

Mathematics for Artificial Intelligence

1강: 벡터가 뭔가요?



임성빈

UNIST

인공지능대학원 & 산업공학과
Learning Intelligent Machine Lab

X
boostcamp AI Tech

© NAVER Connect Foundation

벡터가 뭔가요?

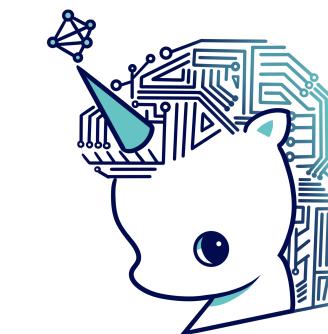
- 벡터는 숫자를 원소로 가지는 **리스트(list)** 또는 **배열(array)**입니다

수식

$$\mathbf{x} = \begin{bmatrix} 1 \\ 7 \\ 2 \end{bmatrix} \quad \mathbf{x}^\top = [1, 7, 2]$$

코드

```
x = [1, 7, 2]  
x = np.array([1, 7, 2])
```



np 는 numpy 를 말합니다

벡터가 뭔가요?

- 벡터는 숫자를 원소로 가지는 **리스트(list)** 또는 **배열(array)**입니다

$$\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}$$

열벡터

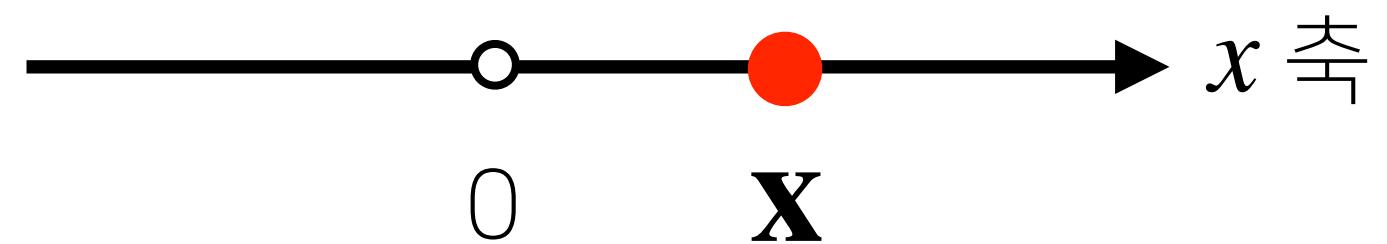
$$\mathbf{X}^T = [x_1, x_2, \dots, x_d]$$

행벡터

벡터의 차원

벡터가 뭔가요?

- 벡터는 공간에서 한 점을 나타냅니다

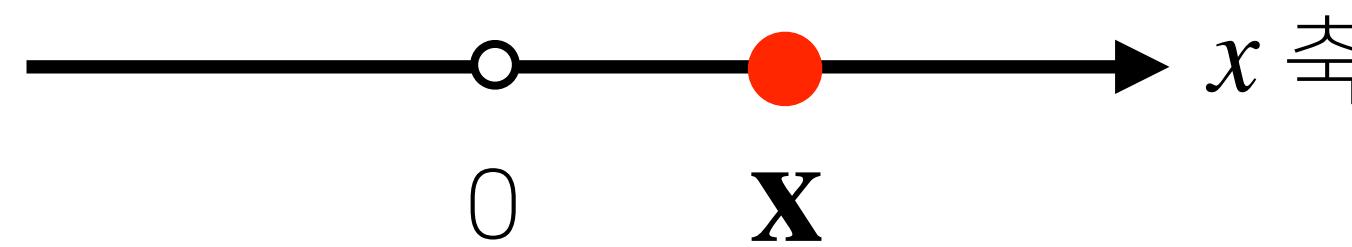


1차원 공간 (수직선)

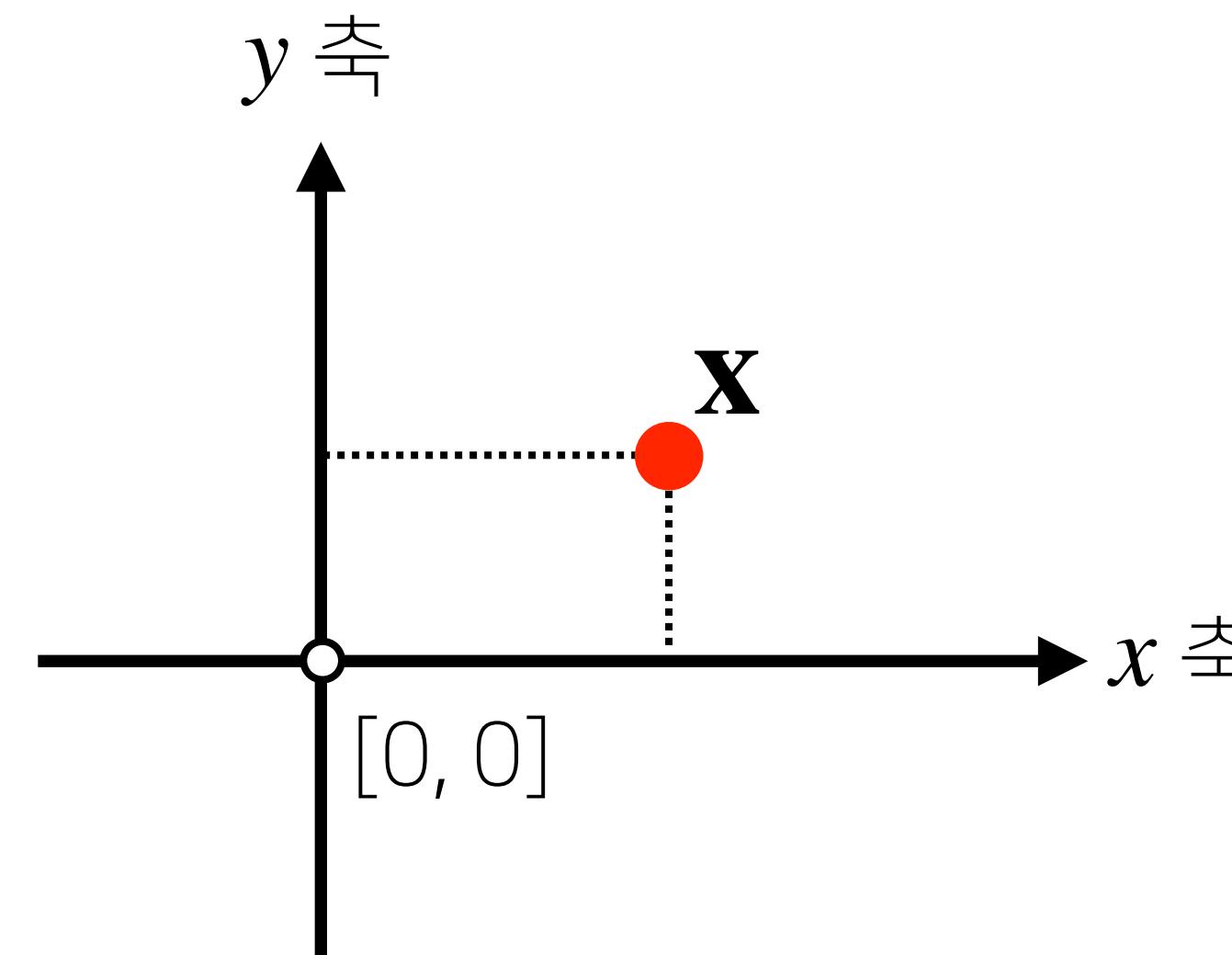
X

벡터가 뭔가요?

- 벡터는 공간에서 **한 점**을 나타냅니다



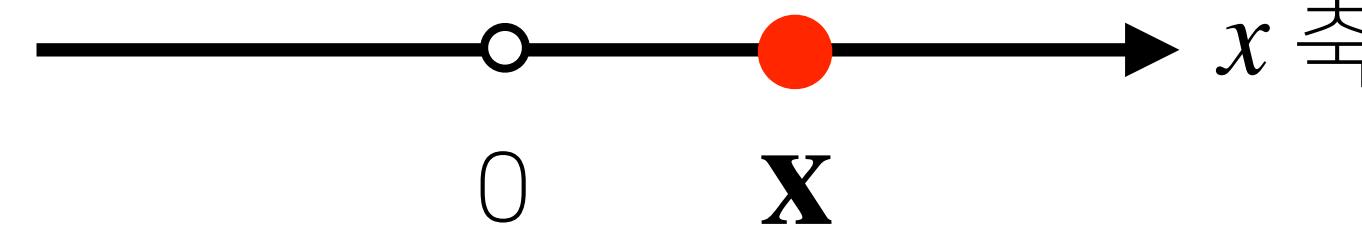
1차원 공간 (수직선)



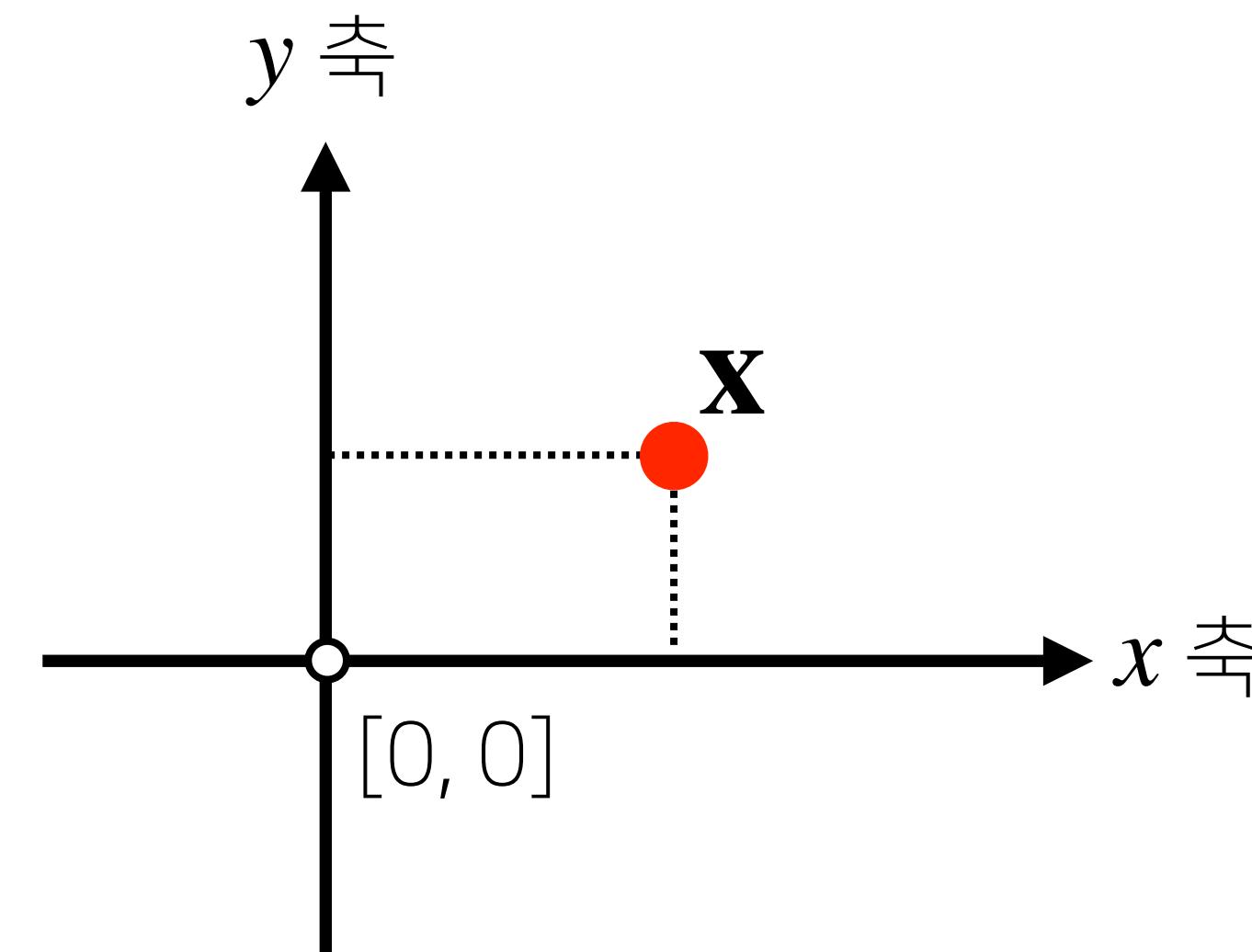
2차원 공간 (좌표평면)

벡터가 뭔가요?

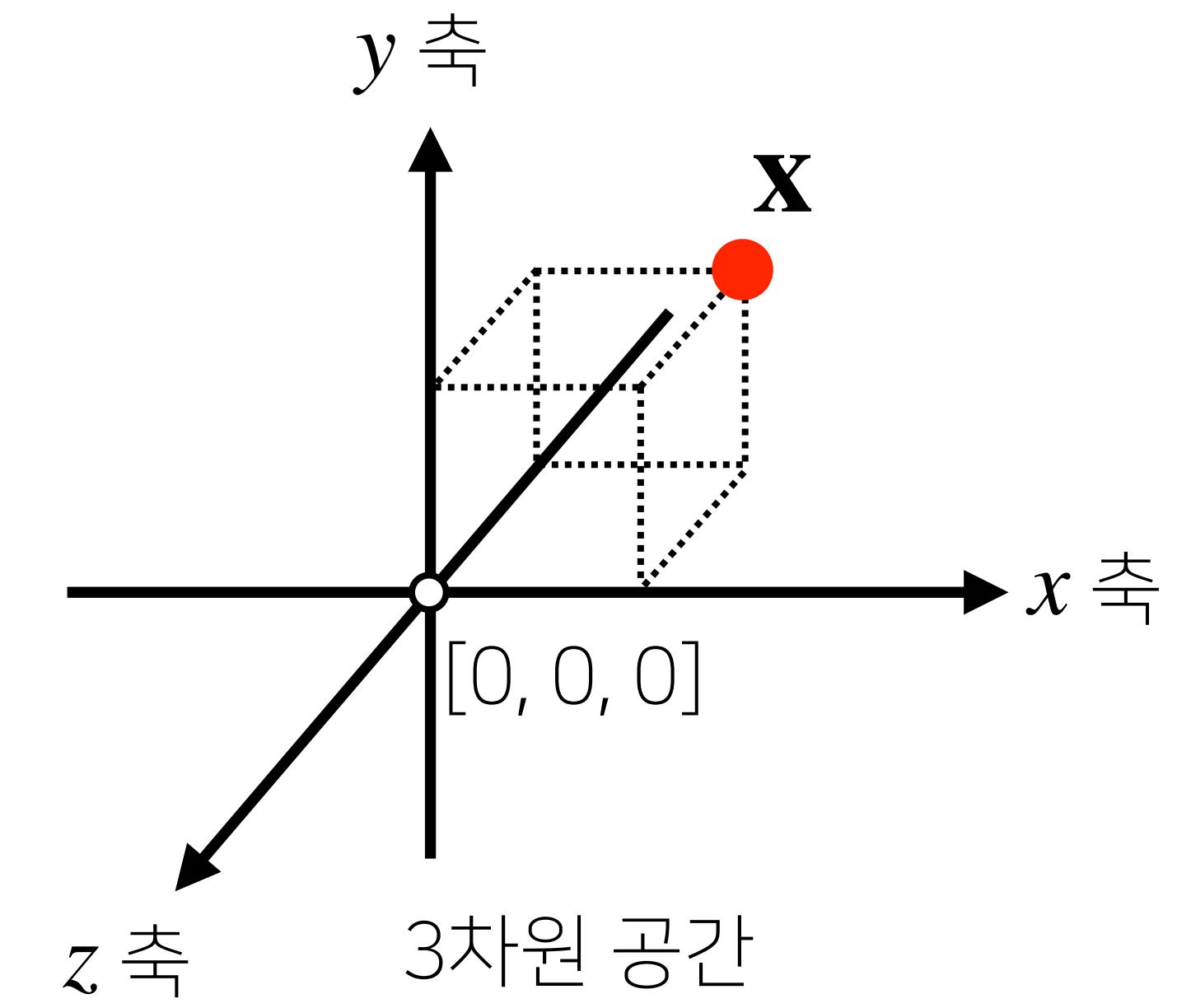
- 벡터는 공간에서 **한 점**을 나타냅니다



1차원 공간 (수직선)



2차원 공간 (좌표평면)

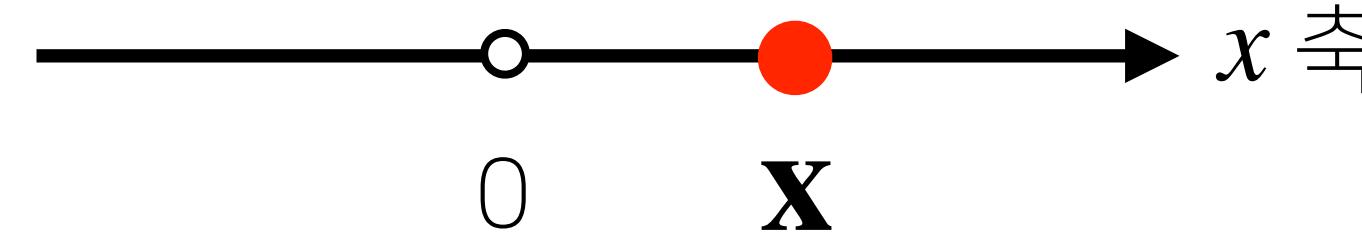


3차원 공간

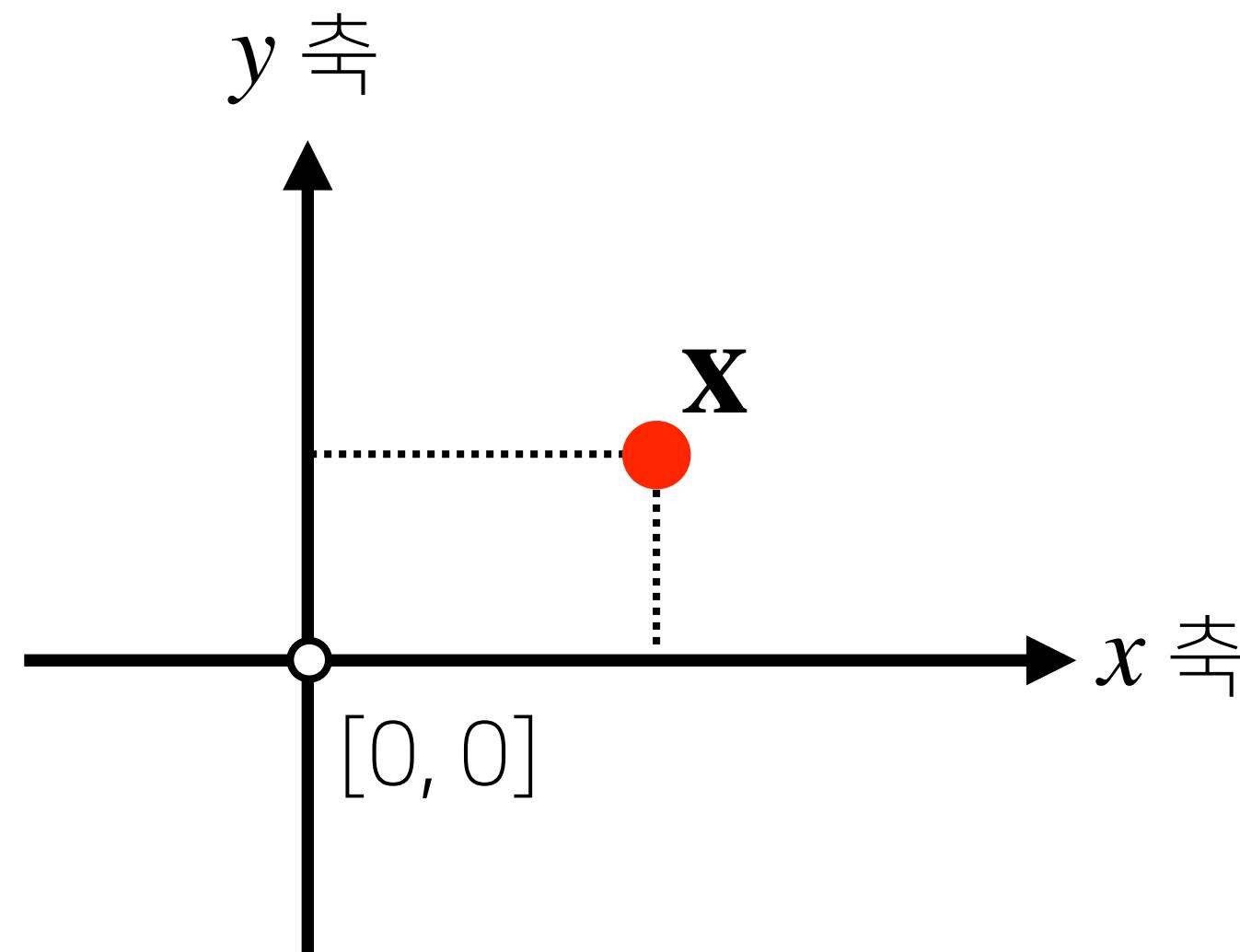
벡터가 뭔가요?

- 벡터는 공간에서 **한 점**을 나타냅니다

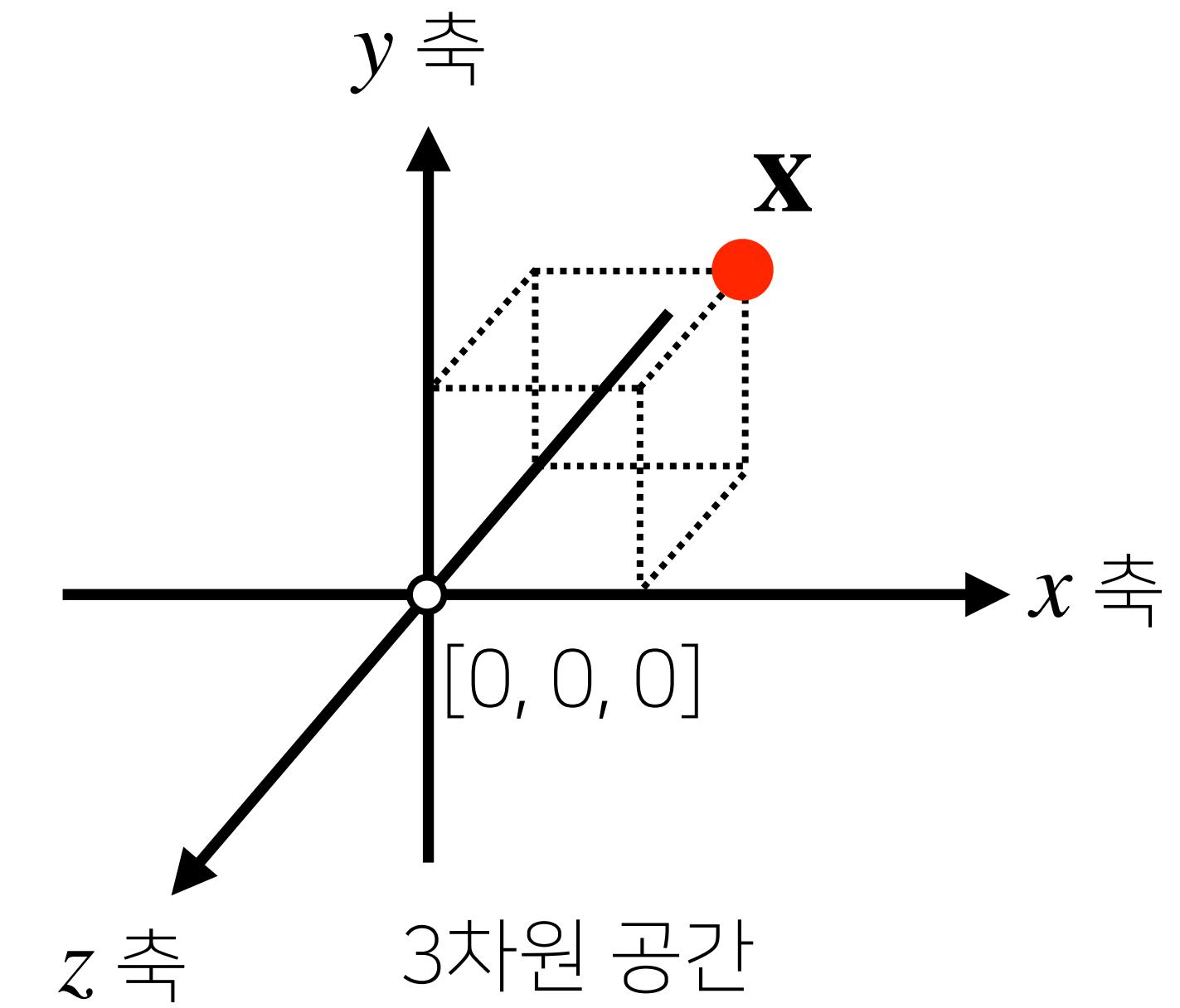
x
n차원 공간



1차원 공간 (수직선)



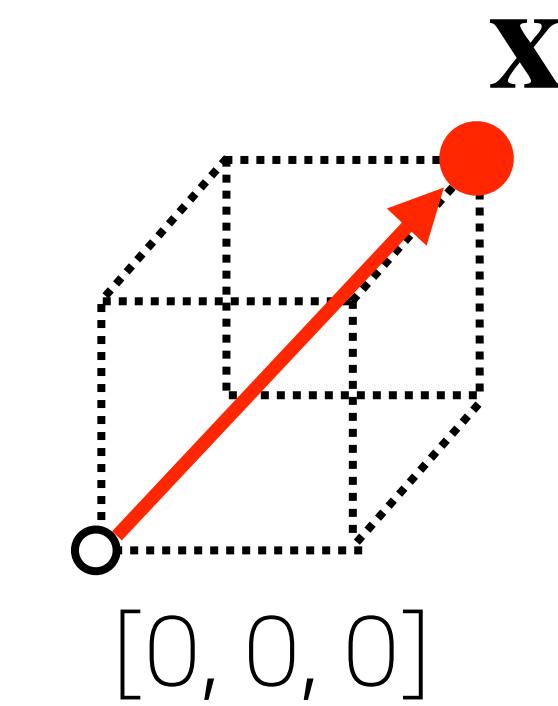
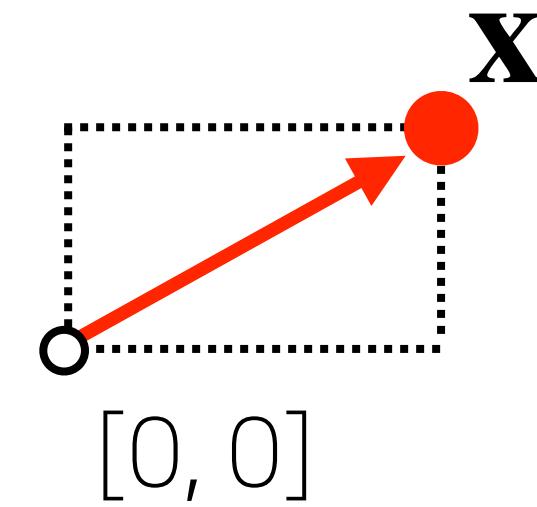
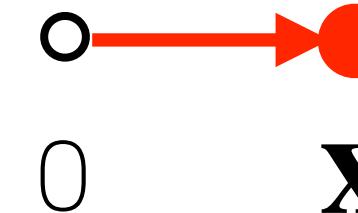
2차원 공간 (좌표평면)



3차원 공간

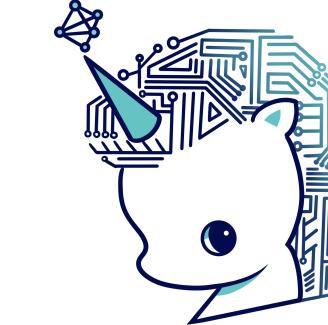
벡터가 뭔가요?

- 벡터는 공간에서 **한 점**을 나타냅니다
- 벡터는 원점으로부터 상대적 **위치**를 표현합니다



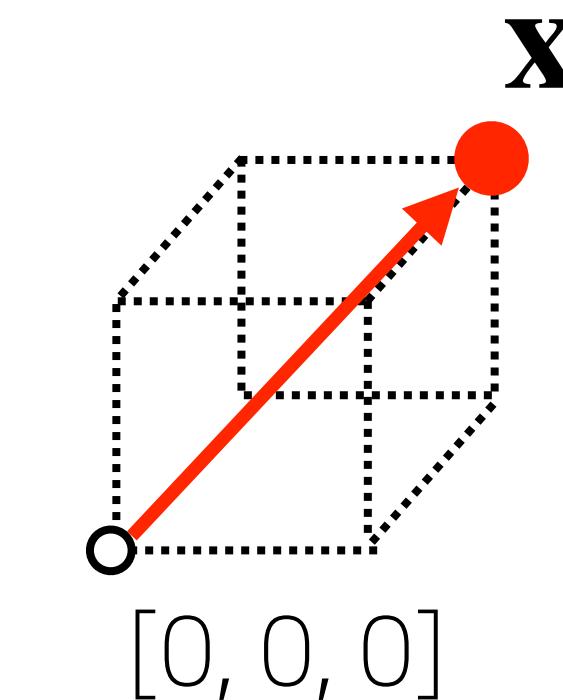
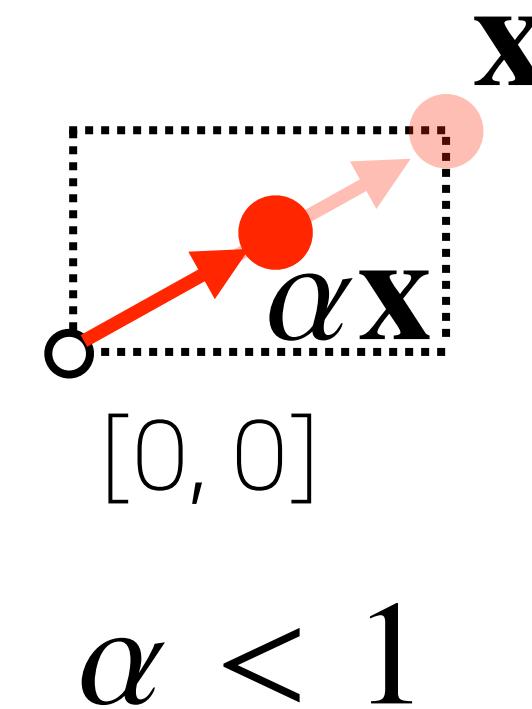
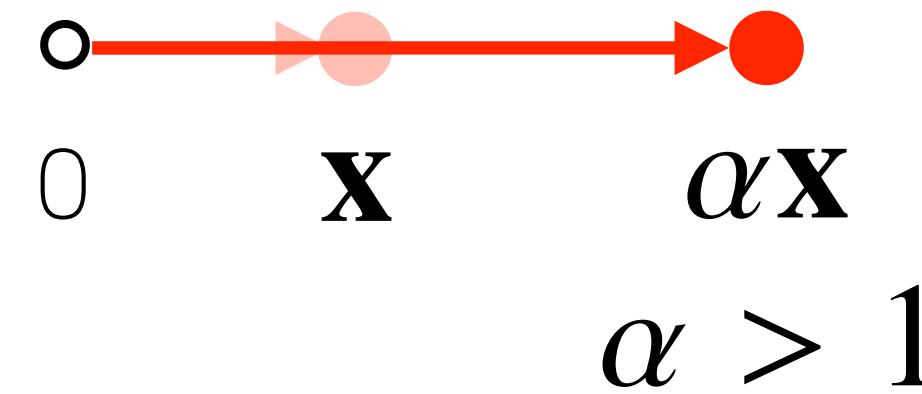
벡터가 뭔가요?

- 벡터는 공간에서 **한 점**을 나타냅니다
- 벡터는 원점으로부터 상대적 **위치**를 표현합니다
- 벡터에 숫자를 곱해주면 **길이만 변합니다**



이걸 **스칼라곱**이라 부릅니다

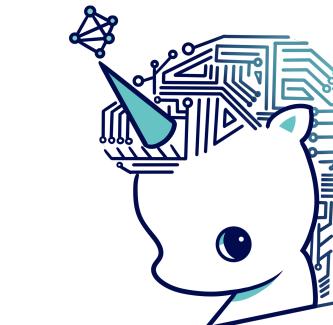
$$\alpha \mathbf{x} = \begin{bmatrix} \alpha x_1 \\ \alpha x_2 \\ \vdots \\ \alpha x_d \end{bmatrix}$$



1보다 크면 길이가 늘어나고, 1보다 작으면 길이가 줄어듭니다

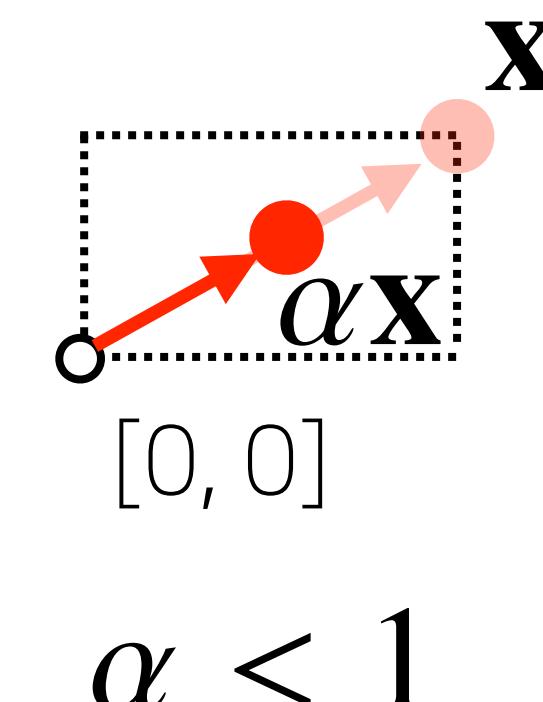
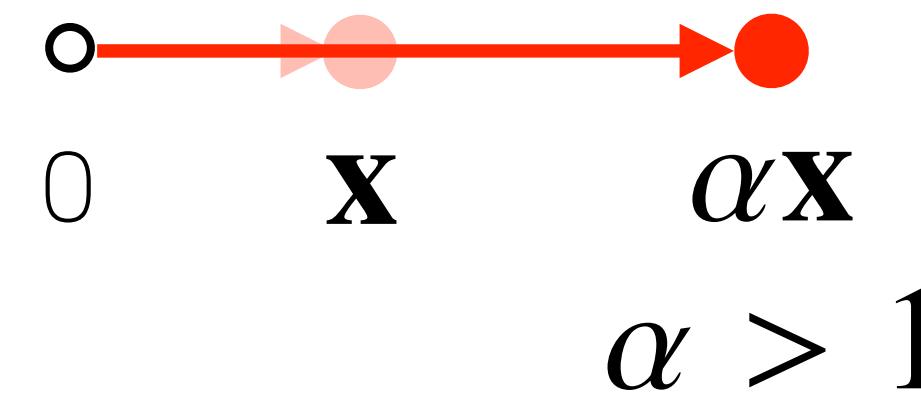
벡터가 뭔가요?

- 벡터는 공간에서 **한 점**을 나타냅니다
- 벡터는 원점으로부터 상대적 **위치**를 표현합니다
- 벡터에 숫자를 곱해주면 **길이만 변합니다**

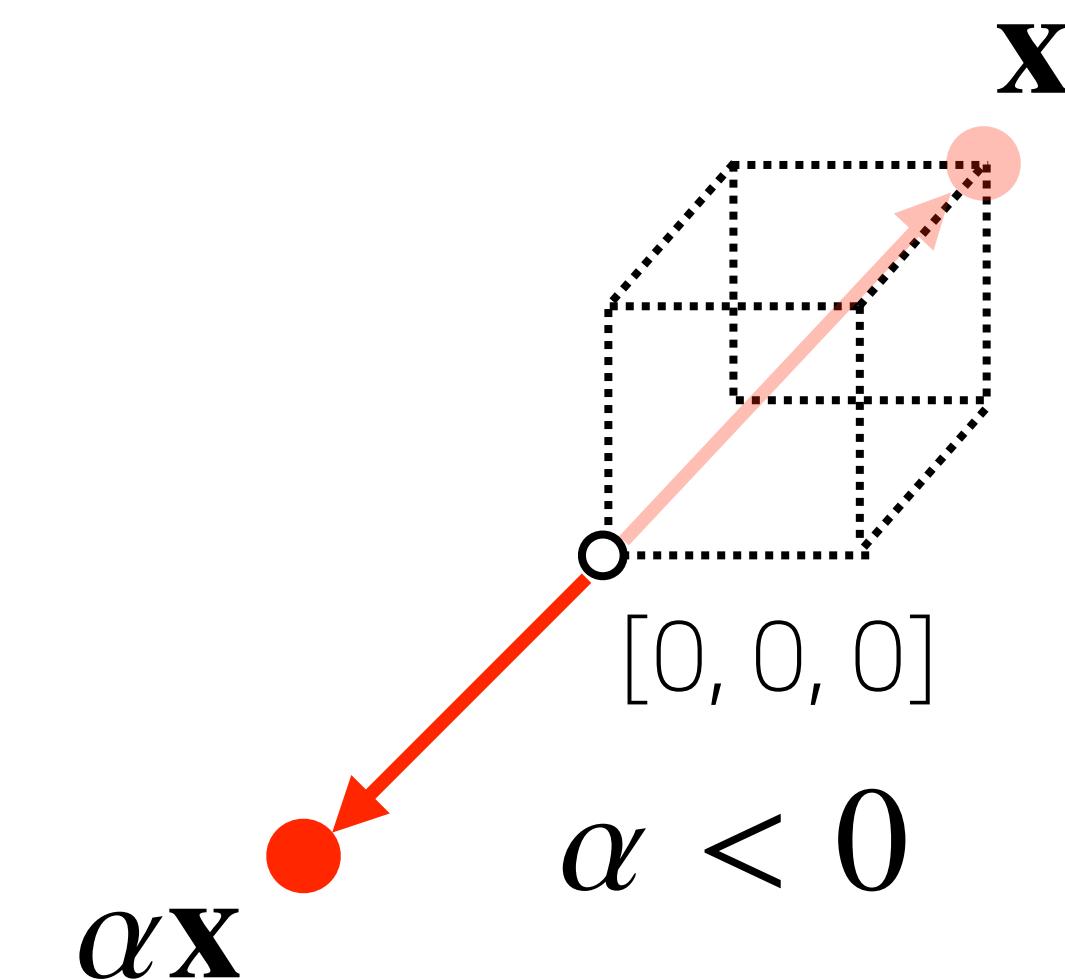


이걸 **스칼라곱**이라 부릅니다

$$\alpha \mathbf{x} = \begin{bmatrix} \alpha x_1 \\ \alpha x_2 \\ \vdots \\ \alpha x_d \end{bmatrix}$$



1보다 크면 길이가 늘어나고, 1보다 작으면 길이가 줄어듭니다



단, 0보다 작으면 **반대 방향**이 됩니다

벡터가 뭔가요?

- 벡터는 숫자를 원소로 가지는 **리스트(list)** 또는 **배열(array)**입니다
- 벡터끼리 같은 모양을 가지면 덧셈, 뺄셈을 계산할 수 있습니다

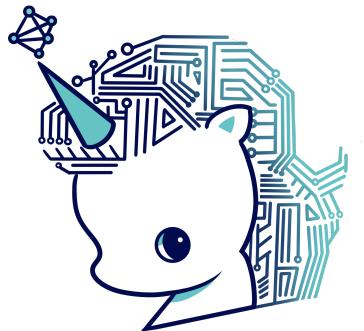
$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_d \end{bmatrix} \quad \mathbf{x} \pm \mathbf{y} = \begin{bmatrix} x_1 \pm y_1 \\ x_2 \pm y_2 \\ \vdots \\ x_d \pm y_d \end{bmatrix}$$

벡터가 뭔가요?

- 벡터는 숫자를 원소로 가지는 **리스트(list)** 또는 **배열(array)**입니다
- 벡터끼리 같은 모양을 가지면 성분곱(Hadamard product)을 계산할 수 있다

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_d \end{bmatrix} \quad \mathbf{x} \odot \mathbf{y} = \begin{bmatrix} x_1 y_1 \\ x_2 y_2 \\ \vdots \\ x_d y_d \end{bmatrix}$$

벡터가 뭔가요?



numpy 에서도 똑같이 계산됩니다

- 벡터는 숫자를 원소로 가
- 벡터끼리 같은 모양을 가

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}$$

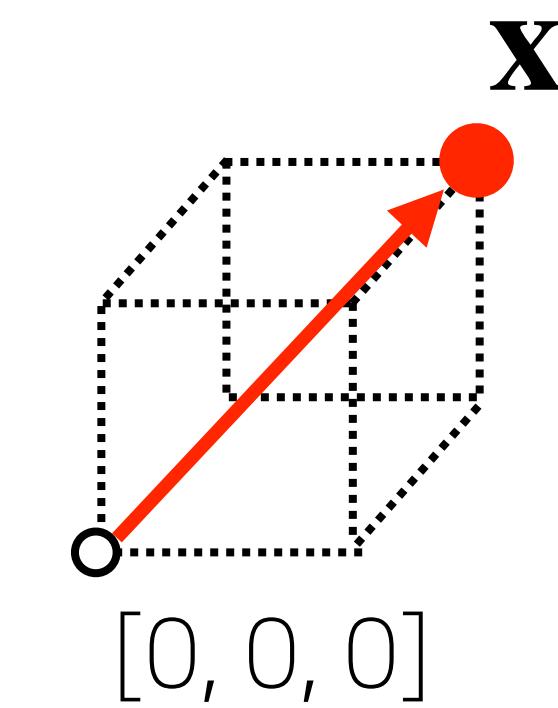
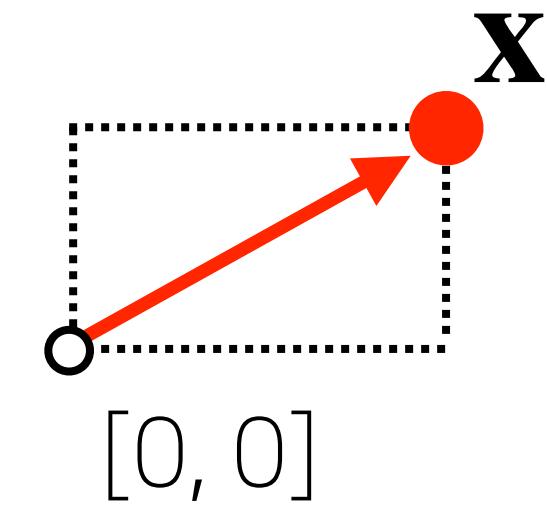
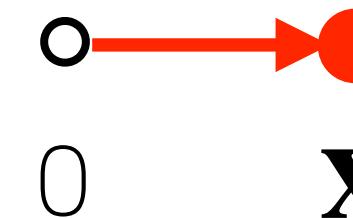
\times

```
[1] 1 import numpy as np
[2] 1 x = np.array([1, 7, 2])
    2 y = np.array([5, 2, 1])
[3] 1 x + y
    array([6, 9, 3])
[4] 1 x - y
    array([-4, 5, 1])
[5] 1 x * y
    array([ 5, 14,  2])
```

$$\begin{bmatrix} x_1 y_1 \\ x_2 y_2 \\ \vdots \\ x_d y_d \end{bmatrix}$$

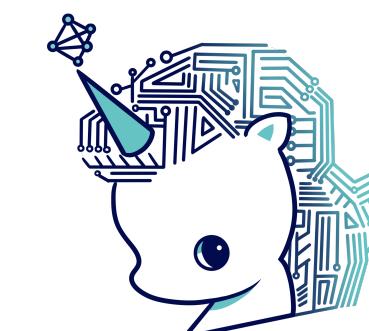
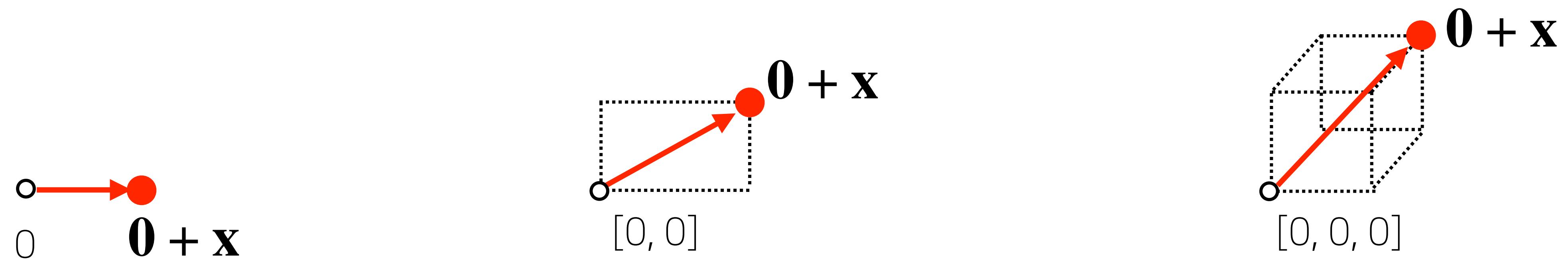
벡터의 덧셈을 알아보자

- 벡터는 공간에서 **한 점**을 나타냅니다
- 벡터는 원점으로부터 상대적 **위치**를 표현합니다



벡터의 덧셈을 알아보자

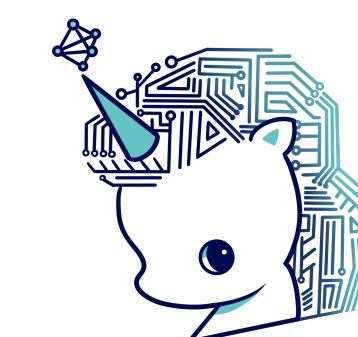
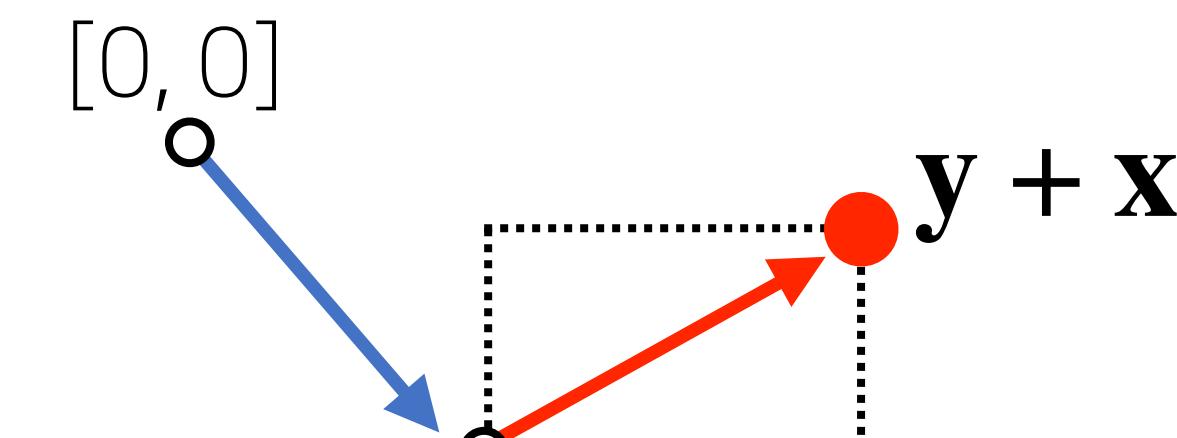
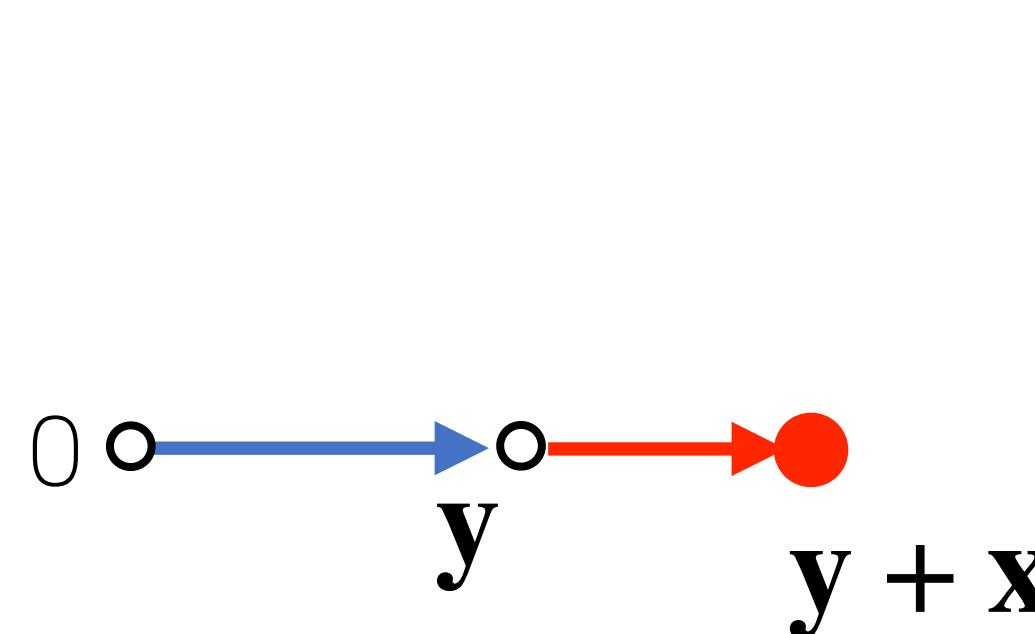
- 벡터는 공간에서 **한 점**을 나타냅니다
- 벡터는 원점으로부터 상대적 **위치**를 표현합니다
- 두 벡터의 덧셈은 다른 벡터로부터 **상대적 위치이동**을 표현합니다



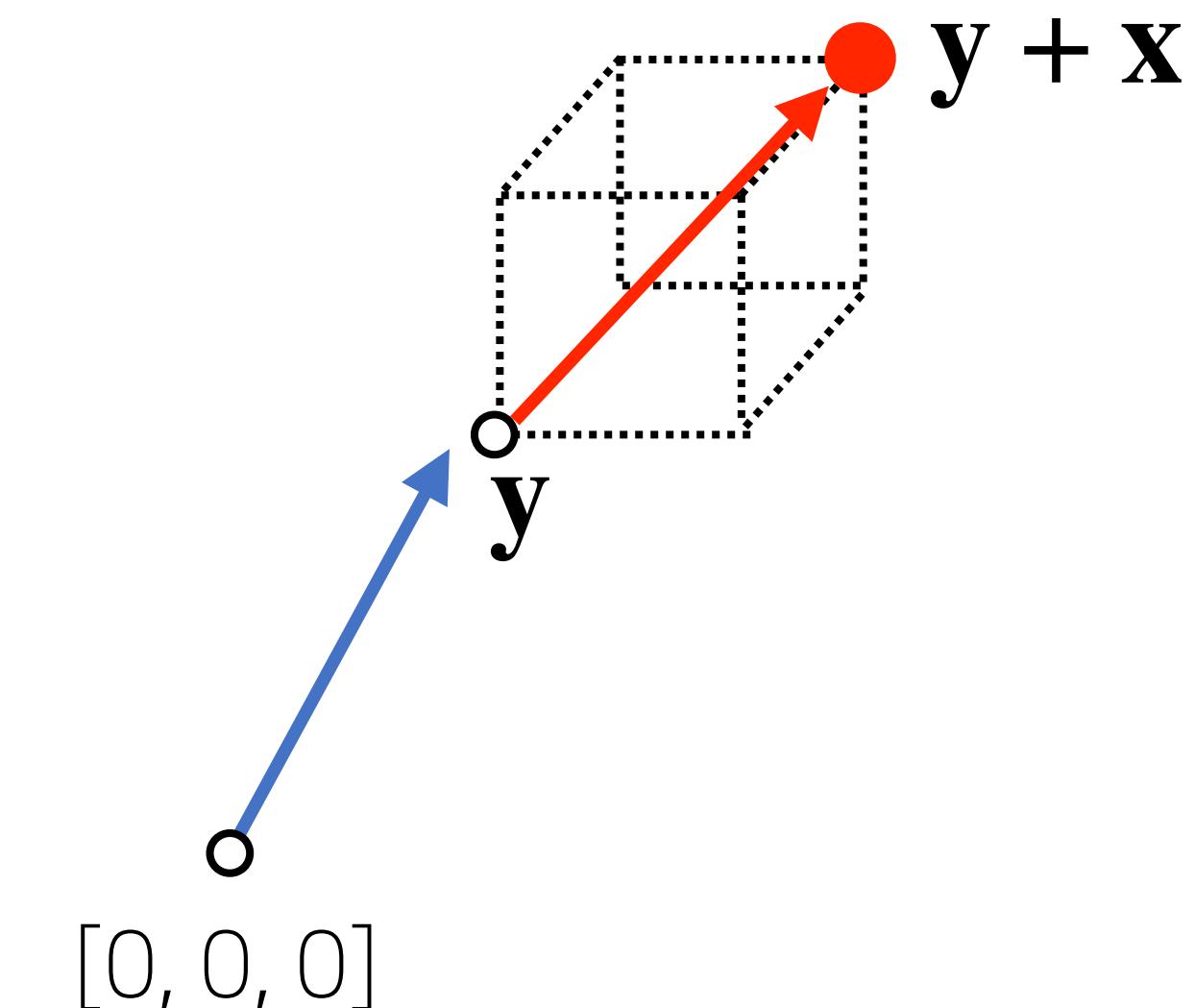
$\mathbf{x} = \mathbf{0} + \mathbf{x}$ 가 성립합니다

벡터의 덧셈을 알아보자

- 벡터는 공간에서 **한 점**을 나타냅니다
- 벡터는 원점으로부터 상대적 **위치**를 표현합니다
- 두 벡터의 덧셈은 다른 벡터로부터 **상대적 위치이동**을 표현합니다

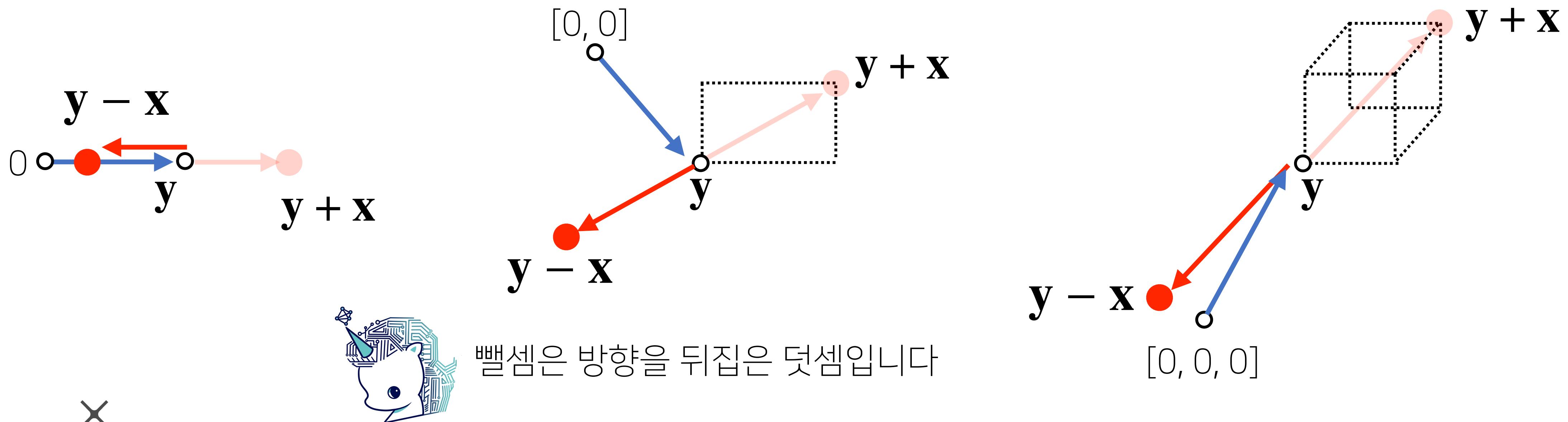


원점을 **y**로 옮기는 것이
벡터의 덧셈입니다



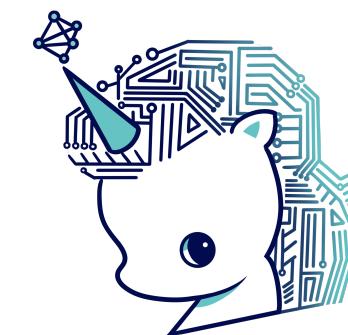
벡터의 뺄셈은?

- 벡터는 공간에서 **한 점**을 나타냅니다
- 벡터는 원점으로부터 상대적 **위치**를 표현합니다
- 두 벡터의 뺄셈은 다른 벡터로부터 **상대적 위치이동**을 표현합니다



벡터의 노름 구해보기

- 벡터의 노름(norm)은 원점에서부터의 거리를 말합니다

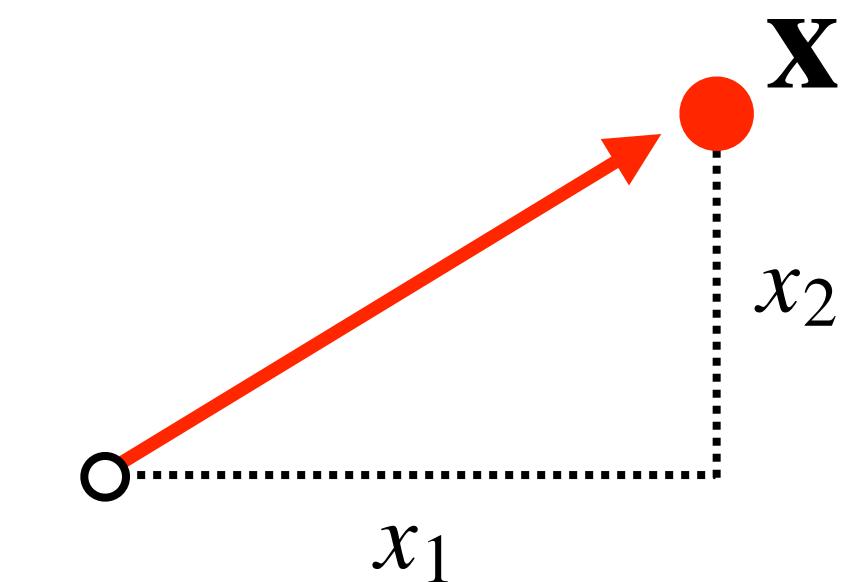


$\|\cdot\|$ 기호는 노름(norm)이라 부릅니다

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}$$

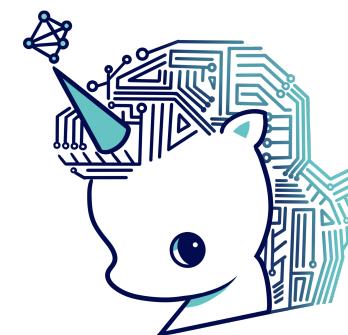
$$\|\mathbf{x}\|_1 = \sum_{i=1}^d |x_i|$$

$$\|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^d |x_i|^2}$$



벡터의 노름 구해보기

- 벡터의 노름(norm)은 원점에서부터의 거리를 말합니다

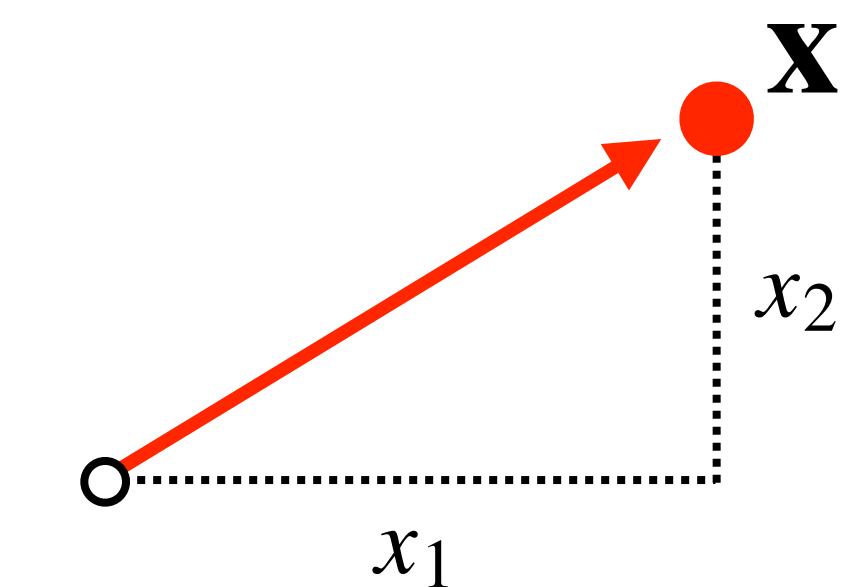


임의의 차원 d 에 대해 성립하는 것을 명심하세요

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}$$

$$\|\mathbf{x}\|_1 = \sum_{i=1}^d |x_i|$$

$$\|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^d |x_i|^2}$$



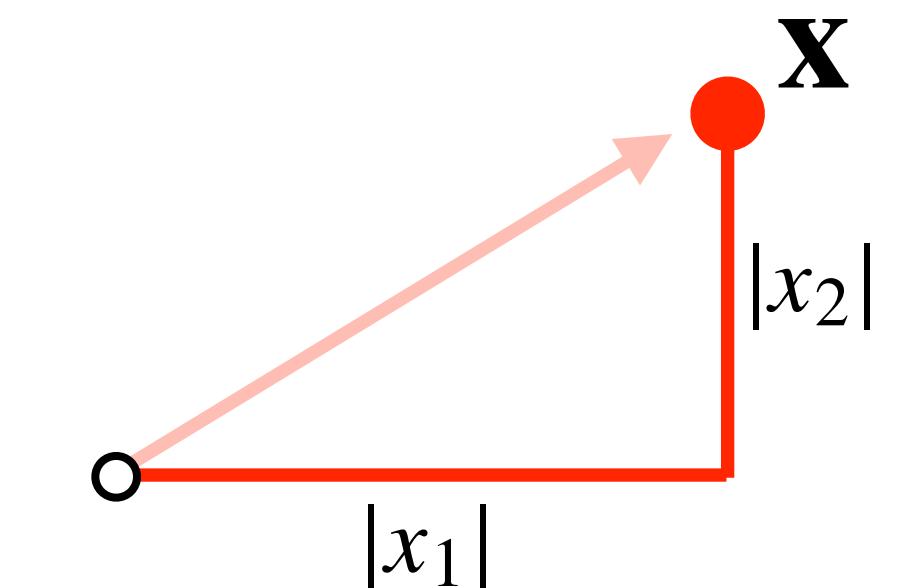
벡터의 노름 구해보기

- 벡터의 노름(norm)은 원점에서부터의 거리를 말합니다
- L_1 -노름은 각 성분의 변화량의 절대값을 모두 더합니다

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}$$

$$\|\mathbf{x}\|_1 = \sum_{i=1}^d |x_i|$$

$$\|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^d |x_i|^2}$$



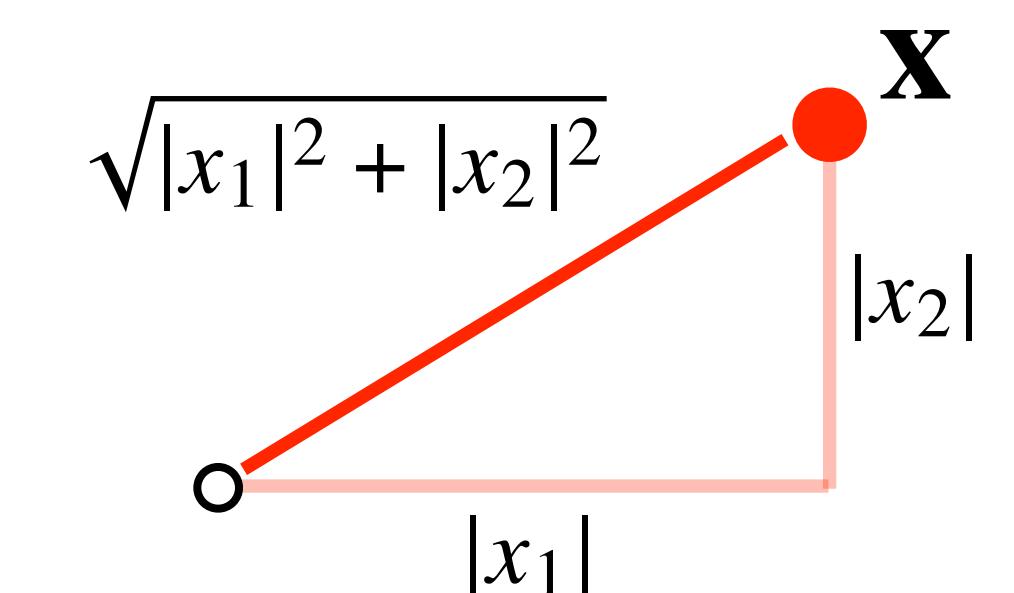
벡터의 노름 구해보기

- 벡터의 노름(norm)은 원점에서부터의 거리를 말합니다
- L_1 -노름은 각 성분의 변화량의 절대값을 모두 더합니다
- L_2 -노름은 피타고라스 정리를 이용해 유클리드 거리를 계산합니다

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}$$

$$\|\mathbf{x}\|_1 = \sum_{i=1}^d |x_i|$$

$$\|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^d |x_i|^2}$$

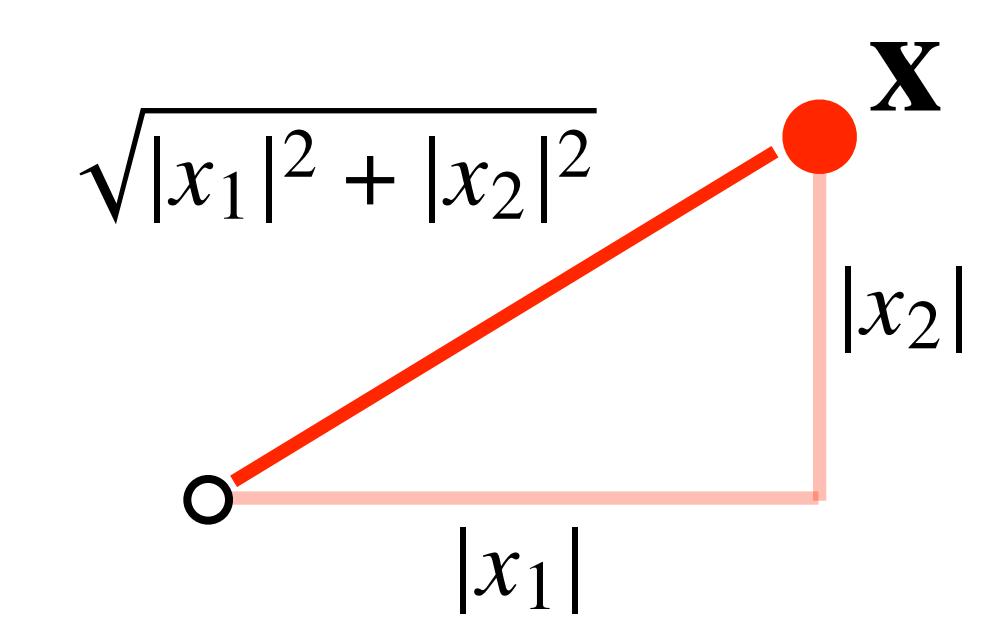


벡터의 노름 구해보기

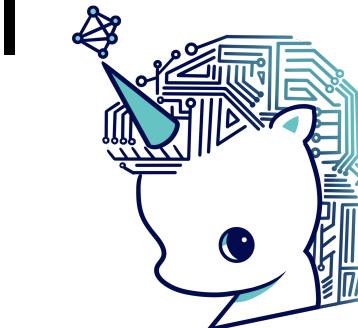
- 벡터의 노름(norm)은 원점에서부터의 거리를 말합니다
- L_1 -노름은 각 성분의 변화량의 절대값을 모두 더합니다
- L_2 -노름은 피타고라스 정리를 이용해 유clidean 거리를 계산합니다

```
1 def l1_norm(x):  
2     x_norm = np.abs(x)  
3     x_norm = np.sum(x_norm)  
4     return x_norm  
5  
6 def l2_norm(x):  
7     x_norm = x*x  
8     x_norm = np.sum(x_norm)  
9     x_norm = np.sqrt(x_norm)  
10    return x_norm
```

$$\|\mathbf{x}\|_1 = \sum_{i=1}^d |x_i|$$



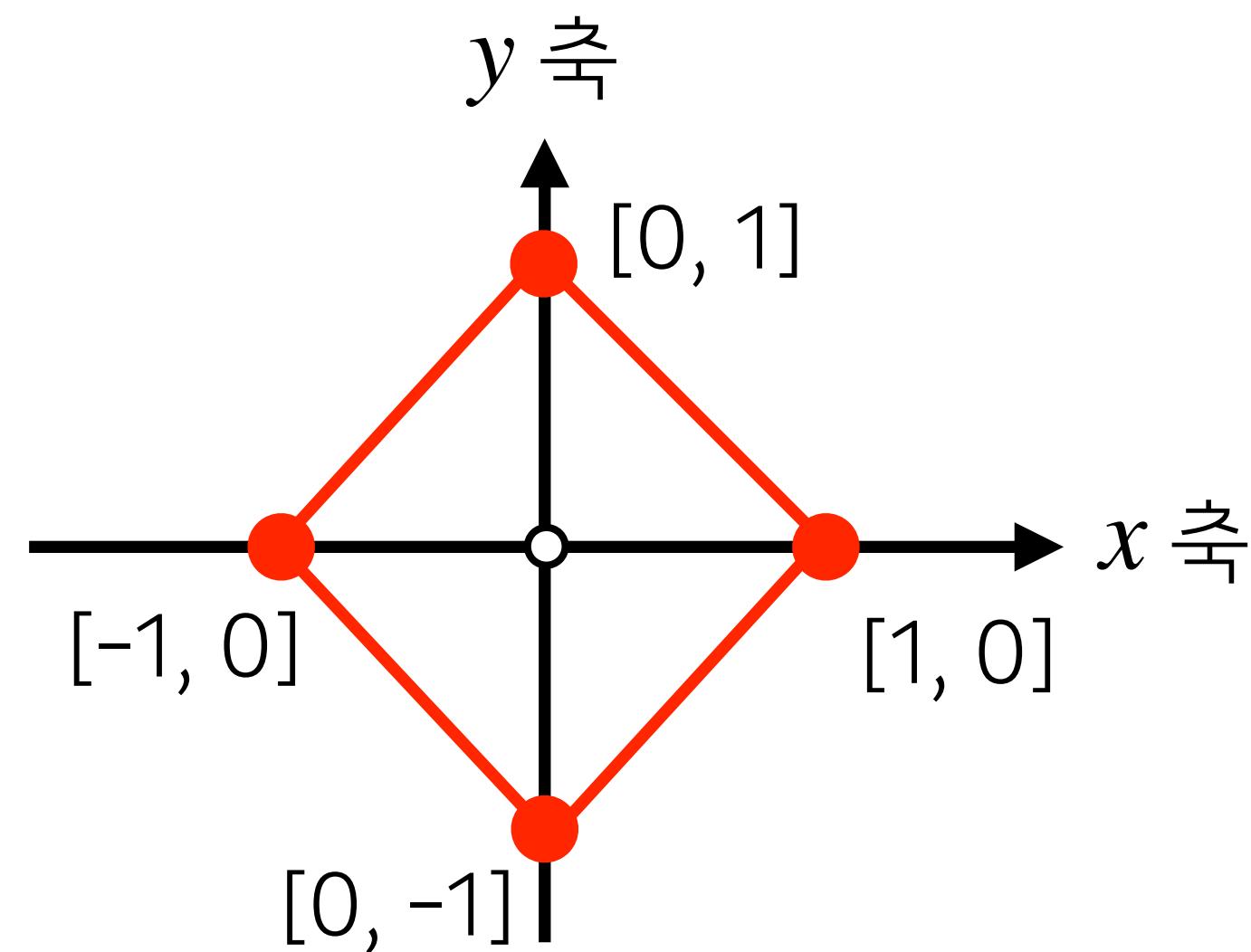
$$\|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^d |x_i|^2}$$



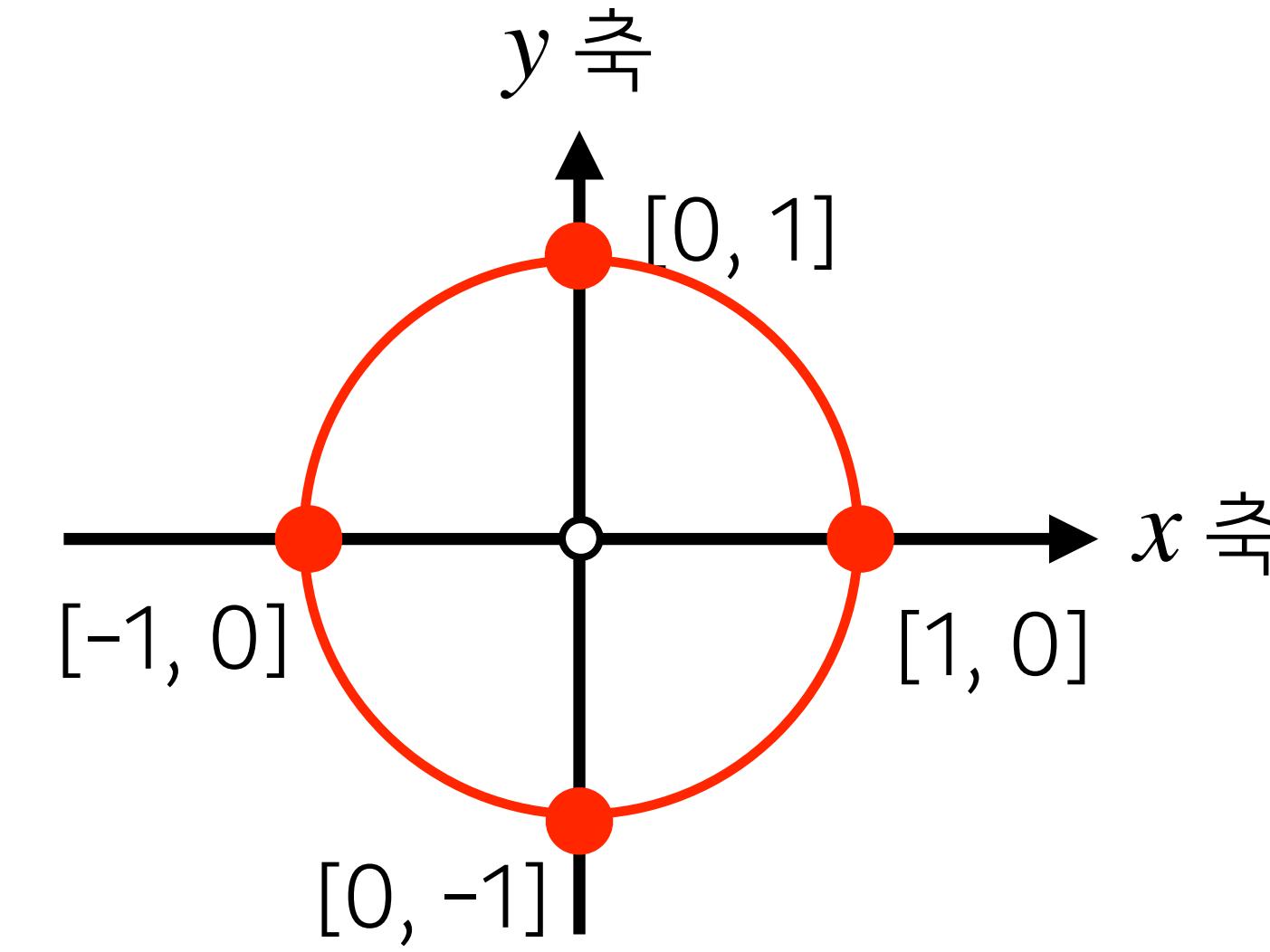
L_2 -노름은 `np.linalg.norm`을 이용해도 구현 가능하다

왜 다른 노름을 소개하나요?

- 노름의 종류에 따라 기하학적 성질이 달라집니다



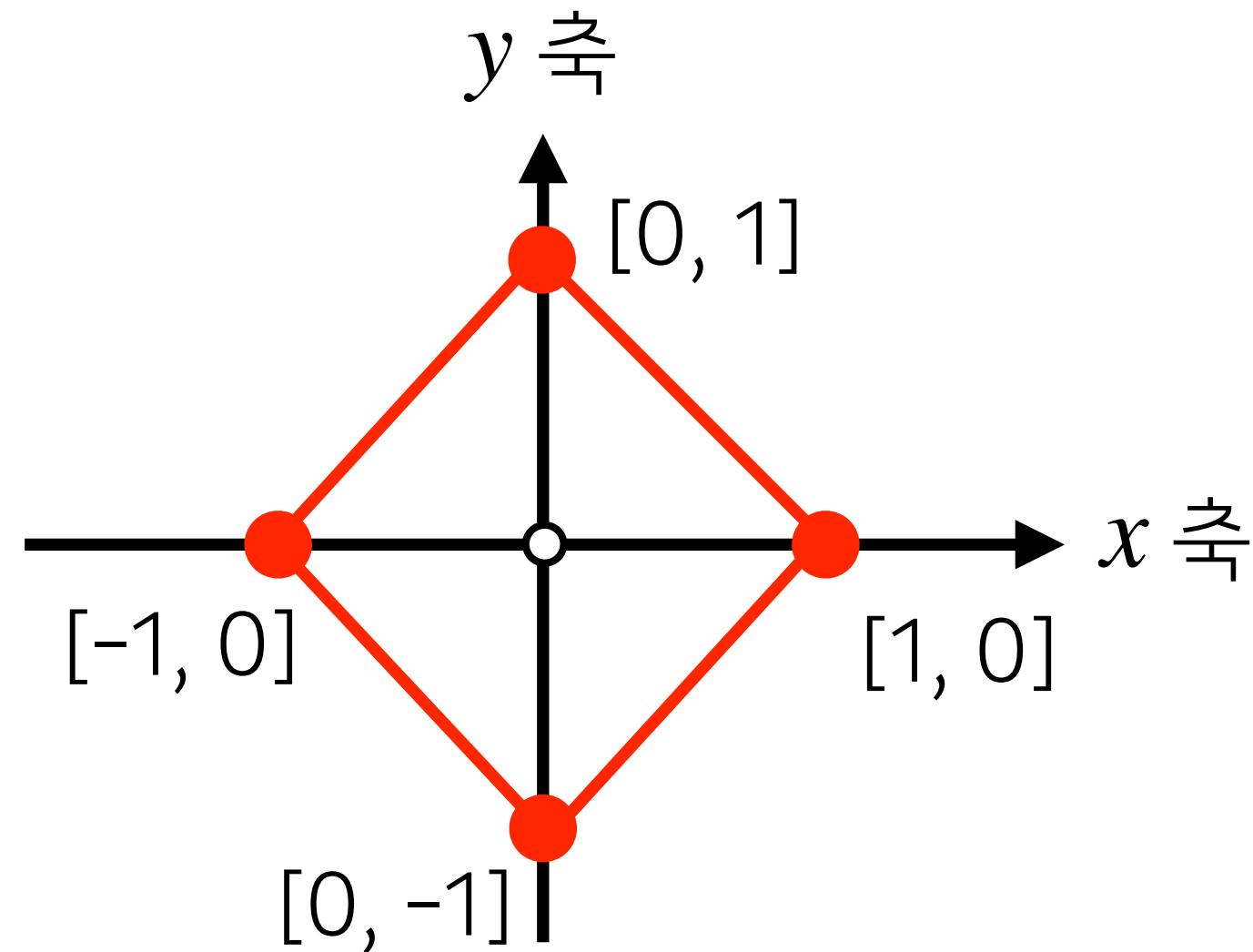
L_1 -노름 상의 원: $\{\mathbf{x} : \|\mathbf{x}\|_1 = 1\}$



L_2 -노름 상의 원: $\{\mathbf{x} : \|\mathbf{x}\|_2 = 1\}$

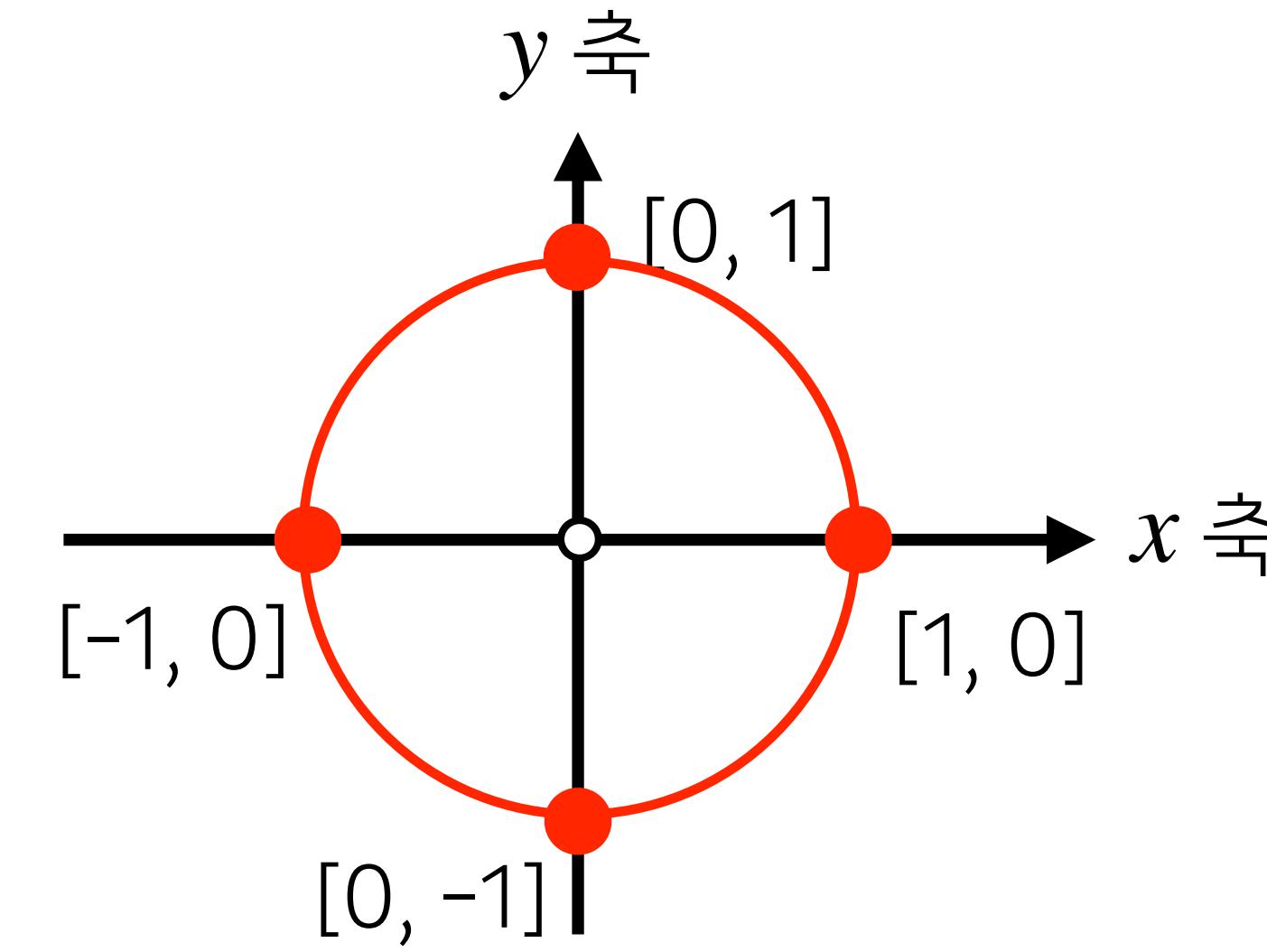
왜 다른 노름을 소개하나요?

- 노름의 종류에 따라 **기하학적 성질**이 달라집니다
- 머신러닝에선 각 성질들이 필요할 때가 있으므로 둘 다 사용합니다



L_1 -노름 상의 원: $\{\mathbf{x} : \|\mathbf{x}\|_1 = 1\}$

예: Robust 학습, Lasso 회귀

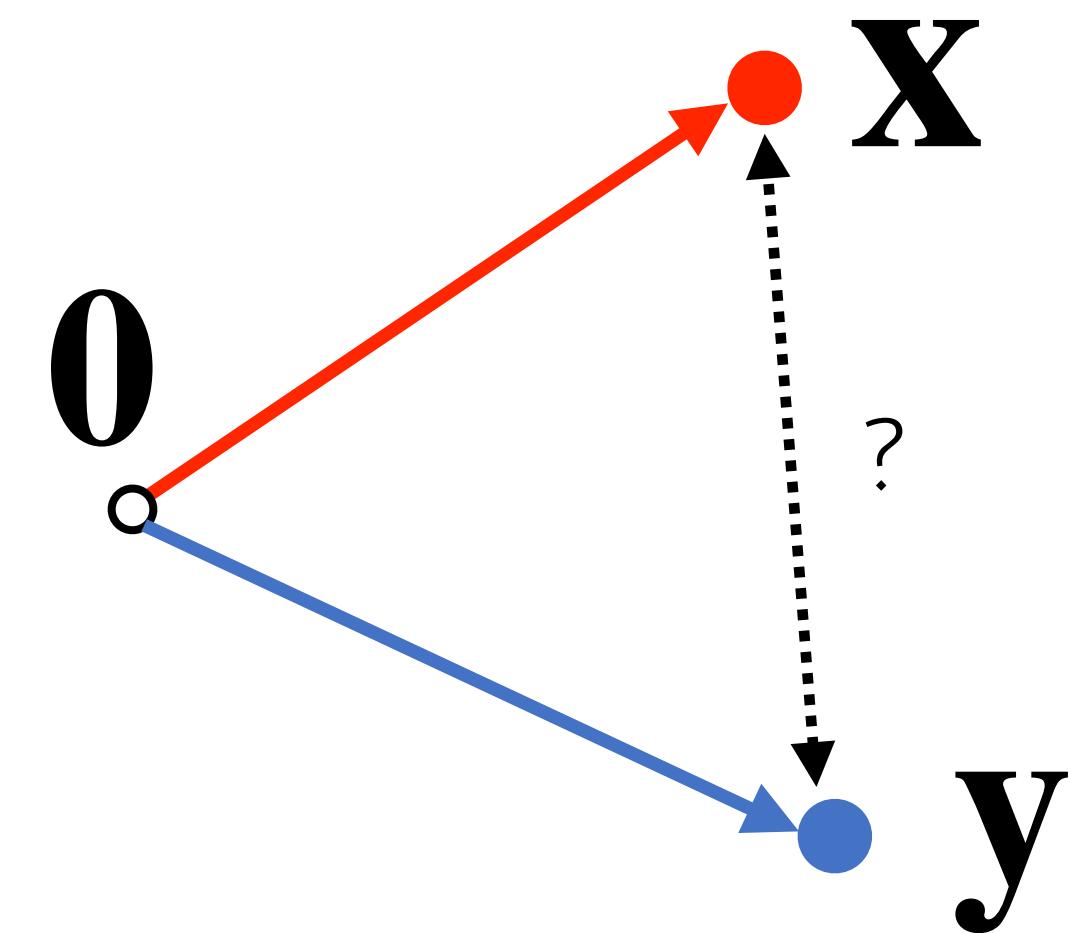


L_2 -노름 상의 원: $\{\mathbf{x} : \|\mathbf{x}\|_2 = 1\}$

예: Laplace 근사, Ridge 회귀

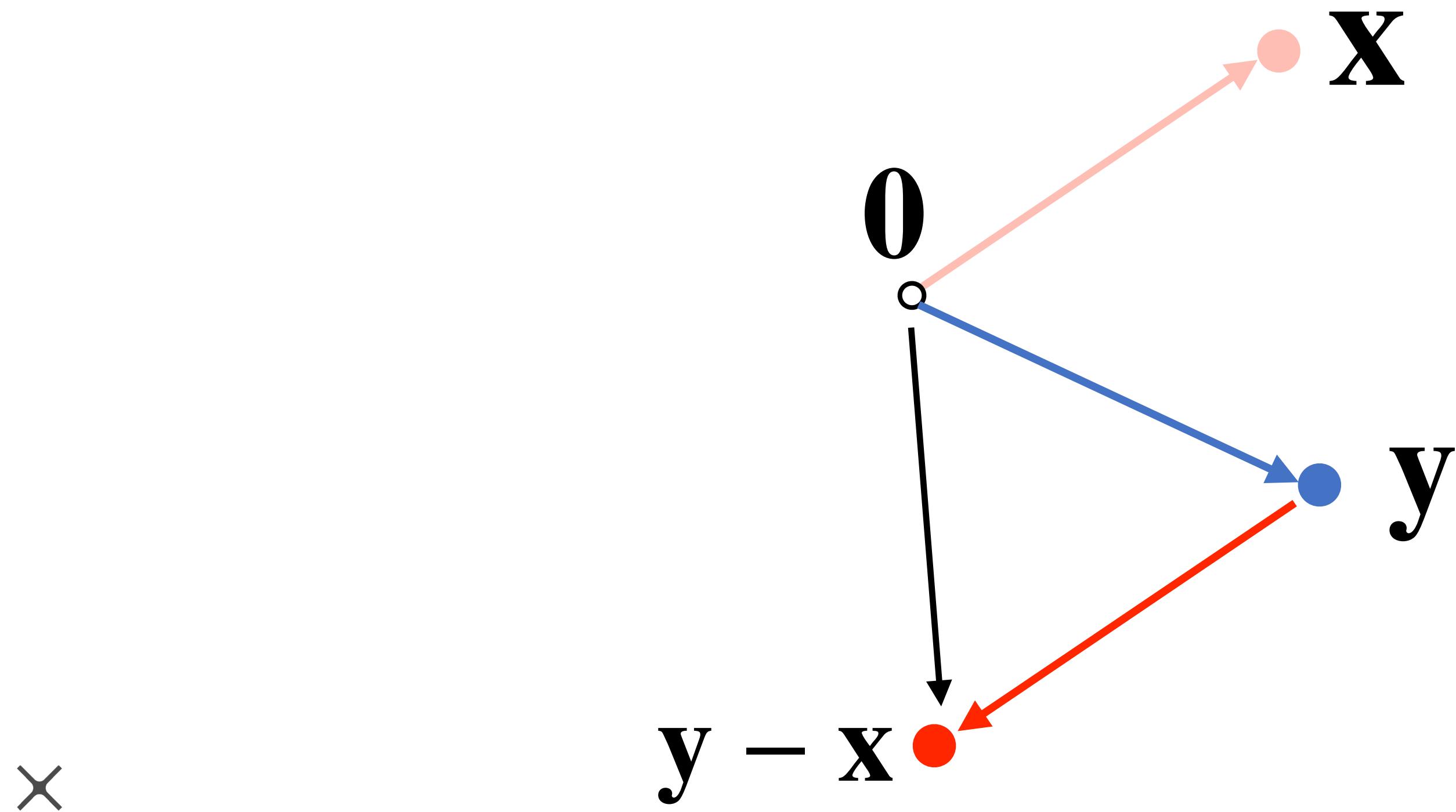
두 벡터 사이의 거리를 구해보자!

- L_1, L_2 -노름을 이용해 두 벡터 사이의 거리를 계산할 수 있습니다



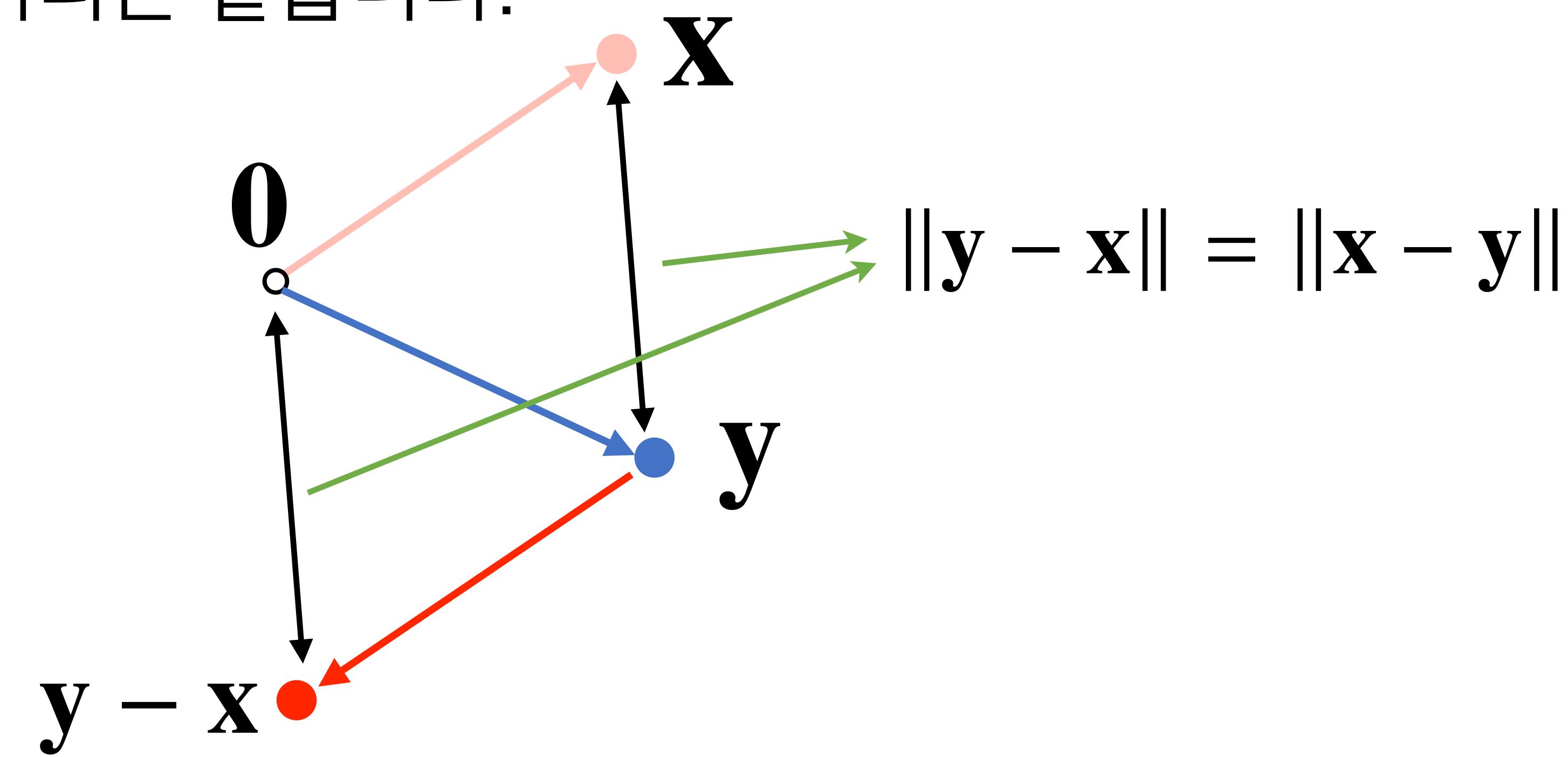
두 벡터 사이의 거리를 구해보자!

- L_1, L_2 -노름을 이용해 두 벡터 사이의 거리를 계산할 수 있습니다
- 두 벡터 사이의 거리를 계산할 때는 벡터의 뺄셈을 이용합니다



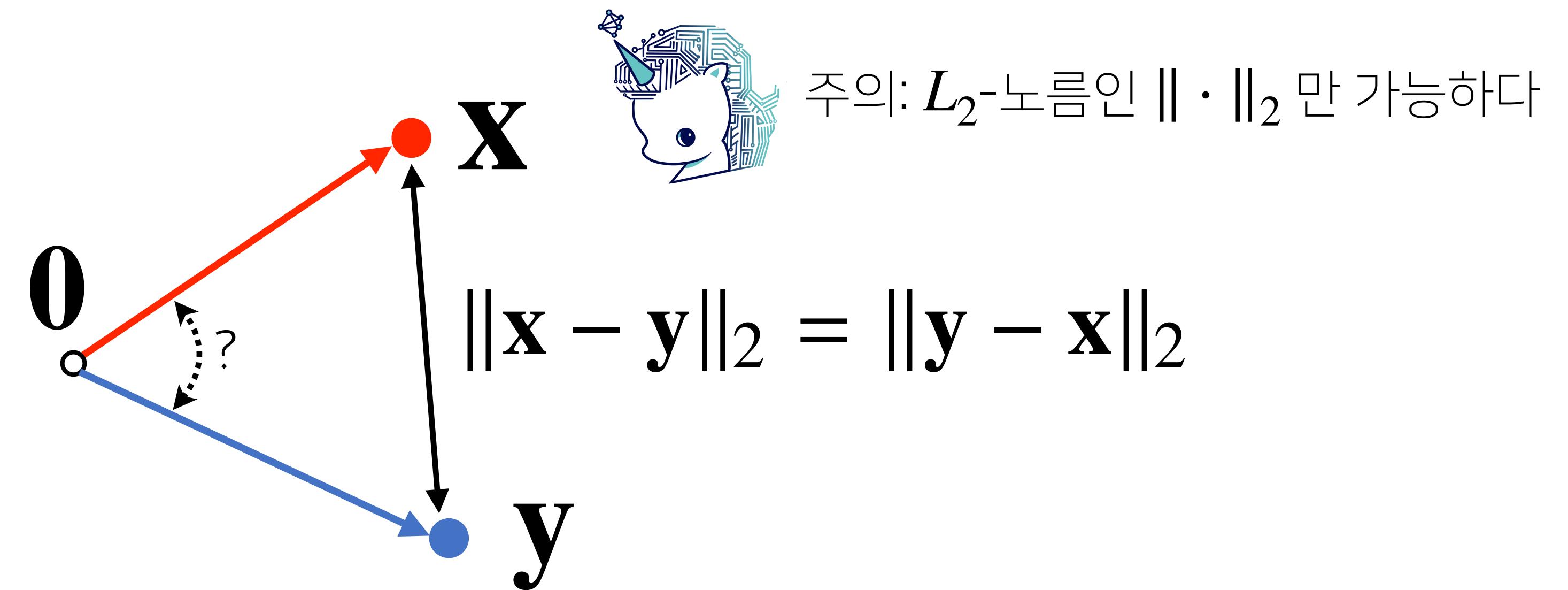
두 벡터 사이의 거리를 구해보자!

- L_1, L_2 -노름을 이용해 두 벡터 사이의 거리를 계산할 수 있습니다
- 두 벡터 사이의 거리를 계산할 때는 벡터의 뺄셈을 이용합니다
- 뺄셈을 거꾸로 해도 거리는 같습니다!



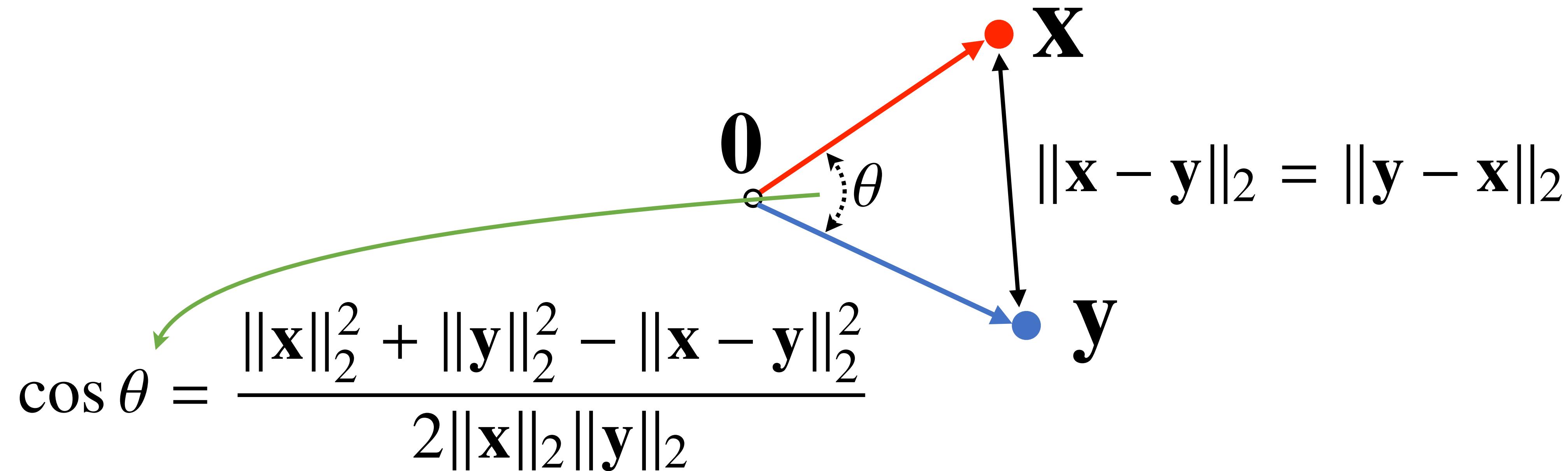
두 벡터 사이의 각도 구해보기

- 두 벡터 사이의 거리를 이용하여 각도도 계산해 볼 수 있을까?



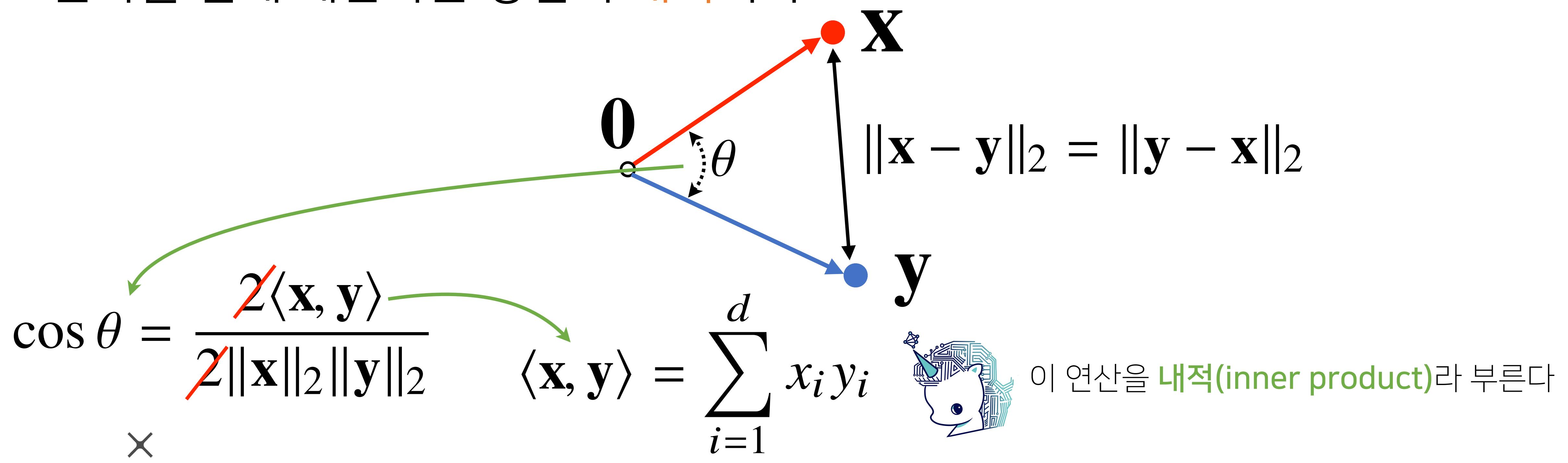
두 벡터 사이의 각도 구해보기

- 두 벡터 사이의 거리를 이용하여 각도도 계산해 볼 수 있을까?
- 제2 코사인 법칙에 의해 두 벡터 사이의 각도를 계산할 수 있다



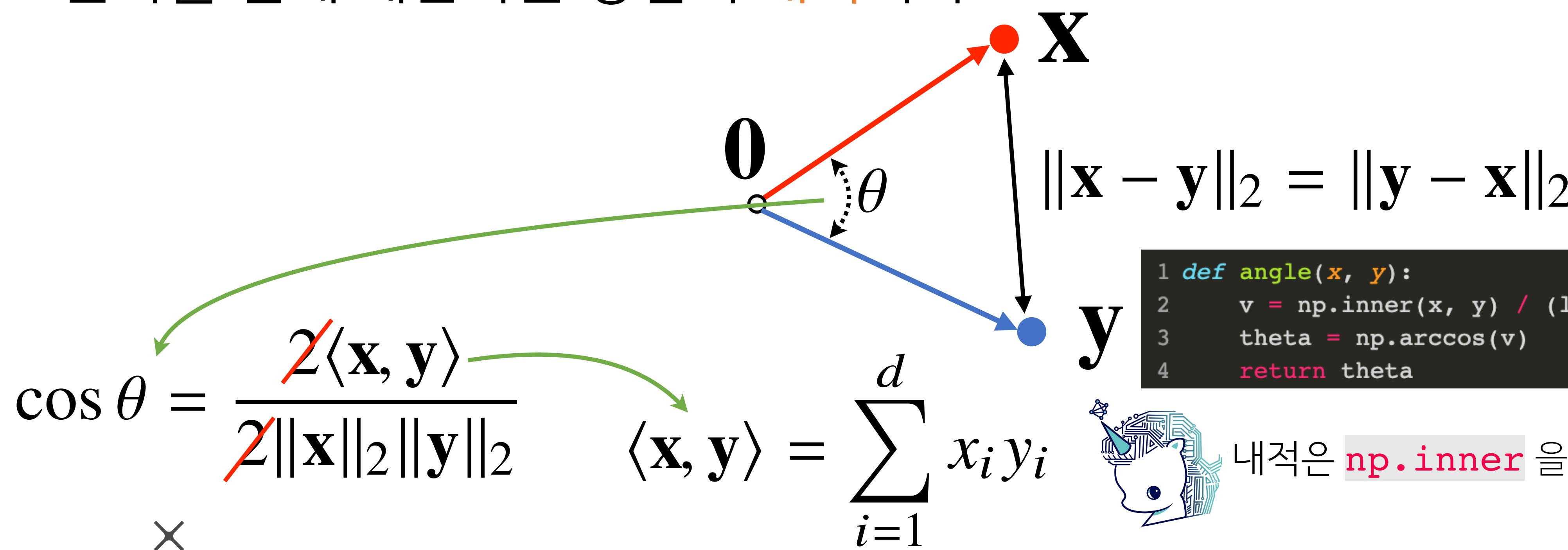
두 벡터 사이의 각도 구해보기

- 두 벡터 사이의 거리를 이용하여 각도도 계산해 볼 수 있을까?
- 제2 코사인 법칙에 의해 두 벡터 사이의 각도를 계산할 수 있다
- 분자를 쉽게 계산하는 방법이 내적이다



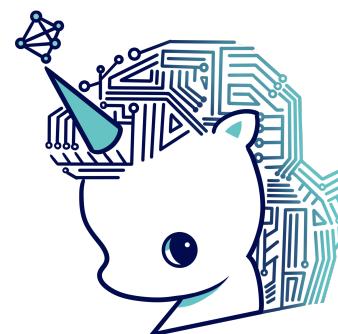
두 벡터 사이의 각도 구해보기

- 두 벡터 사이의 거리를 이용하여 각도도 계산해 볼 수 있을까?
- 제2 코사인 법칙에 의해 두 벡터 사이의 각도를 계산할 수 있다
- 분자를 쉽게 계산하는 방법이 내적이다

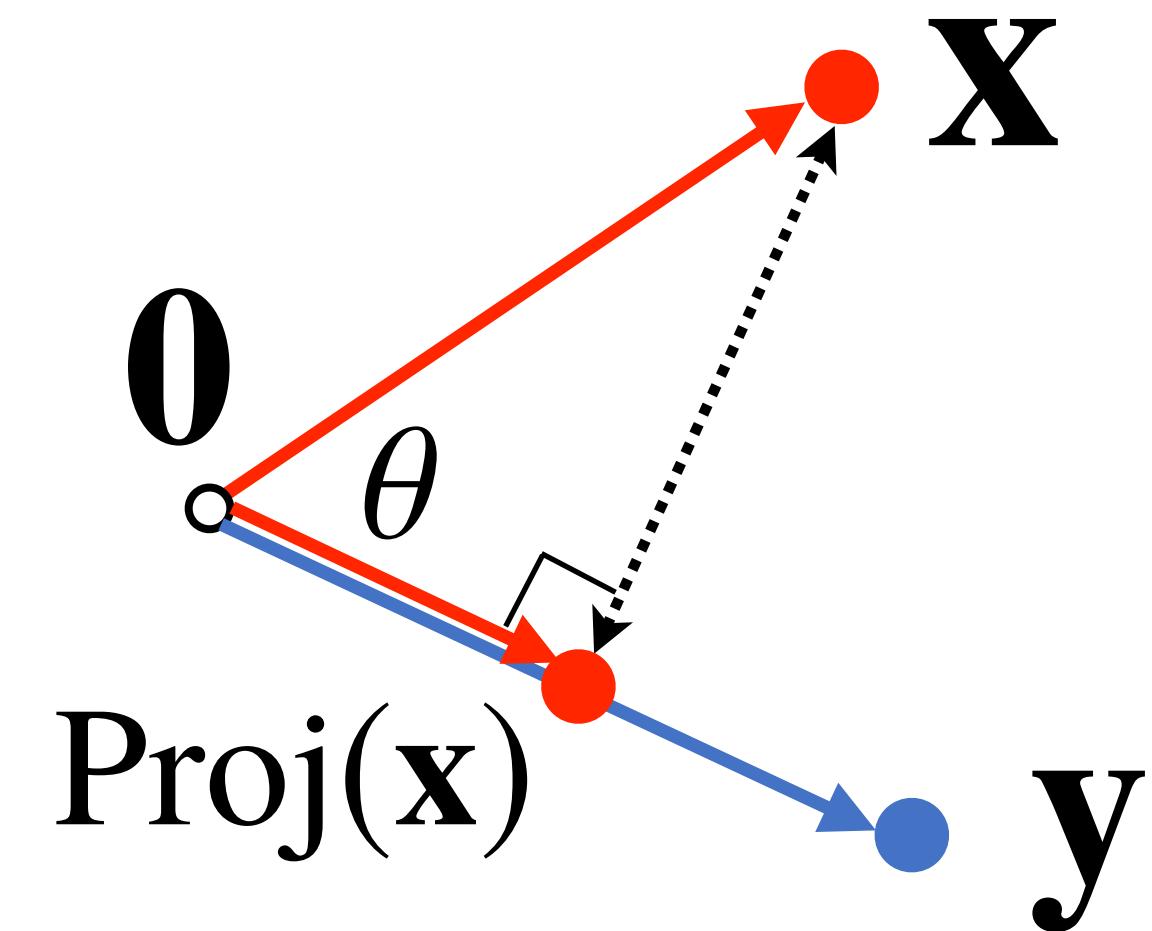


내적은 어떻게 해석할까?

- 내적은 정사영(orthogonal projection)된 벡터의 길이와 관련 있다

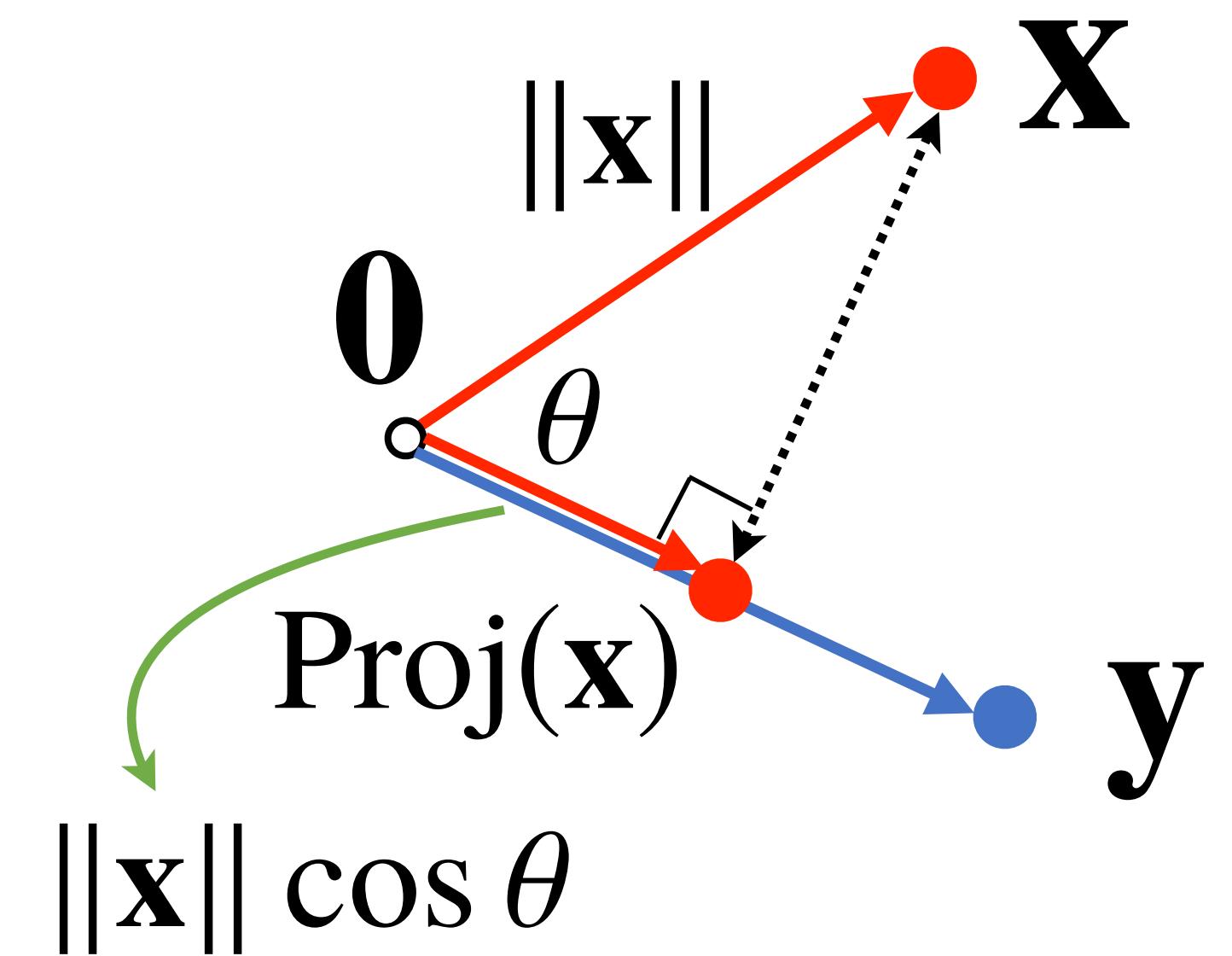


Proj(\mathbf{x}) 는 벡터 \mathbf{y} 로 정사영된 벡터 \mathbf{x} 의 **그림자**를 의미한다



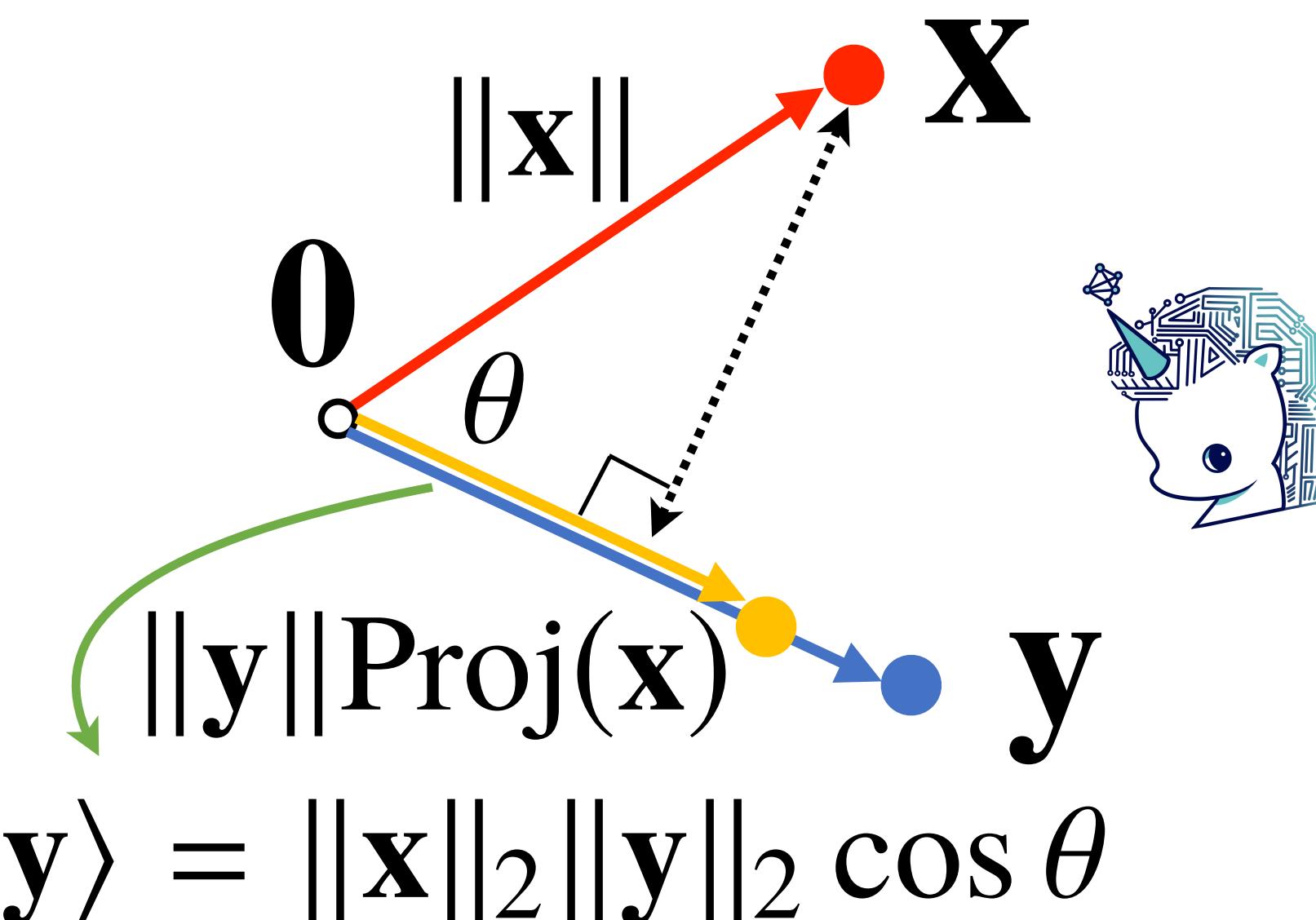
내적은 어떻게 해석할까?

- 내적은 정사영(orthogonal projection)된 벡터의 길이와 관련 있다
- $\text{Proj}(\mathbf{x})$ 의 길이는 코사인법칙에 의해 $\|\mathbf{x}\| \cos \theta$ 가 된다



내적은 어떻게 해석할까?

- 내적은 정사영(orthogonal projection)된 벡터의 길이와 관련 있다
- $\text{Proj}(\mathbf{x})$ 의 길이는 코사인법칙에 의해 $\|\mathbf{x}\| \cos \theta$ 가 된다
- 내적은 정사영의 길이를 벡터 \mathbf{y} 의 길이 $\|\mathbf{y}\|$ 만큼 조정한 값이다



내적은 두 벡터의 유사도(similarity)를 측정하는데 사용 가능하다

THE END

다음 시간에 보아요!