2. 선형변환

1. 선형 변환(Linear Transformation)

   벡터 공간에서 벡터를 다른 벡터로 변환하는 함수입니다. 선형 변환의 중요한 특징은 두 가지입니다

   1. 덧셈에 대한 보존 (Additivity)

      덧셈에 대한 보존은 다음과 같은 의미를 가집니다.

      $$T(\mathbf{u}+\mathbf{v}) = T(\mathbf{u}) + T(\mathbf{v})$$

      여기서 T는 선형 변환을 의미하고, u와 v는 벡터입니다.

      벡터 u와 v의 합을 먼저 계산한 다음에 그 결과에 대해 선형 변환 T를 적용하는 것과, u와 v 각각에 먼저 선형 변환을 적용한 다음 그 결과를 더하는 것이 동일하다는 뜻입니다. 다시 말해, 선형 변환은 벡터 덧셈을 보존한다는 것을 의미합니다.

      예시

      - 선형 변환을 행렬 A로 생각하고, 두 벡터 $u = [1, 2], v = [3, 4]$가 있다고 합시다.

      1. 먼저 u + v를 계산하면

         $$u + v = [1+3, 2+4] = [4, 6]$$

      2. 행렬 $A = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$를 적용하여 선형 변환을 수행하면

         $$T(u+v) = A[4,6] = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix} \begin{pmatrix} 4 \\ 6 \end{pmatrix} = [8, 12]$$

      3. u와 v 각각에 선형 변환을 적용한 후 더하면

         $$T(u) = A[1,2] = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \end{pmatrix} = [2, 4]$$
         $$T(v) = A[3,4] = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix} \begin{pmatrix} 3 \\ 4 \end{pmatrix} = [6, 8]$$
         $$T(u) + T(v) = [2,4] + [6,8] = [8, 12]$$
         $$\text{즉}, T(u+v) = T(u) + T(v)\text{가성립하는것을볼수있음}$$

2. 스칼라 곱에 대한 보존 (Scalar Multiplication Preservation)

   스칼라 곱에 대한 보존은 다음과 같은 의미를 가집니다

$$T(cv) = cT(v)$$

   여기서, c는 스칼라(숫자), v는 벡터, 그리고 T는 선형 변환 입니다.

   벡터 v에 스칼라 c를 먼저 곱한 후에 선형 변환 T를 적용하는 것과, 벡터 v에 먼저 선형변환 T를 적용한 후 그 결과에 스칼라 c를 곱하는 것이 동일하다는 뜻입니다. 즉, 선형 변환은 스칼라 곱도 보존한다는 의미입니다.

   예시:

   벡터 $v = [1, 2]$와 스칼라 $c = 3$, 그리고 행렬 $A = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$로 선형 변환을 정의하겠습니다.

   1. 먼저 스칼라 c와 벡터 v를 곱하면

$$cv = 3x[1, 2] = [3, 6]$$

   2. 변환 T를 적용하면

$$T(cv) = A[3, 6] = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix} \begin{pmatrix} 3 \\ 6 \end{pmatrix} = [6, 12]$$

   3. 벡터 v에 먼저 변환을 적용한 후 스칼라를 곱하면

$$T(v) = A[1, 2] = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \end{pmatrix} = [2, 4]$$
$$cT(v) = 3 * [2, 4] = [6, 12]$$

   즉, $T(cv) = cT(v)$가 성립하는 것을 알 수 있습니다.

선형 변환은 보통 행렬로 표현됩니다. 즉, 어떤 행렬 A를 이용해 벡터 v를 변환하면, 새로운 벡터 w가 나오는 것을 의미합니다.

$$T(v) = Av$$

이때, A는 행렬이고, v는 벡터입니다.

2D 선형 변환 예시

- 2×2 행렬을 사용하여 2D벡터 변환

```
import numpy as np
import matplotlib.pyplot as plt

# 원래 벡터
v = np.array([2, 1])

# 선형 변환 행렬 (90도 회전)
A = np.array([[0, -1],
              [1,  0]])

# 변환된 벡터
v_transformed = A @ v

# 벡터 시각화
def plot_vectors(vectors, colors=['r', 'b'], scale=4):
    plt.axvline(x=0, color='grey', zorder=0)
    plt.axhline(y=0, color='grey', zorder=0)
    for i, v in enumerate(vectors):
        plt.quiver(0, 0, v[0], v[1], angles='xy', scale_un

    plt.xlim(-5, 5)
    plt.ylim(-5, 5)
    plt.grid(True)
    plt.show()

# 원래 벡터와 변환된 벡터 시각화
plot_vectors([v, v_transformed], colors=['r', 'b'])
```

설명

- $v = [2, 1] : 2D 벡터$
- $A = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} : 90도 회전하는 행렬$
- 행렬 A를 벡터 v에 곱하여 변환된 벡터 $v_{transformed}$를 얻습니다.
- plot_vectors 함수는 원래 벡터와 변환된 벡터를 시각적으로 표현합니다.

선형 변환과 딥러닝

딥러닝에서 선형 변환은 주로 각 레이어의 입력을 출력으로 변환하는 과정에서 사용됩니다. 예를 들어, 신경망에서 입력값에 가중치 행렬을 곱해 새로운 값으로 변환하는 과정이 선형 변환입니다. 이 때 비선형 활성화 함수가 추가되면 비선형성을 가진 신경망을 만들 수 있습니다.

딥러닝에서의 선형 변환은 다음과 같은 형식으로 이루어 집니다.

$$y = W_x + b$$

여기서

- W는 가중치 행렬
- x는 입력 벡터
- b는 편향 벡터

선형변환 후에 ReLU나 시그모이드 같은 비선형 활성화 함수가 적용됩니다.

```
import torch

# 입력 벡터
x = torch.tensor([1.0, 2.0])

# 가중치 행렬 (2x2)
W = torch.tensor([[0.5, 0.2],
                  [0.8, 0.3]])

# 편향 벡터
b = torch.tensor([0.1, 0.2])

# 선형 변환
y = W @ x + b

print(y)  # 출력된 벡터
```

이 코드는 간단한 딥러닝 레이어의 선형 변환을 보여줍니다. W와 b가 각각 가중치와 편향 벡터로, x는 입력 벡터입니다.