬의 성분(entry)이라고 함
- n행 m열을 가질 경우  $n \times m$  행렬이라고 함
- 같은 크기의 벡터들이 모이면 행렬이 됨
- Python에서 List의 List 혹은 Tuple의 Tuple로 표현 가능

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

```
A = [[1, 2, 3],  
      [4, 5, 6]]
```

```
B = [[1, 2],  
      [3, 4],  
      [5, 6]]
```

# 행렬의 합, 차

- 행렬의 합/차는 같은 위치의 값끼리 더한/뺀 것

$$A = \begin{bmatrix} 2 & 1 & 4 & 0 \\ -7 & 3 & 6 & 1 \\ 8 & -4 & -2 & 3 \\ 1 & 9 & 4 & -2 \end{bmatrix}$$

$$B = \begin{bmatrix} 7 & -1 & 2 & 4 \\ 2 & 8 & -5 & 1 \\ -3 & -4 & 2 & -2 \\ 9 & 6 & -2 & 0 \end{bmatrix}$$

- $A + B = ?$
- $A - B = ?$
- $B - A = ?$

# 행렬과 스칼라의 곱

- 행렬 A와 스칼라 k의 곱은 행렬의 각 요소에 k를 곱한 것
  - 예) 은행 및 연도별 이자수익 = 투자금액 x 은행 및 연도별 이자율

$$kA = k \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{2m} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nm} \end{bmatrix} = \begin{bmatrix} ka_{11} & ka_{12} & \dots & ka_{1m} \\ ka_{21} & ka_{22} & \dots & ka_{2m} \\ \dots & \dots & \dots & \dots \\ ka_{n1} & ka_{n2} & \dots & ka_{nm} \end{bmatrix}$$

$$-4 \begin{bmatrix} -1 & 2 & 5 \\ 3 & 2 & -7 \\ 8 & 3 & 1 \end{bmatrix} = ?$$

# 행렬의 곱

- 행렬의 곱셈은 다음과 같이 정의됨

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{2m} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nm} \end{bmatrix} \times \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1s} \\ b_{21} & b_{22} & \dots & b_{2s} \\ \dots & \dots & \dots & \dots \\ b_{r1} & b_{r2} & \dots & b_{rs} \end{bmatrix} = \begin{bmatrix} c_{11} & \dots & c_{1j} & \dots & c_{1s} \\ \dots & \dots & \dots & \dots & \dots \\ c_{i1} & \dots & c_{ij} & \dots & c_{is} \\ \dots & \dots & \dots & \dots & \dots \\ c_{n1} & \dots & c_{nj} & \dots & c_{ns} \end{bmatrix}$$
$$c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \dots + a_{im}b_{mj} = \sum_{k=1}^m a_{ik}b_{kj}$$

- 예) 시간대 및 기계 종류별 가동시간 x (시간당) 기계 종류 및 자원별 필요량 = 시간대 및 자원별 필요량

# 행렬의 곱

연산

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{2m} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nm} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1s} \\ b_{21} & b_{22} & \dots & b_{2s} \\ \dots & \dots & \dots & \dots \\ b_{m1} & b_{m2} & \dots & b_{ms} \end{bmatrix}$$

$$= \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} + \dots + a_{1m}b_{m1} & a_{11}b_{12} + a_{12}b_{22} + \dots + a_{1m}b_{m2} & \dots & a_{11}b_{1s} + a_{12}b_{2s} + \dots + a_{1m}b_{ms} \\ a_{21}b_{11} + a_{22}b_{21} + \dots + a_{2m}b_{m1} & a_{21}b_{12} + a_{22}b_{22} + \dots + a_{2m}b_{m2} & \dots & a_{21}b_{1s} + a_{22}b_{2s} + \dots + a_{2m}b_{ms} \\ \dots & \dots & \dots & \dots \\ a_{n1}b_{11} + a_{n2}b_{21} + \dots + a_{nm}b_{m1} & a_{n1}b_{12} + a_{n2}b_{22} + \dots + a_{nm}b_{m2} & \dots & a_{n1}b_{1s} + a_{n2}b_{2s} + \dots + a_{nm}b_{ms} \end{bmatrix}$$

- $A$ 의  $i$ 번째 행과  $B$ 의  $j$ 번째 열이 서로 대응하여 연산
  - $A$ 의 열 크기와  $B$ 의 행 크기가 같아야 연산 가능
- 행렬  $A$ 의 크기가  $n \times m$  이고 행렬  $B$ 의 크기가  $m \times s$  일 때, 곱  $AB$ 의 결과로 나오는 행렬의 크기는  $n \times s$  임.
- $A \times A \times A = A^3, A \times A \times A \times \dots \times A = A^n$

# 행렬의 응용 - 컴퓨터 그래픽스

## 3차원 기하변환

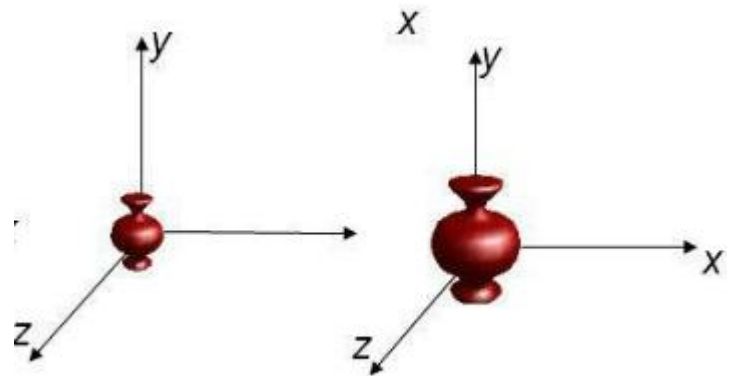
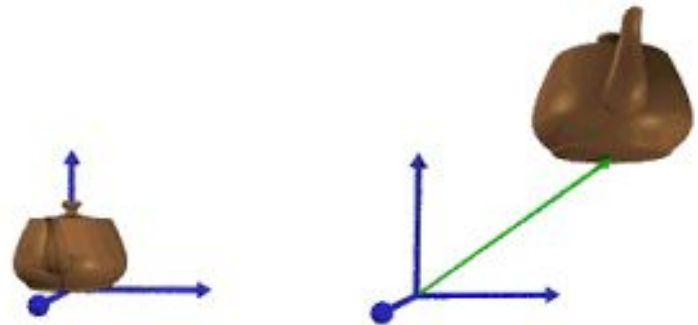
- 이동

$$\begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix}.$$

- 확대/축소

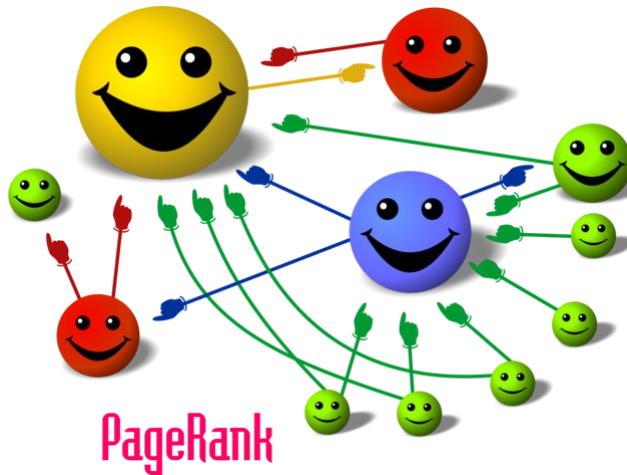
$$\begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix}.$$

- 회전, 반사, 밀림 등...



# 행렬의 응용 - 그래프 이론

$$\mathbf{r} = 0.85 \times \mathbf{P}^T \times \mathbf{r} + \frac{0.15}{N} \times \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$



국민대학교

전체 지도 이미지 뉴스 동영상 더보기 설정 도구

검색결과 약 29,100,000개 (0.52초)

국민대학교 - 고등교육의 새로운 표준을 제시하는 대학(Visual mode)

<https://www.kookmin.ac.kr/>

국민대학교는 1946년 해공 신익희 선생을 중심으로 상해 임시정부 요인들이 세운 민족혼이 깃든, 광복 후 최초의 민족사학으로 출발했습니다. 1959년 이래 학교를 ...

kookmin.ac.kr 검색결과

국민대학교 종합정보시스템

종합정보시스템의 사용자 아이디 (USER\_ID)는 학생은 학번, 교직 ...

국민대학교::조형대학

한국 디자인의 역사를 돌아보면, 제품 디자인, 그래픽디자인, 패션디 ...

학사일정

국민대학교는 1946년 해공 신익희 선생을 중심으로 상해 임시정부 ...

KMU eCampus

공지사항. [긴급-수정] eCampus 서비스 일시 중단 안내 N· 가상대학 ...

학사공지

기회! 도전 kookmin niversity. 홈 > 학사/행정 > 학사게시판 > 학사 ...

캠퍼스안내

이미지내의 건물번호를 클릭하시면 하단에 건물명과 설명을 보실 수 ...

국민대학교 - 나무위키

<https://namu.wiki/w/국민대학교>

3일 전 - 대한민국 임시정부의 정신을 계승하여, 독립 국가에 필요한 인재를 육성한다는 건국이념으로 국민대학교를 세웠다.

국민대학교 - 위키백과, 우리 모두의 백과사전

<https://ko.wikipedia.org/wiki/국민대학교>

국민대학교(國民大學校, Kookmin University)는 대한민국의 사립 종합 대학이다. 1946년 9월 1일 서울 창성동에 국민대학관으로 설립되었다. 독립 후 김구, 조소앙, ...

설립자: 신익희

학부생 수: 14,628명 (2018년)



교직원 수: 전임교원: 624명 (2018); 직원: 447명...















학교법인: 국민학원



# 행렬의 응용 - 추천시스템

Matrix  
Factorization

	M1	M2	M3	M4	M5
 Comedy	3	1	1	3	1
 Action	1	2	4	1	3

	 Comedy	 Action
 A		
 B		
 C		
 D		

	M1	M2	M3	M4	M5
	3	1	1	3	1
	1	2	4	1	3
	3	1	1	3	1
	4	3	5	4	4

NETFLIX

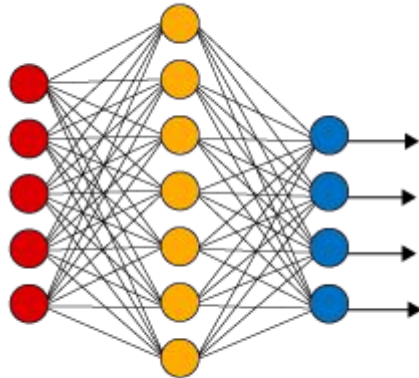
Melón

facebook

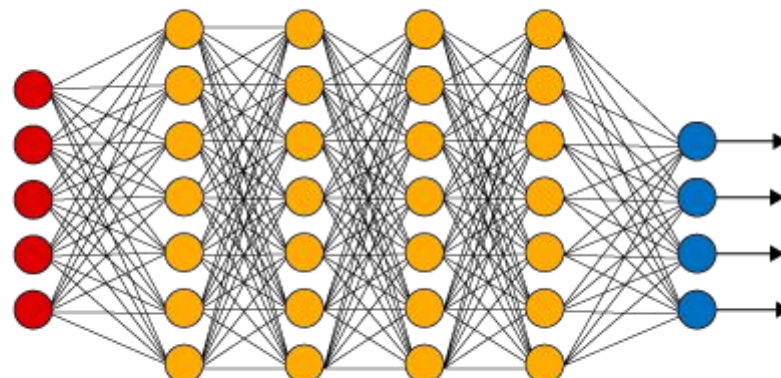
You Tube

# 행렬의 응용 - 딥러닝

Simple Neural Network



Deep Learning Neural Network

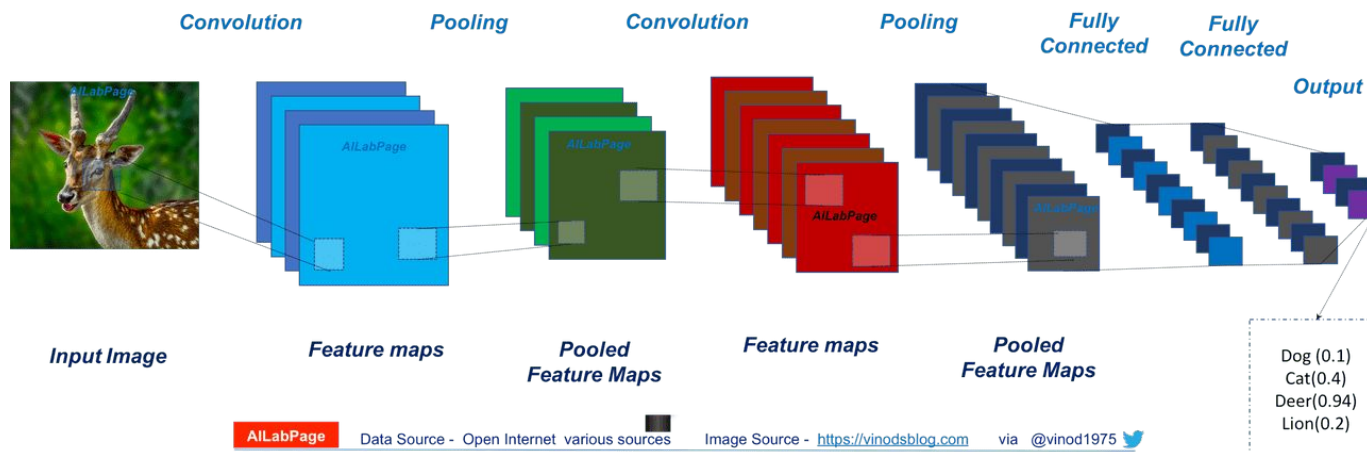


● Input Layer

● Hidden Layer

● Output Layer

## Convolution Neural Network



AILabPage

Data Source - Open Internet various sources

Image Source - <https://vinodsblog.com>

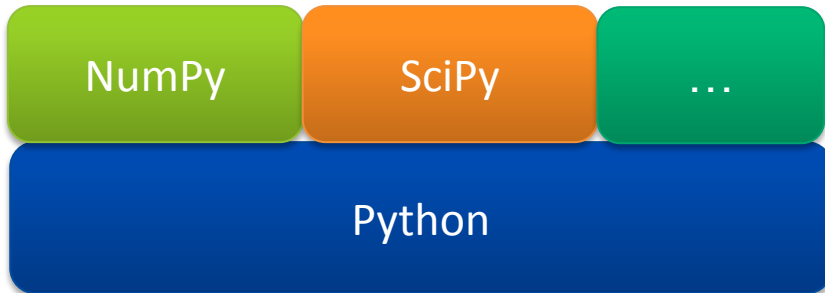
via @vinod1975

# 목차

- 벡터
- 행렬
- NumPy

# NumPy

- 데이터, 수치 분석을 위한 Python 패키지
- 효율적인 선형대수 프로그래밍 가능
- 사용하기 쉬움



It's so easy, a dog can do it!



# NumPy

- 효율적인 선형대수 연산 제공
  - 행렬(Matrix)과 벡터(Vector) 사용
  - 차수가 높은 행렬/벡터 연산도 손쉽게 가능

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} ax + by \\ cx + dy \end{bmatrix}$$

$$\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} ax + by + c \\ dx + ey + f \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} ax + by + cz \\ dx + ey + fz \\ gx + hy + iz \end{bmatrix}$$

$$\begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} ax + by + cz + d \\ ex + fy + gz + h \\ ix + jy + kz + l \\ 1 \end{bmatrix}$$



# 벡터 in NumPy

- `np.array()`를 사용하여 벡터 생성

```
import numpy as np

a = np.array([1,2,3])
b = np.array([2,3,4])

print(f"a+b = {a+b}") # 벡터 더하기
print(f"a-b = {a-b}") # 벡터 빼기
print(f"a*b = {a*b}") # 벡터 요소별 곱하기
print(f"a.dot(b) = {a.dot(b)}") # 벡터 내적
```

# 행렬 in NumPy

- `np.array()`를 사용하여 행렬 생성

```
import numpy as np

A = np.array([[1,2,3], [2,3,4]])
B = np.array([[2,3,4], [3,4,5]])

print(f"A+B = {A+B}") # 행렬 더하기
print(f"A-B = {A-B}") # 행렬 빼기
print(f"A.T = {A.T}") # 전치행렬 (Transpose)
print(f"A*B = {A*B}") # 행렬 요소별 곱하기
print(f"A.dot(B.T) = {A.dot(B.T)}") # 행렬 곱하기
```

# Questions?