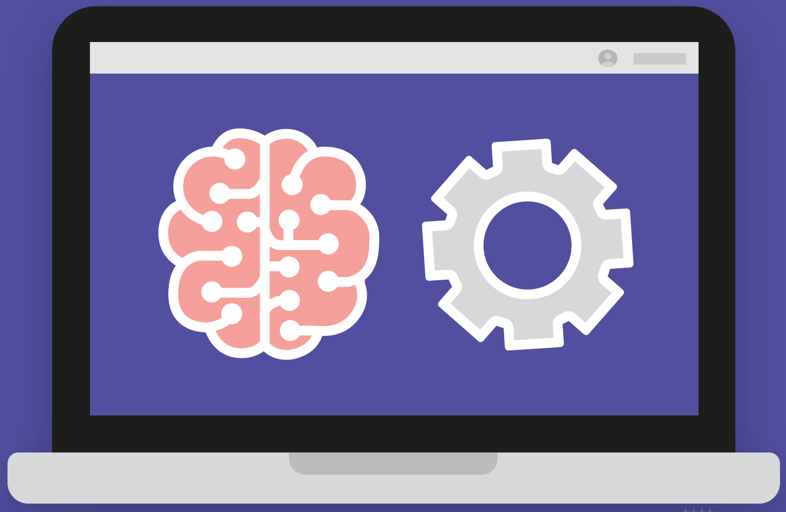
/* elice */

인공지능/머신러닝 기초

Module 2: 선형대수학/Numpy 기초

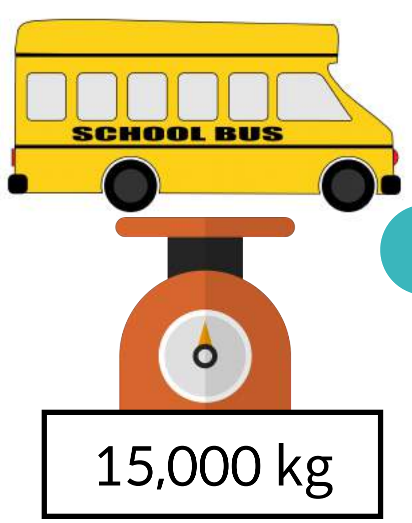


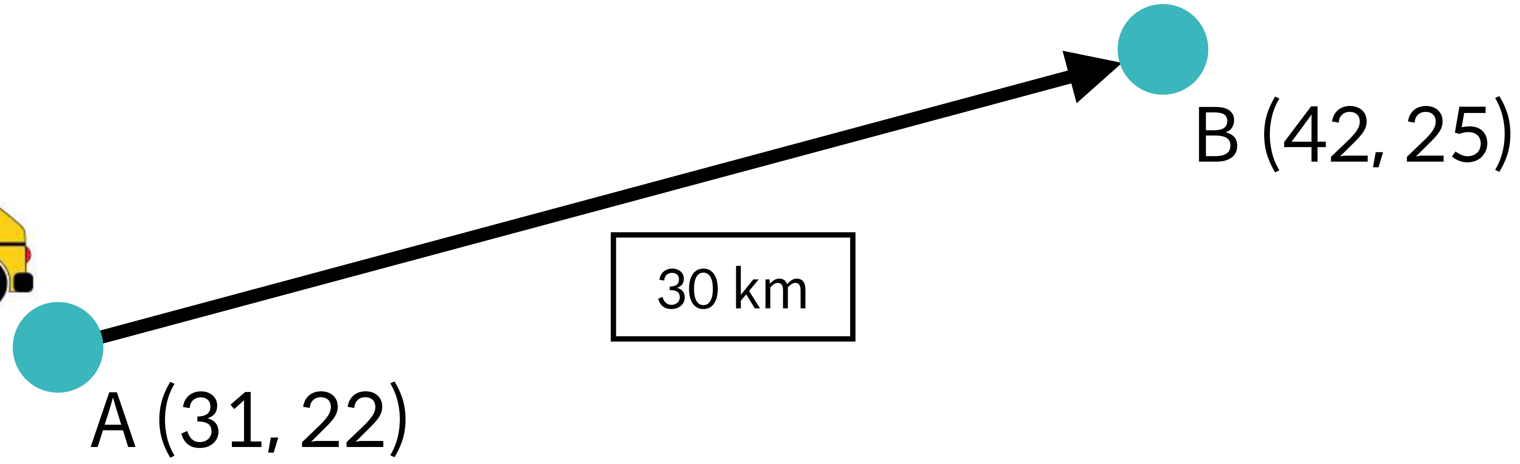
2월 7일 ~ 3월 7일

선형대수학 다시보기

Scalar와 Vector

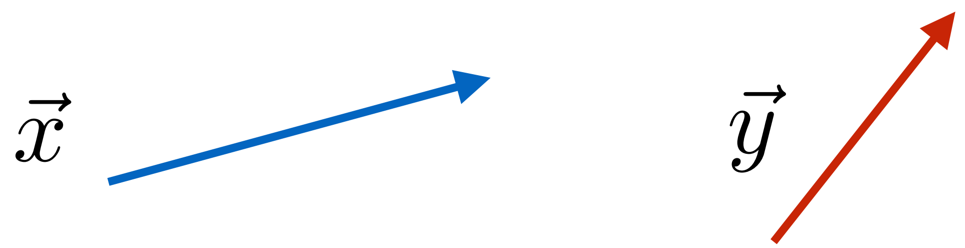
스칼라(scalar)	길이, 넓이, 질량, 온도 **크기만 존재하는 양**
벡터(vector)	속도, 위치 이동, 힘 **크기와 방향이 모두 존재하는 양**



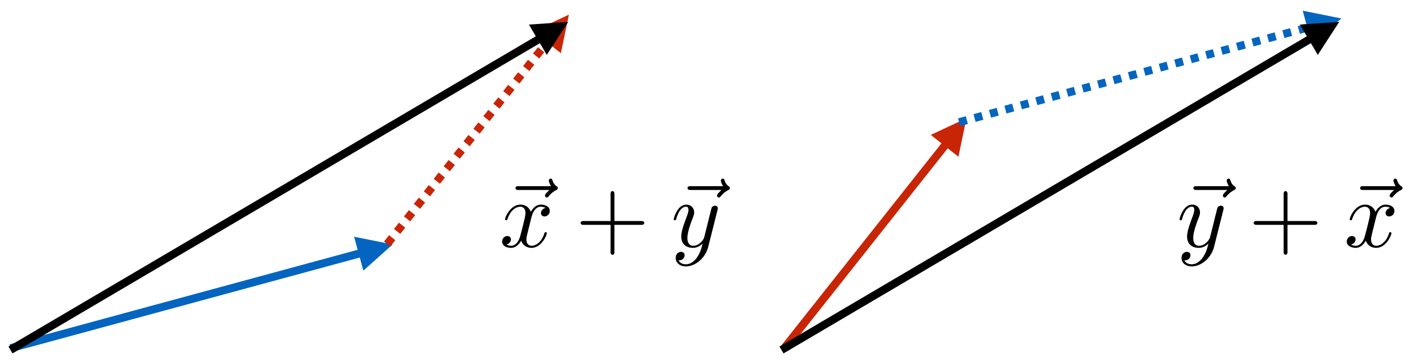


Vector Arithmetic

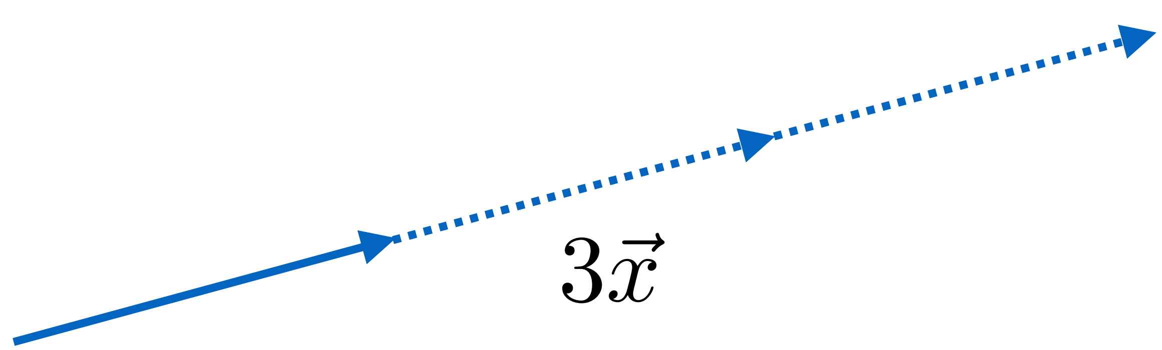
$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \vec{y} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$ 일 때



$$\vec{x} + \vec{y} = \begin{bmatrix} x_1 + y_1 \\ x_2 + y_2 \end{bmatrix}$$



$$k\vec{x} = \begin{bmatrix} kx_1 \\ kx_2 \end{bmatrix}$$



Quiz: N-Dim Vectors

$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \vec{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$ 일 때

$$\vec{x} + \vec{y} =$$

$$k\vec{x} =$$

Quiz: N-Dim Vectors

$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \vec{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$ 일 때

$$\vec{x} + \vec{y} = \begin{bmatrix} x_1 + y_1 \\ x_2 + y_2 \\ x_3 + y_3 \end{bmatrix}$$

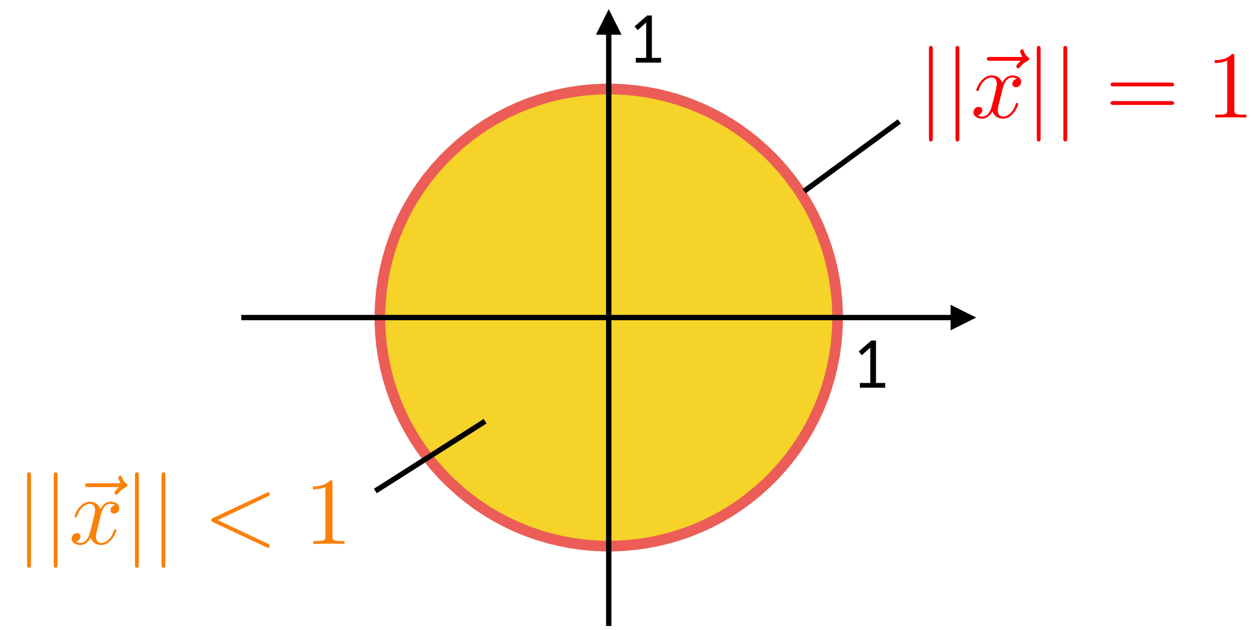
$$k\vec{x} = \begin{bmatrix} kx_1 \\ kx_2 \\ kx_3 \end{bmatrix}$$

Norm

n차원 벡터 $\vec{x} = (x_1, x_2, ..., x_n)$ 에 대해

Norm $||\vec{x}|| = \sqrt{x_1^2 + x_2^2 + ... + x_n^2}$

"원점 O에서 점 $(x_1, x_2, ..., x_n)$ 까지 이르는 거리"



내적

Euclidean inner product, Dot product

$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \vec{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$ 일 때 $\vec{x} \cdot \vec{y} = x_1 y_1 + x_2 y_2 + x_3 y_3$

Quiz: 내적

Euclidean inner product, Dot product

$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \vec{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$ 일 때 $\vec{x} \cdot \vec{y} = x_1 y_1 + x_2 y_2 + x_3 y_3$

$\vec{x} = (x_1, x_2, ..., x_n), \vec{y} = (y_1, y_2, ..., y_n)$ 일 때

$\vec{x} \cdot \vec{y} =$

Quiz: 내적

Euclidean inner product, Dot product

$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \vec{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$ 일 때 $$\vec{x} \cdot \vec{y} = x_1 y_1 + x_2 y_2 + x_3 y_3$$

$$\vec{x} = (x_1, x_2, \dots, x_n), \vec{y} = (y_1, y_2, \dots, y_n)$$ 일 때

$$\vec{x} \cdot \vec{y} = (x_1 y_1 + x_2 y_2 + \dots + x_n y_n)$$

Matrix

실수를 다음과 같이 직사각형 모양으로
배열한 것을 **행렬**이라고 한다.

$$A = \begin{bmatrix} x_{11} & x_{12} & x_{13} & \cdots & x_{1n} \\ x_{21} & x_{22} & x_{23} & \cdots & x_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{d1} & x_{d2} & x_{d3} & \cdots & x_{dn} \end{bmatrix}$$

행 (row)

열 (column)

Matrix Arithmetic

같은 **차원**을 가진 행렬끼리만 더하거나 뺄 수 있다.

$$A = \begin{bmatrix} 1 & 2 & -1 \\ 4 & 3 & 0 \end{bmatrix}$$ 이고 $$B = \begin{bmatrix} 3 & 1 & 2 \\ 0 & -2 & 5 \end{bmatrix}$$ 일 때,

$A,\ B$ 모두 $(2,\ 3)$ 의 차원을 가진다.

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

Matrix Arithmetic

행렬끼리 곱할 때는 차원을 주의해야 한다.

$$A = \begin{bmatrix} a & b \\ c & d \\ e & f \end{bmatrix}, B = \begin{bmatrix} g & h & i \\ j & k & l \end{bmatrix}$$ 일 때,

A는 $(3,\ 2)$, B는 $(2,\ 3)$ 의 차원을 가진다.

이 두 차원이 같아야 행렬곱이 가능하다.

행렬곱의 결과는 이 차원들에 의해 결정된다.

Matrix Arithmetic

행렬끼리 곱할 때는 차원을 주의해야 한다.

$$A = \begin{bmatrix} a & b \\ c & d \\ e & f \end{bmatrix}, B = \begin{bmatrix} g & h & i \\ j & k & l \end{bmatrix}$$ 일 때,

A는 $(3,\ 2)$, B는 $(2,\ 3)$ 의 차원을 가진다.

$$AB = \begin{bmatrix} ag+bj & ah+bk & ai+bl \\ cg+dj & ch+dk & ci+dl \\ eg+fj & eh+fk & ei+fl \end{bmatrix}$$

Quiz: Matrix Arithmetic

행렬끼리 곱할 때는 차원을 주의해야 한다.

$A = \begin{bmatrix} 1 & 2 & -1 \\ 4 & 3 & 0 \end{bmatrix}$ 이고 $B = \begin{bmatrix} 4 & 2 \\ 6 & -3 \\ 5 & 1 \end{bmatrix}$ 일 때,

$$\mathrm{AB} =$$

Quiz: Matrix Arithmetic

행렬끼리 곱할 때는 차원을 주의해야 한다.

$A = \begin{bmatrix} 1 & 2 & -1 \\ 4 & 3 & 0 \end{bmatrix}$ 이고 $B = \begin{bmatrix} 4 & 2 \\ 6 & -3 \\ 5 & 1 \end{bmatrix}$ 일 때,

$$AB = \begin{bmatrix} 11 & -6 \\ 34 & -1 \end{bmatrix}$$

Transpose

전치행렬은 원행렬의 행과 열을 뒤바꾼 행렬이다.

$$A = \begin{bmatrix} 1 & 2 & -1 \\ 4 & 3 & 0 \end{bmatrix}$$ 일 때,

$$A^T = \begin{bmatrix} 1 & 4 \\ 2 & 3 \\ -1 & 0 \end{bmatrix}$$

Numpy 소개

Numpy?

Python에서 사용되는 **과학 컴퓨팅**용 라이브러리

Python 언어에서 기본으로 지원하지 않는 행렬과 같은 **데이터 구조** 지원 및 **수학/과학 계산 함수** 포함

NumPy

$$A = \begin{pmatrix} 1 & 4 & 5 & 8 \\ 2 & 1 & 7 & 3 \\ 5 & 4 & 5 & 9 \end{pmatrix}$$

행렬이 왜 필요한가?

머신러닝에서 대부분의 데이터는 행렬로 표현됨

X

X1	X2	X3	X4
2.4	0.05	14.3	1.42
3.2	0.48	5.2	3.22
2.1	0.92	8.7	4.76
0.8	0.11	5.1	2.77
1.7	0.38	9.4	4.82

->

Y

Y
22.1
9.4
10.3
8.2
17.2

행렬 만들기

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

이 행렬을 어떻게 표현할까?

```python
import numpy as np

A = np.array([[1, 2],
              [3, 4]])
print(A)
```

→

```
[[1 2]
 [3 4]]
```

Numpy Array

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

이렇게 만들어진 행렬은

곱셈, 덧셈, 뺄셈이 가능

```
print(A * 3)

print(A + A)

print(A - A)
```

```
[[ 3  6]
 [ 9 12]]
[[2 4]
 [6 8]]
[[0 0]
 [0 0]]
```

Numpy Array - 산술 연산

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

행렬 내 원소에 대한 산술연산도 가능

"element-wise operation"

```
print(A ** 2)

print(3 ** A)

print(A * A)
```

```
[[ 1  4]
 [ 9 16]]
[[ 3  9]
 [27 81]]
[[ 1  4]
 [ 9 16]]
```

행렬 곱셈

```
x = np.array([[1, 2], [3, 4]])
y = np.array([[3, 4], [3, 2]])
```

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}\begin{pmatrix} 3 & 4 \\ 3 & 2 \end{pmatrix} = \begin{pmatrix} 9 & 8 \\ 21 & 20 \end{pmatrix}$$

`np.dot(x, y)` 는 `x * y` 와 다릅니다. 어떻게?

```
print(np.dot(x, y))

print(x * y)
```

→

```
[[ 9  8]
 [21 20]]
[[3 8]
 [9 8]]
```

Numpy Array - 비교 연산

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

비교연산을 통해 array 내의 값을 **빠르게** 비교 가능

```
print(a == b)

print(a > b)
```

```
[False,  True, False,  True]

[False, False,  True, False]
```

Numpy Array - 논리 연산

```
a = np.array([1, 1, 0, 0], dtype=bool)
b = np.array([1, 0, 1, 0], dtype=bool)
```

`logical_and`, `logical_or` 함수를 이용하면

array 내의 element-wise 연산 수행 가능

```
np.logical_or(a, b)

np.logical_and(a, b)
```

```
[ True,  True,  True, False]

[ True, False, False, False]
```

Numpy Array - Reductions

```
a = np.array([1, 2, 3, 4, 5])
```

argmin/max: 최소/최대값의 **인덱스**를 반환

```
np.sum(a)
a.sum()

a.min()
a.max()

a.argmin()
a.argmax()
```

```
15
15

1
5

0
4
```

Logical Reductions

```
a = np.array([True, True, True])
b = np.array([True, True, False])
```

`all`: Array 내의 **모든 값**이 `True`인가?

`any`: Array 내의 값이 **하나라도** `True`인가?

```
np.all(a)
np.all(b)

np.any(a)
np.any(b)
```

→

```
True
False

True
True
```

Statistical Reductions

```
x = np.array([1, 2, 3, 1])
```

`np.mean(x)` : 평균값

`np.median(x)` : 중간값

`np.std(x)` : 표준편차

```
print(np.mean(x))
print(np.median(x))
print(np.std(x))
```

```
1.75
1.5
0.82915619758884995
```

실습: Numpy 따라하기