

# Deep Learning Programming

## Lecture 2.1: Pytorch Introduction

Sangryul Jeon

School of Computer Science and Engineering

[srjeonn@pusan.ac.kr](mailto:srjeonn@pusan.ac.kr)

# **PART 3:** Tensor Operations Vectors & Matrices

## 1. Tensor의 노름

- 1.1 1-D Tensor 복습
- 1.2  $\text{L}_n$  노름
- 1.3 학습정리

## 2. 유사도

- 2.1 유사도
- 2.2 맨해튼 유사도
- 2.3 유클리드 유사도
- 2.4 코사인 유사도
- 2.5 학습정리

## 3. 2-D Tensor 행렬 연산

- 3.1 2-D Tensor 행렬 곱셈 연산
- 3.2 2-D Tensor 행렬 곱셈 연산 활용
- 3.3 학습정리

1.

## Tensor의 노름

이번 챕터에서는  $L_1$ ,  $L_2$ ,  $L_\infty$  노름의 정의와 여러가지 표현 방식에 대해 살펴봅니다.

### 1-D Tensor(=Vector) Review

- 1강에서 1-D Tensor(=Vector)가 무엇인지에 대해서 살펴보았음
  - 순서가 지정된 여러 개의 숫자들이 일렬로 나열된 구조

$$\begin{aligned}a &= [4.0, 3.0] \\&= [a_1 \ a_2]\end{aligned}$$

↑  
요소 또는 성분

4.0	3.0
-----	-----

`a = torch.tensor([4.0, 3.0])`

## 1-D Tensor(=Vector) Review

- 2강에서 1-D Tensor(=Vector)의 크기에 대해서 살펴보았음
  - 1-D Tensor의 크기는 1-D Tensor 축에 포함된 요소의 개수를 의미함

$$\begin{aligned}a &= [4.0, 3.0] \\&= [a_1, a_2]\end{aligned}$$



`a.shape` 또는 `a.size()`

### 1-D Tensor(=Vector) Review

- 2강에서 1-D Tensor(=Vector)의 크기에 대해서 살펴보았음
  - 1-D Tensor의 크기는 1-D Tensor 축에 포함된 요소의 개수를 의미함
  - 요소의 개수가 더 많다고 해서 크기가 더 크다고 판단하는 것은 아님

$$\begin{aligned}a &= [4.0, 3.0] \\&= [a_1, a_2]\end{aligned}$$



a.shape 또는 a.size()

## 노름이란

- 1-D Tensor의 노름은 Vector가 원점에서 얼마나 떨어져 있는지를 의미함
- 이러한 1-D Tensor의 노름은 Vector의 길이를 측정하는 방법으로 사용됨
- 또한 1-D Tensor의 노름에는 L1 노름, L2 노름, L $\infty$  노름 등 여러가지 유형의 노름이 존재함

$$\|x\|_p = \left( \sum_{i=1}^n |x_i|^p \right)^{\frac{1}{p}} \quad (p \geq 1) \quad (L_p \text{ 노름의 수식 표현})$$

### L1 노름의 정의와 수식 표현

- L1 노름은 1-D Tensor에 포함된 요소의 절대값의 합으로 정의할 수 있음
- $x = [x_1, x_2, \dots, x_n]$  일 때, 수식 표현은 다음과 같음

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

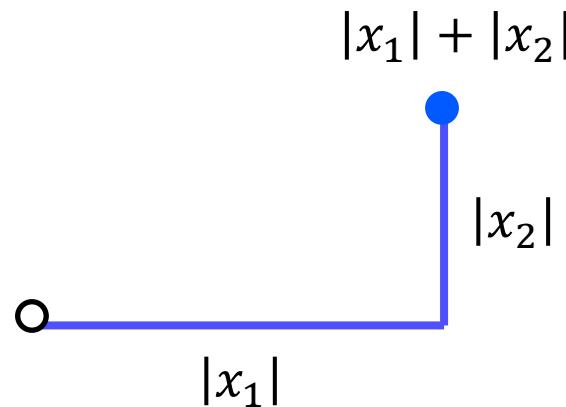
## 1.2 Ln 노름

### 1. Tensor의 노름

---

#### L1 노름의 기하학적 표현

- L1 노름은 1-D Tensor에 포함된 요소의 절대값의 합으로 정의할 수 있음
- $x = [x_1, x_2]$  일 때, L1 노름의 기하학적 표현은 다음과 같음

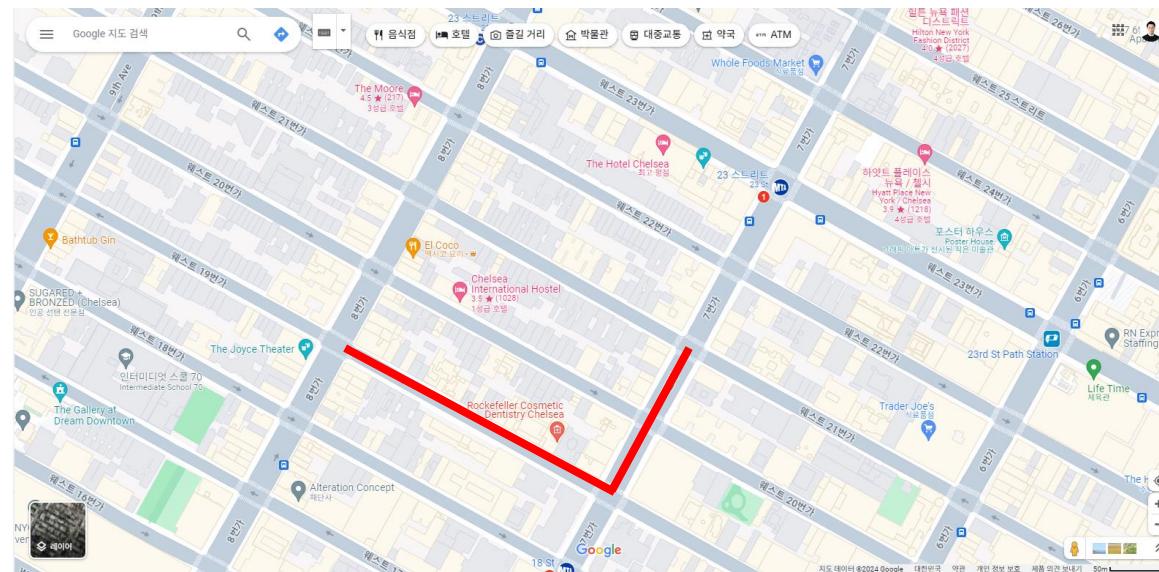


## 1.2 Ln 노름

### 1. Tensor의 노름

#### 맨해튼 노름

- L1 노름은 1-D Tensor에 포함된 요소의 절대값의 합으로 정의할 수 있음
- 또한 L1 노름을 뉴욕시 맨해튼의 격자형 도로망과 유사하다고 해서 맨해튼 노름이라고도 부름



## 1.2 Ln 노름

### 1. Tensor의 노름

---

#### L1 노름의 코드 표현

- $a = \text{torch.tensor}([4.0, 3.0])$ 일 때, L1 노름의 값을 구하는 [코드 표현](#)
  - `torch.norm(a, p = 1)`

## 1.2 Ln 노름

1. Tensor의 노름

---

### L2 노름의 정의와 수식 표현

- L2 노름은 1-D Tensor에 포함된 요소의 제곱합의 제곱근으로 정의할 수 있음
- $x = [x_1, x_2, \dots, x_n]$  일 때, 수식 표현은 다음과 같음

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

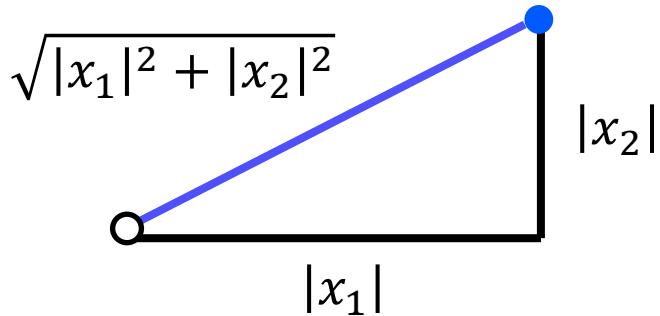
## 1.2 Ln 노름

### 1. Tensor의 노름

---

#### L2 노름의 기하학적 표현

- L2 노름은 1-D Tensor에 포함된 요소의 제곱합의 제곱근으로 정의할 수 있음
- $x = [x_1, x_2]$  일 때, L2 노름의 기하학적 표현은 다음과 같음

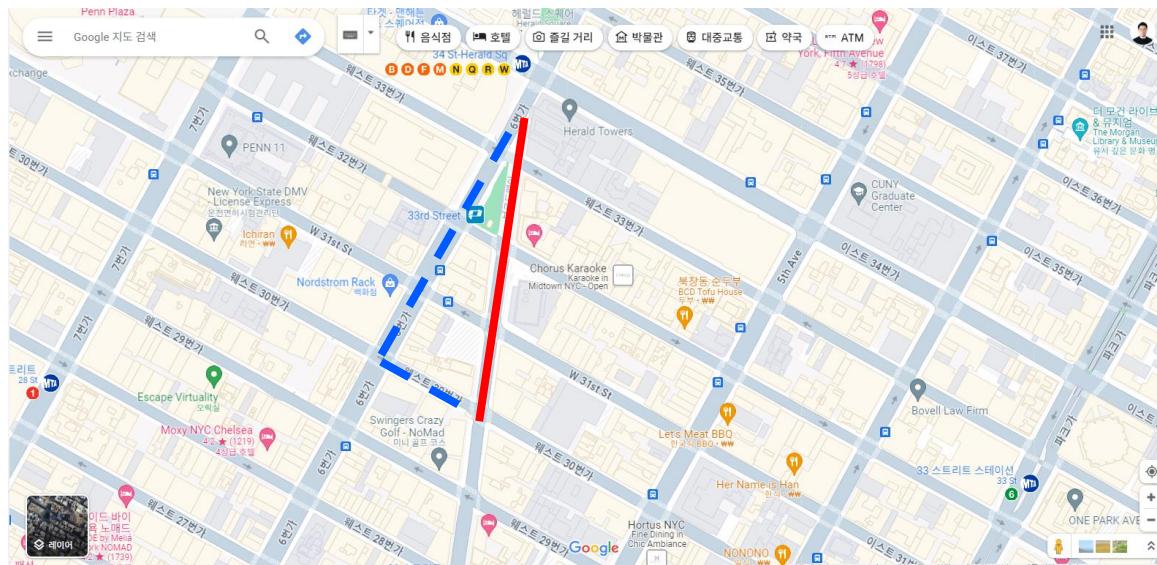


## 1.2 Ln 노름

### 1. Tensor의 노름

#### 유클리드 노름

- L2 노름은 1-D Tensor에 포함된 요소의 제곱합의 제곱근으로 정의할 수 있음
- 또한 L2 노름을 유클리드 공간에서 두 점 사이의 최단 거리를 측정하는 방법과 같다고 해서 유클리드 노름이라고도 부름



출처: Google 지도. <https://www.google.com/maps/@40.7478824,-73.9871117,17.18z?entry=ttu>

### L2 노름의 코드 표현

- $a = \text{torch.tensor}([4.0, 3.0])$ 일 때, L2 노름의 값을 구하는 [코드 표현](#)
  - `torch.norm(a, p = 2)`

## 1.2 Ln 노름

### 1. Tensor의 노름

---

#### $L^\infty$ 노름의 정의와 수식 표현

- $L^\infty$  노름은 1-D Tensor에 포함된 요소의 절대값 중 최대값으로 정의할 수 있음
- $x = [x_1, x_2, \dots, x_n]$  일 때, 수식 표현은 다음과 같음

$$\|x\|_\infty = \max(|x_1|, |x_2|, \dots, |x_n|)$$

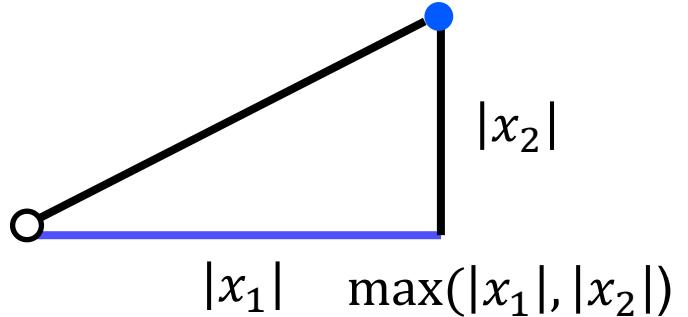
## 1.2 Ln 노름

### 1. Tensor의 노름

---

#### $L^\infty$ 노름의 기하학적 표현

- $L^\infty$  노름은 1-D Tensor에 포함된 요소의 절대값 중 최대값으로 정의할 수 있음
- $x = [x_1, x_2, \dots, x_n]$  일 때,  $L^\infty$  노름의 기하학적 표현은 다음과 같음



## 1.2 Ln 노름

### 1. Tensor의 노름

---

#### $L^\infty$ 노름의 코드 표현

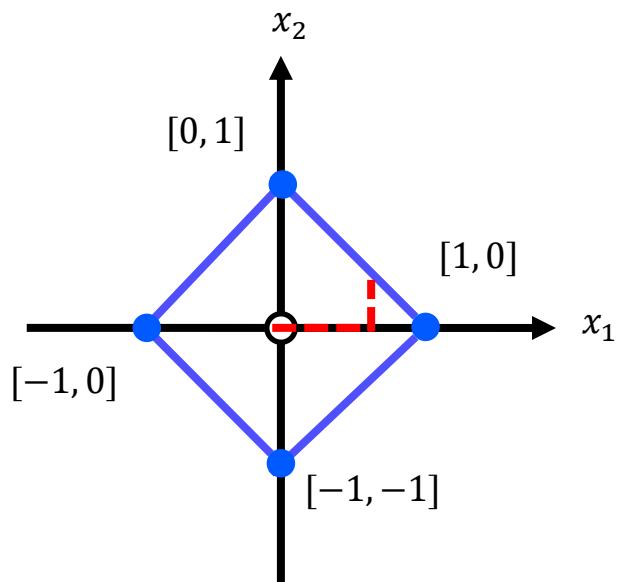
- $a = \text{torch.tensor}([4.0, 3.0])$ 일 때,  $L^\infty$  노름의 값을 구하는 [코드 표현](#)
  - `torch.norm(a, p = float('inf'))`

## 1.2 Ln 노름

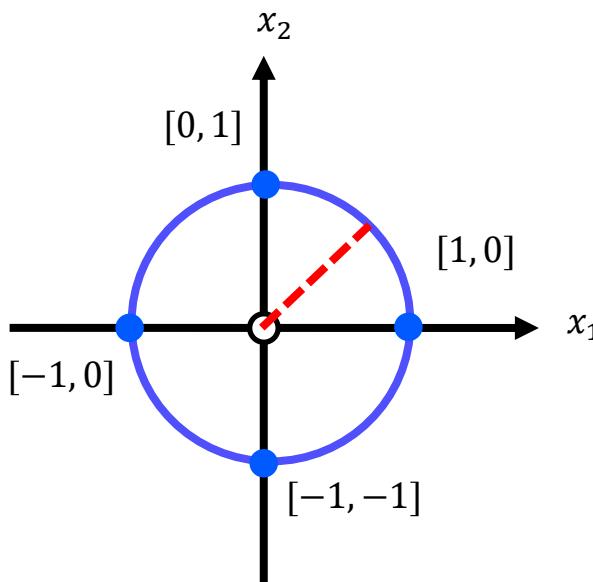
### 1. Tensor의 노름

#### 노름에 따른 기하학적 의미

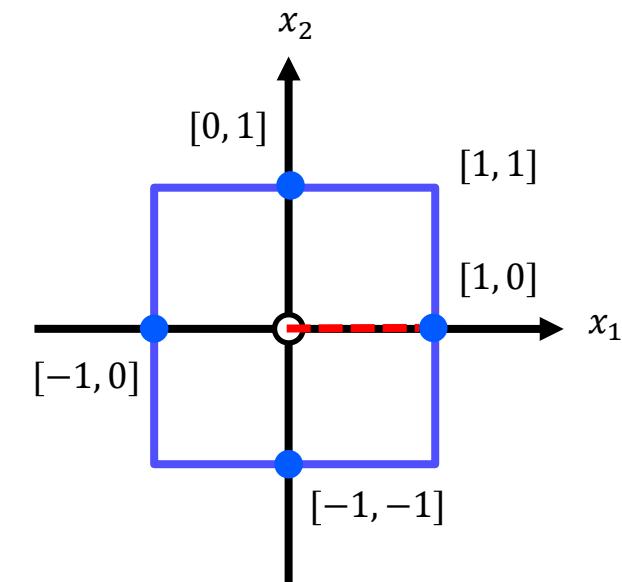
- 노름의 유형에 따라 각 노름이 1인 점들의 집합을 좌표평면에 표현하면 다음과 같음



$$\|x\|_1 = 1$$



$$\|x\|_2 = 1$$



$$\|x\|_\infty = 1$$

#### Tensor의 노름

- 1-D Tensor의 노름은 Vector가 원점에서 얼마나 떨어져 있는지를 의미한다.
- L1 노름은 1-D Tensor에 포함된 요소의 절대값의 합으로 정의할 수 있다.
- L2 노름은 1-D Tensor에 포함된 요소의 제곱합의 제곱근으로 정의할 수 있다.
- $L^\infty$  노름은 1-D Tensor에 포함된 요소의 절대값 중 최대값으로 정의할 수 있다.

2.

## 유사도

이번 챕터에서는 맨해튼 유사도, 유클리드 유사도, 코사인 유사도에 대해 살펴봅니다.

### 유사도

- 유사도(Similarity)란 두 1-D Tensor(=Vector)가 얼마나 유사한지에 대한 측정값을 의미함
- 이러한 유사도는 군집화(Clustering) 알고리즘에서 데이터들이 얼마나 유사한지를 판단하는 중요한 기준으로 사용됨
  - 군집화란 비지도 학습의 한 형태로서, 유사한 데이터들을 그룹으로 묶는 기법
- 유사도를 계산하는 방법에는 여러가지가 있으며, 본 강의에서는 맨해튼 유사도, 유clidean 유사도, 코사인 유사도를 소개함

### 맨해튼 유사도의 정의와 특징

- 맨해튼 유사도는 두 1-D Tensor 사이의 맨해튼 거리를 역수로 변환하여 계산한 값
- 두 1-D Tensor 사이의 맨해튼 거리의 값이 작아질 수록 맨해튼 유사도의 값은 커짐
- 맨해튼 유사도의 값은 1에 가까울수록 두 Tensor가 유사하다고 판단함

### 맨해튼 유사도의 수식 표현

- 맨해튼 유사도는 두 1-D Tensor 사이의 맨해튼 거리를 역수로 변환하여 계산한 값
- $x = [x_1, x_2, \dots, x_n]$ 이고,  $y = [y_1, y_2, \dots, y_n]$  일 때, 맨해튼 거리와 맨해튼 유사도의 수식 표현은 다음과 같음

$$\text{Manhattan Distance} = \sum_{i=1}^n |x_i - y_i|$$

$$\text{Manhattan Similarity} = \frac{1}{1 + \text{Manhattan Distance}}$$

### 맨해튼 유사도의 코드 표현

- $b = \text{torch.tensor}([1.0, 0.0, 2.0])$ ,  $c = \text{torch.tensor}([0.0, 1.0, 2.0])$ 일 때,  
맨해튼 거리를 구하는 [코드 표현](#)
  - `manhattan_distance = torch.norm(b - c, p = 1)`
- 맨해튼 유사도를 구하는 [코드 표현](#)
  - `1 / (1 + manhattan_distance)`

### 유클리드 유사도의 정의와 특징

- 유클리드 유사도는 두 1-D Tensor 사이의 유클리드 거리를 역수로 변환하여 계산한 값
- 두 1-D Tensor 사이의 유클리드 거리의 값이 작아질수록 유클리드 유사도의 값은 커짐
- 유클리드 유사도의 값은 1에 가까울수록 두 Tensor가 유사하다고 판단함

### 유클리드 유사도의 수식 표현

- 유클리드 유사도는 두 1-D Tensor 사이의 유클리드 거리를 역수로 변환하여 계산한 값
- $x = [x_1, x_2, \dots, x_n]$ 이고,  $y = [y_1, y_2, \dots, y_n]$  일 때, 유클리드 거리와 유클리드 유사도의 수식 표현은 다음과 같음

$$Euclidean\ Distance = \sqrt{\sum_{i=1}^n |x_i - y_i|^2}$$

$$Euclidean\ Similarity = \frac{1}{1 + Euclidean\ Distance}$$

### 유클리드 유사도의 코드 표현

- $b = \text{torch.tensor}([1.0, 0.0, 2.0])$ ,  $c = \text{torch.tensor}([0.0, 1.0, 2.0])$ 일 때,  
유클리드 거리를 구하는 [코드 표현](#)
  - `euclidean_distance = torch.norm(b - c, p = 2)`
- 유클리드 유사도를 구하는 [코드 표현](#)
  - `1 / (1 + euclidean_distance)`

### 코사인 유사도의 정의와 특징

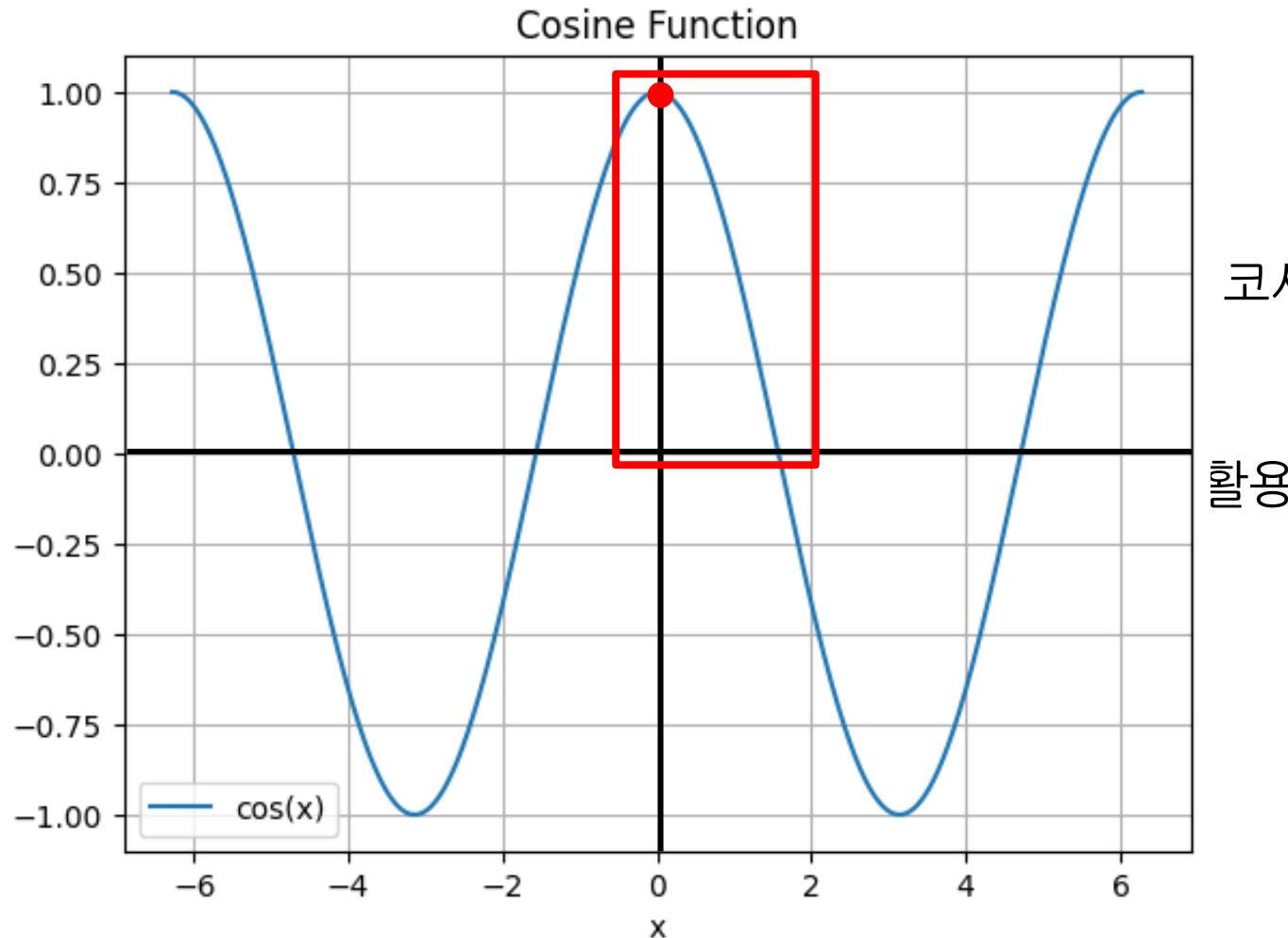
- 코사인 유사도는 두 1-D Tensor 사이의 각도를 측정하여 계산한 값
- 코사인 유사도는 각도를 측정하여 두 Tensor의 유사도를 판단하기 때문에 코사인 유사도의 값이 1에 가까울 수록 두 Tensor가 유사하다고 판단함

## 2.4 코사인 유사도

### 2. 유사도

#### 코사인 유사도의

- 코사인 유사도
- 코사인 유사도
- 1에 가까울 수
- 어떻게 두 1-[  
- 1-D Tens]



코사인 유사도의 값이  
활용

### 코사인 유사도의 정의와 특징

- 코사인 유사도는 두 1-D Tensor 사이의 각도를 측정하여 계산한 값
- 코사인 유사도는 각도를 측정하여 두 Tensor의 유사도를 판단하기 때문에 코사인 유사도의 값이 1에 가까울 수록 두 Tensor가 유사하다고 판단함
- 어떻게 두 1-D Tensor 사이의 각도를 측정할 수 있을까?
  - 1-D Tensor(=Vector)의 내적(dot product 또는 inner product)을 활용

### 1-D Tensor의 내적

- 1-D Tensor의 내적은 두 1-D Tensor 사이의 관계를 하나의 0-D Tensor(=Scalar)로 변환하는 것으로, 이러한 변환은 두 1-D Tensor의 유사성을 수량화할 수 있는 장점을 가짐
- 1-D Tensor의 내적을 구하는 방법에는 두 가지가 있음
  - 첫 번째, 두 1-D Tensor의 각 요소를 곱해서 더하는 방법
  - 두 번째, 두 1-D Tensor의 길이를 곱하는 방법

### 1-D Tensor의 내적

- 1-D Tensor의 내적을 구하는 첫 번째 방법은 두 Tensor의 각 요소를 곱해서 더하는 것임
- 두 1-D Tensor  $x = [x_1, x_2, \dots, x_n]$ ,  $y = [y_1, y_2, \dots, y_n]$ 의 내적에 대한 대수적 표현은 다음과 같음

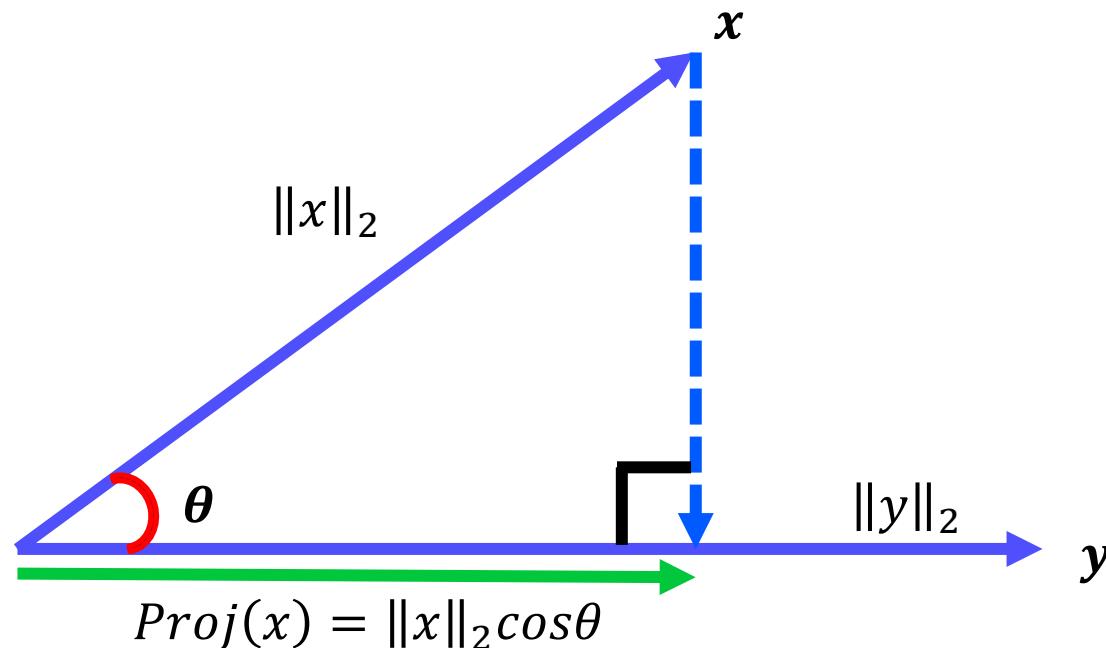
$$\langle x, y \rangle = x_1 \cdot y_1 + x_2 \cdot y_2 + \dots + x_n \cdot y_n$$

### 1-D Tensor의 내적

- 1-D Tensor의 내적을 구하는 첫 번째 방법은 두 Tensor의 각 요소를 곱해서 더하는 것임
- 두 1-D Tensor  $b = \text{torch.tensor}([1, 0, 2])$ 와  $c = \text{torch.tensor}([0, 1, 2])$ 의 내적에 대한  
코드 표현
  - `torch.dot(b, c)`

### 1-D Tensor의 내적

- 1-D Tensor의 내적을 구하는 두 번째 방법은 두 1-D Tensor의 길이를 곱하는 것임
- 두 1-D Tensor  $x$ 와  $y$ 가 이루는 각이  $\theta$ 일 때, 아래의 같이 도식화 할 수 있음



### 1-D Tensor의 내적

- 1-D Tensor의 내적은 각도의 성질이 포함된 1-D Tensor  $x$ 에 대한 정사영을 또 다른 1-D Tensor  $y$ 의 길이와 곱한 값
- 따라서 두 1-D Tensor  $x$ 와  $y$ 의 내적에 대한 대수적 표현은 다음과 같음

$$\langle x, y \rangle = \|x\|_2 \cos\theta \times \|y\|_2 = \|x\|_2 \|y\|_2 \cos\theta$$

## 2.4 코사인 유사도

2. 유사도

---

### 코사인 유사도의 수식 표현

- 코사인 유사도는 두 1-D Tensor 사이의 각도를 측정하여 계산한 값
- $x = [x_1, x_2, \dots, x_n]$ 이고,  $y = [y_1, y_2, \dots, y_n]$  일 때, 코사인 유사도의 수식 표현은 다음과 같음

$$\cos(x, y) = \frac{\langle x, y \rangle}{\|x\|_2 \|y\|_2} \quad (\because 1 - D Tensor의 내적)$$

## 2.4 코사인 유사도

2. 유사도

### 코사인 유사도의 수식 표현 유도

$$\langle x, y \rangle = \|x\|_2 \|y\|_2 \cos\theta$$

$$\Leftrightarrow \frac{\langle x, y \rangle}{\|x\|_2 \|y\|_2} = \cos\theta \quad \text{양변을 } \|x\|_2 \|y\|_2 \text{로 나눔}$$

$$\Leftrightarrow \cos\theta = \frac{\langle x, y \rangle}{\|x\|_2 \|y\|_2} \quad \text{좌변과 우변을 바꿈}$$

$$\Leftrightarrow \cos(x, y) = \frac{\langle x, y \rangle}{\|x\|_2 \|y\|_2} \quad \theta \text{는 1-D Tensor } x, y \text{에 대한 함수로 표현 가능하므로 } (x, y) \text{로 작성}$$

# Thank you

**Prof. Jeon, Sangryul**

Computer Vision Lab.

Pusan National University, Korea

Tel: +82-51-510-2423

Web: <http://sr-jeon.github.io/>

E-mail: [srjeonn@pusan.ac.kr](mailto:srjeonn@pusan.ac.kr)

### 코사인 유사도의 코드 표현

- $b = \text{torch.tensor}([1.0, 0.0, 2.0])$ ,  $c = \text{torch.tensor}([0.0, 1.0, 2.0])$ 일 때,  
코사인 유사도를 구하는 [코드 표현](#)
  - `cosine_similarity = torch.dot(b, c) / (torch.norm(b, p = 2) * (torch.norm(c, p = 2))`

### 유사도

- 유사도(Similarity)란 두 1-D Tensor(=Vector)가 얼마나 유사한지에 대한 측정값을 의미한다.
- 맨해튼 유사도는 두 1-D Tensor 사이의 맨해튼 거리를 역수로 변환하여 계산한 값이다.
- 유클리드 유사도는 두 1-D Tensor 사이의 유클리드 거리를 역수로 변환하여 계산한 값이다.
- 코사인 유사도는 두 1-D Tensor 사이의 각도를 측정하여 계산한 값이다.

3.

## 2-D Tensor의 곱셈 연산

이번 챕터에서는 2-D Tensor의 곱셈 연산과 그 활용에 대해 살펴봅니다.

## 3.1 2-D Tensor의 행렬 곱셈 연산

### 3. 2-D Tensor의 행렬 곱셈 연산

---

#### 2-D Tensor의 행렬 곱셈 연산

- 2-D Tensor(=Matrix)의 행렬 곱셈은 두 행렬을 결합하여 새로운 행렬을 생성하는 연산
- 2-D Tensor의 행렬 곱셈은 신경망 구현에 핵심이 되는 연산임

## 3.1 2-D Tensor의 행렬 곱셈 연산

### 3. 2-D Tensor의 행렬 곱셈 연산

#### 2-D Tensor의 행렬 곱셈 연산의 대수적 표현

- 두 2-D Tensor의 행렬 곱셈은 2-D Tensor  $D$ 의  $i$ 번째 행벡터와 2-D Tensor  $E$ 의  $j$ 번째 열벡터 사이의 내적을 성분으로 가지는 Tensor
- 예를 들어, 다음과 같이 2-D Tensor  $D$ 와 2-D Tensor  $E$ 의 곱을 표현할 수 있음

$$D = \begin{bmatrix} 1 & 1 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \quad E = \begin{bmatrix} 1 & 0 \\ 1 & -1 \\ 2 & 1 \end{bmatrix} \quad \rightarrow \quad D \times E = \begin{bmatrix} 1 & 1 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \times \begin{bmatrix} 1 & 0 \\ 1 & -1 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} 8 \end{bmatrix}$$

$3 \times 3 \qquad \qquad \qquad 3 \times 2 \qquad \qquad \qquad 3 \times 3 \qquad \qquad \qquad 3 \times 2$

## 3.1 2-D Tensor의 행렬 곱셈 연산

### 3. 2-D Tensor의 행렬 곱셈 연산

#### 2-D Tensor의 행렬 곱셈 연산의 대수적 표현

- 두 2-D Tensor의 곱셈은 2-D Tensor  $D$ 의  $i$ 번째 행벡터와 2-D Tensor  $E$ 의  $j$ 번째 열벡터 사이의 내적을 성분으로 가지는 Tensor
- 예를 들어, 다음과 같이 2-D Tensor  $D$ 와 2-D Tensor  $E$ 의 곱을 표현할 수 있음

$$D = \begin{bmatrix} 1 & 1 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \quad E = \begin{bmatrix} 1 & 0 \\ 1 & -1 \\ 2 & 1 \end{bmatrix} \quad \longrightarrow \quad D \times E = \begin{bmatrix} 1 & 1 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \times \begin{bmatrix} 1 & 0 \\ 1 & -1 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} 8 & 2 \end{bmatrix}$$

$3 \times 3 \qquad \qquad \qquad 3 \times 2 \qquad \qquad \qquad 3 \times 3 \qquad \qquad \qquad 3 \times 2$

## 3.1 2-D Tensor의 행렬 곱셈 연산

### 3. 2-D Tensor의 행렬 곱셈 연산

#### 2-D Tensor의 행렬 곱셈 연산의 대수적 표현

- 두 2-D Tensor의 곱셈은 2-D Tensor  $D$ 의  $i$ 번째 행벡터와 2-D Tensor  $E$ 의  $j$ 번째 열벡터 사이의 내적을 성분으로 가지는 Tensor
- 예를 들어, 다음과 같이 2-D Tensor  $D$ 와 2-D Tensor  $E$ 의 곱을 표현할 수 있음

$$D = \begin{bmatrix} 1 & 1 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \quad E = \begin{bmatrix} 1 & 0 \\ 1 & -1 \\ 2 & 1 \end{bmatrix} \quad \rightarrow \quad D \times E = \begin{bmatrix} 1 & 1 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \times \begin{bmatrix} 1 & 0 \\ 1 & -1 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} 8 & 2 \\ 21 & \end{bmatrix}$$

$3 \times 3 \qquad \qquad \qquad 3 \times 2 \qquad \qquad \qquad 3 \times 3 \qquad \qquad \qquad 3 \times 2$

## 3.1 2-D Tensor의 행렬 곱셈 연산

### 3. 2-D Tensor의 행렬 곱셈 연산

#### 2-D Tensor의 행렬 곱셈 연산의 대수적 표현

- 두 2-D Tensor의 곱셈은 2-D Tensor  $D$ 의  $i$ 번째 행벡터와 2-D Tensor  $E$ 의  $j$ 번째 열벡터 사이의 내적을 성분으로 가지는 Tensor
- 예를 들어, 다음과 같이 2-D Tensor  $D$ 와 2-D Tensor  $E$ 의 곱을 표현할 수 있음

$$D = \begin{bmatrix} 1 & 1 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \quad E = \begin{bmatrix} 1 & 0 \\ 1 & -1 \\ 2 & 1 \end{bmatrix} \quad \longrightarrow \quad D \times E = \begin{bmatrix} 1 & 1 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \times \begin{bmatrix} 1 & 0 \\ 1 & -1 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} 8 & 2 \\ 21 & 1 \end{bmatrix}$$

$3 \times 3 \qquad\qquad\qquad 3 \times 2 \qquad\qquad\qquad 3 \times 3 \qquad\qquad\qquad 3 \times 2$

## 3.1 2-D Tensor의 행렬 곱셈 연산

### 3. 2-D Tensor의 행렬 곱셈 연산

#### 2-D Tensor의 행렬 곱셈 연산의 대수적 표현

- 두 2-D Tensor의 곱셈은 2-D Tensor  $D$ 의  $i$ 번째 행벡터와 2-D Tensor  $E$ 의  $j$ 번째 열벡터 사이의 내적을 성분으로 가지는 Tensor
- 예를 들어, 다음과 같이 2-D Tensor  $D$ 와 2-D Tensor  $E$ 의 곱을 표현할 수 있음

$$D = \begin{bmatrix} 1 & 1 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \quad E = \begin{bmatrix} 1 & 0 \\ 1 & -1 \\ 2 & 1 \end{bmatrix} \quad \longrightarrow \quad D \times E = \begin{bmatrix} 1 & 1 & 3 \\ 4 & 5 & 6 \\ \boxed{7 & 8 & 9} \end{bmatrix} \times \begin{bmatrix} 1 & 0 \\ 1 & -1 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} 8 & 2 \\ 21 & 1 \\ \boxed{33} \end{bmatrix}$$

$3 \times 3 \qquad \qquad \qquad 3 \times 2 \qquad \qquad \qquad 3 \times 3 \qquad \qquad \qquad 3 \times 2$

## 3.1 2-D Tensor의 행렬 곱셈 연산

### 3. 2-D Tensor의 행렬 곱셈 연산

#### 2-D Tensor의 행렬 곱셈 연산의 대수적 표현

- 두 2-D Tensor의 곱셈은 2-D Tensor  $D$ 의  $i$ 번째 행벡터와 2-D Tensor  $E$ 의  $j$ 번째 열벡터 사이의 내적을 성분으로 가지는 Tensor
- 예를 들어, 다음과 같이 2-D Tensor  $D$ 와 2-D Tensor  $E$ 의 곱을 표현할 수 있음

$$D = \begin{bmatrix} 1 & 1 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \quad E = \begin{bmatrix} 1 & 0 \\ 1 & -1 \\ 2 & 1 \end{bmatrix} \quad \longrightarrow \quad D \times E = \begin{bmatrix} 1 & 1 & 3 \\ 4 & 5 & 6 \\ \boxed{7 & 8 & 9} \end{bmatrix} \times \begin{bmatrix} 1 & 0 \\ 1 & -1 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} 8 & 2 \\ 21 & 1 \\ 33 & 1 \end{bmatrix}$$

$3 \times 3 \qquad \qquad \qquad 3 \times 2 \qquad \qquad \qquad 3 \times 3 \qquad \qquad \qquad 3 \times 2 \qquad \qquad \qquad 3 \times 2$

### 2-D Tensor의 행렬 곱셈 연산의 코드 표현

- 두 2-D Tensor의 D, E의 행렬 곱셈 연산의 코드 표현은 다음과 같음
- $D = \text{torch.tensor}([[1, 1, 3], [4, 5, 6], [7, 8, 9]])$ ,  $E = \text{torch.tensor}([[1, 0], [1, -1], [2, 1]])$  2-D Tensor를 생성했을 때,
  - Tensor D와 Tensor E의 행렬 곱셈 연산 [코드 표현](#): `D.matmul(E)`
  - Tensor D와 Tensor E의 행렬 곱셈 연산 [코드 표현](#): `D.mm(E)`
  - Tensor D와 Tensor E의 행렬 곱셈 연산 [코드 표현](#): `D @ E`

### 흑백 이미지의 대칭 이동

- 흑백 이미지 처리에서 행렬의 곱셈 연산을 사용하면 대칭 이동을 수행할 수 있음
  - 대칭 이동(반사)이란 주어진 축을 기준으로 이미지를 뒤집는 변환을 의미함

#### 흑백 이미지의 좌우로 대칭 이동의 대수적 표현

- 흑백 이미지 처리에서 행렬의 곱셈 연산을 사용하면 대칭 이동을 수행할 수 있음
  - 대칭 이동(반사)이란 주어진 축을 기준으로 이미지를 뒤집는 변환을 의미함
- 흑백 이미지의 좌우로 대칭 이동은 y축을 기준으로 이미지를 뒤집는 변환을 의미함

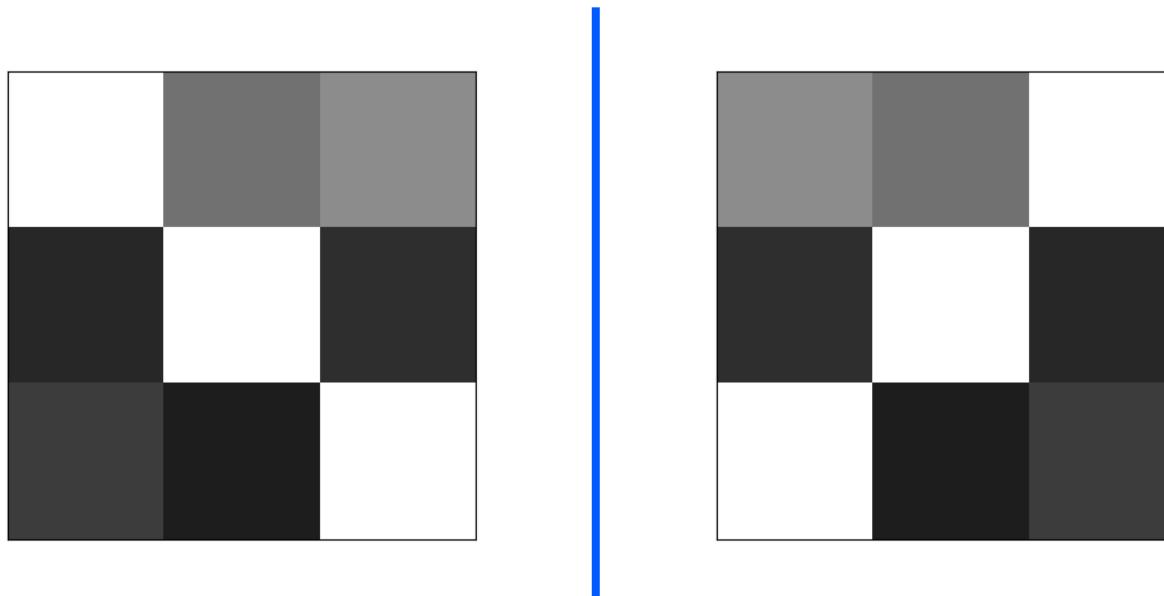
$$I = G \times H = \begin{bmatrix} 255 & 114 & 140 \\ 39 & 255 & 46 \\ 61 & 29 & 255 \end{bmatrix} \times \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 140 & 114 & 255 \\ 46 & 255 & 39 \\ 255 & 29 & 61 \end{bmatrix}$$

## 3.2 2-D Tensor의 행렬 곱셈 연산 활용

### 3. 2-D Tensor의 행렬 곱셈 연산

#### 흑백 이미지의 좌우로 대칭 이동의 이미지 표현

- 흑백 이미지 처리에서 행렬의 곱셈 연산을 사용하면 대칭 이동을 수행할 수 있음
  - 대칭 이동(반사)이란 주어진 축을 기준으로 이미지를 뒤집는 변환을 의미함
- 흑백 이미지의 좌우로 대칭 이동은 y축을 기준으로 이미지를 뒤집는 변환을 의미함



#### 흑백 이미지의 좌우로 대칭 이동의 코드 표현

- 두 2-D Tensor의 G, H
- $G = \text{torch.tensor}([[255, 114, 140], [39, 255, 46], [61, 29, 255]])$ ,  $H = \text{torch.tensor}([[0, 0, 1], [0, 1, 0], [1, 0, 0]])$  2-D Tensor를 생성했을 때,
  - Tensor G의 흑백 이미지를 좌우로 대칭 이동시키는 [코드 표현](#):  $I = G @ H$

### 흑백 이미지의 상하로 대칭 이동

- 흑백 이미지 처리에서 행렬의 곱셈 연산을 사용하면 대칭 이동을 수행할 수 있음
  - 대칭 이동(반사)이란 주어진 축을 기준으로 이미지를 뒤집는 변환을 의미함
- 그렇다면 흑백 이미지의 상하로 대칭 이동은 어떤 축을 기준으로 이미지를 뒤집는 변환일까요?
- 또한, 흑백 이미지를 상하로 대칭 이동시키기 위해서는 어떤 행렬을 어떻게 곱셈해야 할까요?

## 2-D Tensor의 행렬 곱셈 연산

- 2-D Tensor(=Matrix)의 행렬 곱셈은 두 행렬을 결합하여 새로운 행렬을 생성하는 연산이다.
- 흑백 이미지 처리에서 행렬의 곱셈 연산을 사용하면 대칭 이동을 수행할 수 있다.