머신러닝(딥러닝)을 공부하는대 수학이필요한이유
머신러닝 이론을 반복적으로 데이터를 학습하여 어떤지능 적인 결과를 얻기위해
발전된것이므로 컴퓨터공학,알고리즘,수학등이 교차하는분야
머신러닝 및 딥러닝 알고리즘의 내부작동을 잘파악하고 좋은 결과를
얻을려면 이러한 기술 중 많은 부분을 수학적으로 이해하는 것이 필요
선형대수와머신러닝
선형대수는 머신러닝에대한 심층적이해를위한 전제조건으로서
보편적으로 동의되는 수학분야
선형대수는 많은 난해한 이론과 결과가 있는 큰분야지만,다양한 표기법과 도구는 머신러닝
실무자에게 유용함
선형대수는 데이터의수학으로 통계의이해를 필요로하고,푸리애시리즈 및 컴퓨터그래픽과 같은
많은 실용적인 수학도구의 기초임
(1) 백터 Norms
백터 는 하나또는 그이상의 스칼라 값으로 구성된 튜플

$$v = (a_1, a_2, a_3) = [a_1, a_2, a_3] = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

백터의 내적
u · v= | u | | v | cosθ
u=(1,2,3),v=(4,5,6)
u · v=(1×4)+(2×5)+(3×6)=4+10+18=32
백터의 외적
uxv:방향은 u,v에 동시에 수직이며 크기는 | u×v | = | u | | v | sinθ 인 백터
코드

```
# create a vector
import numpy as np
# define vector
v = np.array([1, 2, 3])
print(v)
```

A = (Ax, Ay, Az), B = (Bx, By, Bz)에 대한 외적은 다음과 같이 정의됩니다

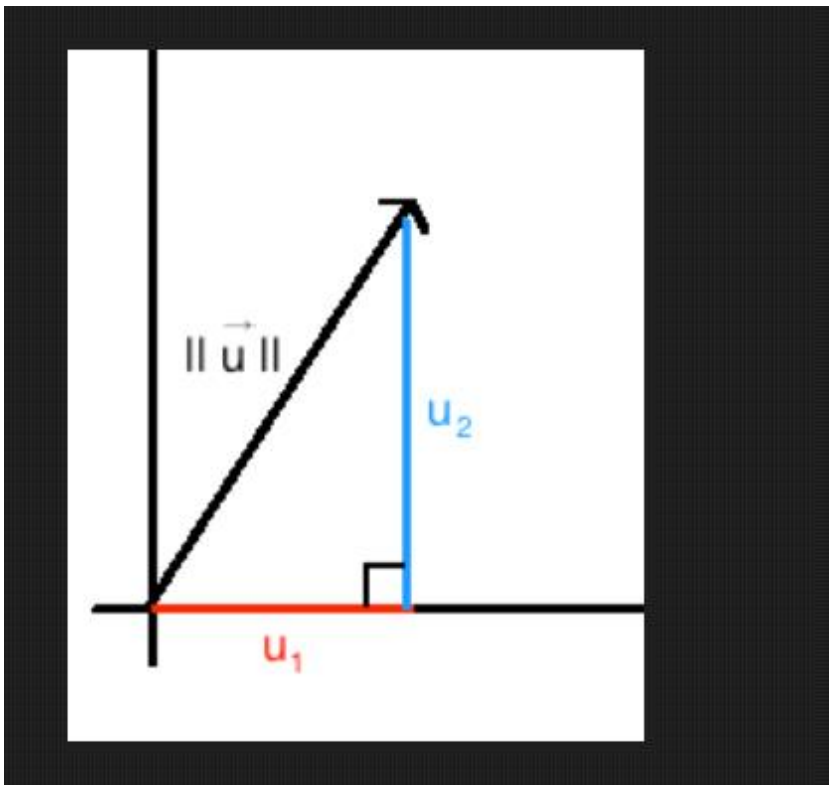
$$\mathbf{A} \times \mathbf{B} = \left(A_y B_z - A_z B_y\right)\mathbf{i} - \left(A_x B_z - A_z B_x\right)\mathbf{j} + \left(A_x B_y - A_y B_x\right)\mathbf{k}$$

i 성분:
(2×6−3×5)=12
−15
=−3(2×6−3×5)=12−15=−3
j
j 성분:
(1×6−3×4)=6−12
=−6(1×6−3×4)=6−12=−6 → 부호 변경:
+6+6
k
k 성분:
(1×5−2×4)=5−8=−3(1×5−2×4)=5−8=−3
u×v=(−3,6,−3)

외적 벡터의 크기는:

$$\left|(-3,6,-3)\right| = \sqrt{(-3)^2 + 6^2 + (-3)^2} = \sqrt{9+36+9} = \sqrt{54} = 3\sqrt{6}$$

백터 Norms
백터의 길이또는 크기를 계산하는 방법
머신러닝의 정규화한 방법이나 백터또는 행렬연산에서 사용될수있음
norm 이측정한 백터의 크기:원점에서 백터좌표까지의 거리,magnitude



Lp Norm

$$L_p(v) = (\sum_i^n |a_i|^p)^{\frac{1}{p}}$$

p는 Norm의차수 :p=1이면 L1 Norm 이고 p=2이면 L2Norm 이다.
n은대상 백터의수

L1 Norm (맨해튼 거리, 택시 거리)
L1 노름($L1$norm)은 벡터의 각 성분의 절댓값을 모두 더한 값입니다.
이는 맨해튼 거리(Manhattan distance) 또는 택시 거리라고도 불립니다.

$$L_1(v) = (\sum_i^n |a_i|) = ||v||_1$$

$$L_1(v) = (\sum_i^3 |a_i|) = |a_1| + |a_2| + |a_3|$$

- L_1 Norm은 L1 정규화(regularization), 컴퓨터 비전 등에서 사용

예제 1: 3차원 벡터
벡터
v =(3,−4,5)v L1 노름을 계산해 보겠습니다.
‖v||1 = | 3 | + | −4 | + | 5 | =3+4+5=12

예제 2: 2차원 벡터
벡터
u=(−2,7)u의 L1 노름을 계산하면:
‖ u ‖ 1= | −2 | + | 7 | =2+7=9

L2 Norm (유클리드 거리, Euclidean Norm)
L2 노름($L2$norm)은 벡터의 크기를 측정하는 가장 일반적인 방법으로, 유클리드 거리(Euclidean distance)라고도 불립니다.
이는 원점에서 벡터까지의 직선 거리(피타고라스 정리 기반)를 의미합니다.

$$L_2(v) = \sqrt{(\sum_i^n |a_i|^2)} = ||v||_2$$

$$L_2(v) = \sqrt{(\sum_i^3 |a_i|^2)} = \sqrt{a_1^2 + a_2^2 + a_3^2}$$

- L_2 Norm은 L2 정규화(regularization), kNN 알고리즘, kmeans 알고리즘 등에서 사용

벡터
v=(3,−4,5)v의 L2 노름을 계산하면:

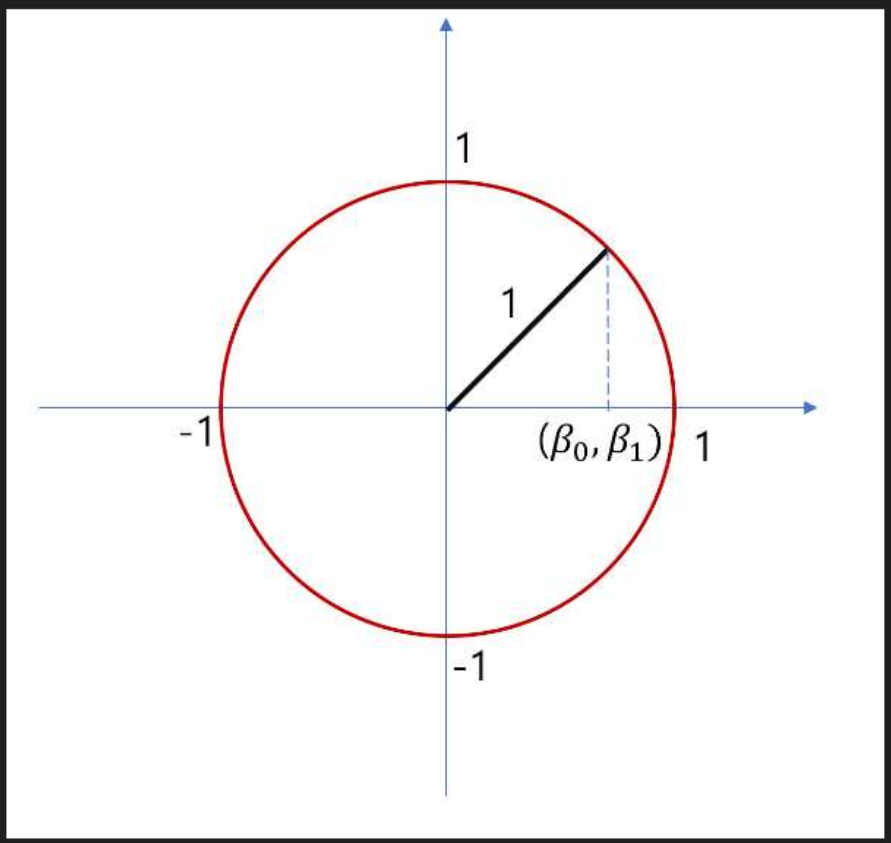
$$\|v\|_2 = \sqrt{3^2 + (-4)^2 + 5^2}$$

$$= \sqrt{9 + 16 + 25} = \sqrt{50} = 5\sqrt{2} \approx 7.07$$

좌표 공간에서 L1과 L2 Norm 시각화

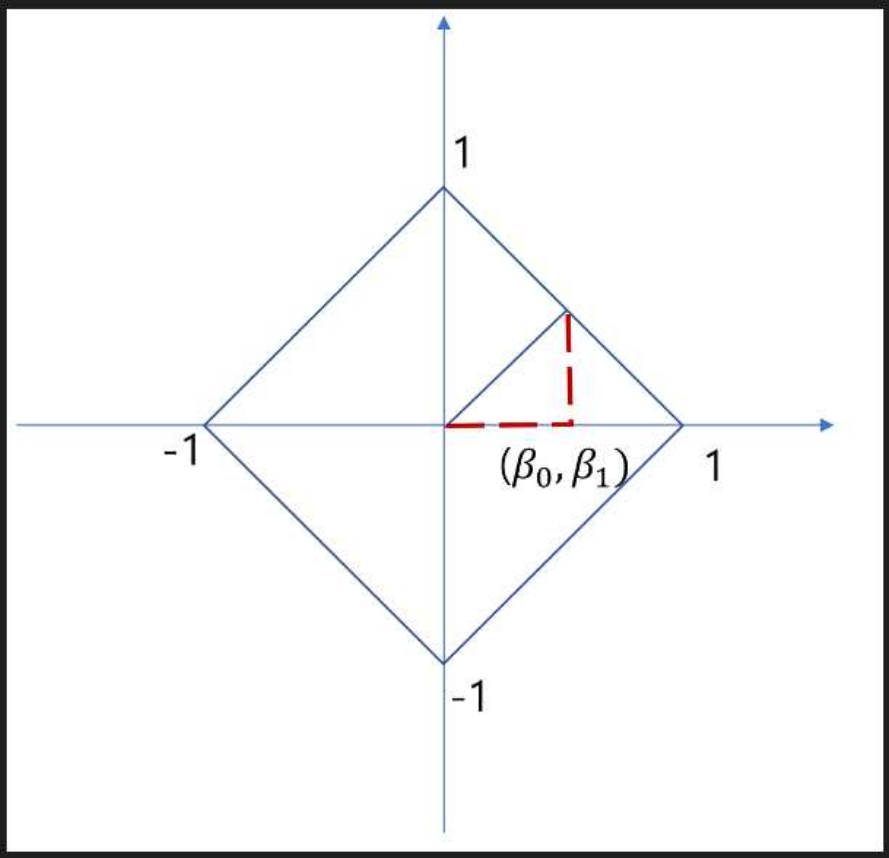
벡터 $B = (\beta_0, \beta_1)$가 있을 때 $L_2(B) = 1$이면,

$||B||_2 = 1 = \sqrt{\beta_0^2 + \beta_1^2}$

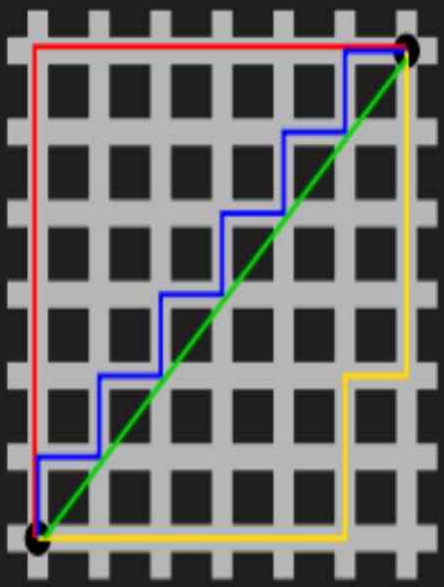


만약 $L_1(B) = 1$이면,

$||B||_1 = 1 = |\beta_0| + |\beta_1|$



L1과 L2 Norm의 직관적 차이 (벡터와 벡터(원점) 사이의 거리 측면)



- 위 그림에서 두 개의 검은 점(벡터)를 잇는 여러 선들이 존재 → 벡터 사이의 거리를 재는 서로 다른 Norm을 표기
- 초록색 선: Euclidean distance = L2 Norm → 단 하나의 경우만 존재.
- 빨간, 파란, 노란 선은 서로 다른 경로지만 모두 같은 L1 Norm임 → Taxicab geometry

행렬
일반적으로 하나이상의 스칼라 행과 열을 가지는 2차원배열을 가르킴
선형대수의 기본요소로 머신러닝 분야 전반에서 사용됨
알고리즘을 훈련할 때 입력데이터 변수(x)와 같은 알고리즘 프로세스 및기술에 사용
행렬의 표기법은 A와 같은 대문자이며 aij와 같은 (i행) 및 j열의 2처원첨자로 참조
백터자체는 하나의 열과 여러행이 있는 행렬로 간주

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \\ a_{3,1} & a_{3,2} \end{pmatrix} = \begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \\ a_{3,1} & a_{3,2} \end{bmatrix}$$

행렬 곱 (Hadamard Product)

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \\ a_{3,1} & a_{3,2} \end{bmatrix}, B = \begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \\ b_{3,1} & b_{3,2} \end{bmatrix}$$

$$C = A \circ B = \begin{bmatrix} a_{1,1} \times b_{1,1} & a_{1,2} \times b_{1,2} \\ a_{2,1} \times b_{2,1} & a_{2,2} \times b_{2,2} \\ a_{3,1} \times b_{3,1} & a_{3,2} \times b_{3,2} \end{bmatrix}$$

행렬 내적 (Dot product)

- $C = A \cdot B = AB$
- 첫 번째 행렬(A)의 열 수는 두 번째 행렬(B)의 행 수와 같아야 함

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \\ a_{3,1} & a_{3,2} \end{bmatrix}, B = \begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{bmatrix}$$

$$C = A \cdot B = AB = \begin{bmatrix} a_{1,1} \times b_{1,1} + a_{1,2} \times b_{2,1} & a_{1,1} \times b_{1,2} + a_{1,2} \times b_{2,2} \\ a_{2,1} \times b_{1,1} + a_{2,2} \times b_{2,1} & a_{2,1} \times b_{1,2} + a_{2,2} \times b_{2,2} \\ a_{3,1} \times b_{1,1} + a_{3,2} \times b_{2,1} & a_{3,1} \times b_{1,2} + a_{3,2} \times b_{2,2} \end{bmatrix}$$

행렬-벡터 간 내적

- $c = A \cdot v = Av$
- 벡터를 행렬로 보면 행렬 간 내적과 동일

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \\ a_{3,1} & a_{3,2} \end{bmatrix}, v = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

$$c = A \cdot v = Av = \begin{bmatrix} a_{1,1} \times v_1 + a_{1,2} \times v_2 \\ a_{2,1} \times v_1 + a_{2,2} \times v_2 \\ a_{3,1} \times v_1 + a_{3,2} \times v_2 \end{bmatrix}$$

행렬 연산

전치 (transpose)

- 전치행렬: 행과 열의 수가 뒤집힌 행렬

$$C = A^T$$

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}, A^T = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}$$

- 행렬이 대칭(symmetrix)일 경우 전치행렬은 원행렬과 동일

$$A = \begin{bmatrix} 3 & -2 & 4 \\ -2 & 6 & 2 \\ 4 & 2 & 3 \end{bmatrix}, A^T = \begin{bmatrix} 3 & -2 & 4 \\ -2 & 6 & 2 \\ 4 & 2 & 3 \end{bmatrix}$$

역 (inverse)

- 행렬의 역은 행렬을 곱하여 단위행렬(unit matrix)이 되는 다른 행렬을 찾는 과정

$AB = BA = I_n$ (n: 행(열)의 크기)

- 행렬의 역은 위 첨자 -1로 표시

$B = A^{-1}$

- 모든 행렬이 역행렬을 구할 수 있는 것은 아님
- 역을 구할 수 없는 정방행렬(정사각형 모양 행렬)을 특이행렬(singluar matrix)이라 함
- 역행렬은 일반적으로 직접 계산할 수 없으며, 행렬 분해(decomposition) 형태의 방법이 가능할 때, 수치 연산을 통해 발견됨

대각합 (trace)

- 행렬의 주 대각선 값의 합
- 정방행렬에서 대각합을 계산하는 연산: $tr(\cdot)$

$$tr(A) = a_{1,1} + a_{2,2} + a_{3,3}$$

희소행렬 (Sparce Matrix)

- 대부분 0 값으로 구성된 행렬
- 대부분 0이 아닌 값으로 구성된 행렬: 조밀 행렬(dense matrix)
- 희소도(sparcity) = (0인 원소의 수) / (전체 원소 수)

$$A = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 1 \\ 0 & 0 & 0 & 2 & 0 & 0 \end{bmatrix}$$

A의 희소도 = 13/18 = 72.2%

- 매우 큰 행렬이 희소행렬일 경우 저장을 위한 메모리 자원 낭비 및 계산 낭비가 심함

희소행렬의 활용

- 특정 유형의 데이터, 특히 활동의 발생 또는 횟수를 기록하는 관측치에서 많이 나타남
  - 사용자가 영화 카탈로그의 영화를 시청했는지 여부
  - 사용자가 제품 카탈로그에서 제품을 구매했는지 여부
  - 노래 카탈로그에서 노래를 청취 한 횟수

텐서(tensor)

- 벡터와 행렬의 일반화 $\rightarrow$ 다차원배열
- 축(axis) 수가 가변적인 일반 그리드에 배열된 숫자 배열
- 행렬의 성질을 공유함: 합, 차, 곱(Hadamard Product), 몫

텐서 곱(tensor product)

- q 차원의 텐서 A와 r 차원의 텐서 B가 주어질 때 텐서 곱은 $q + r$ 차원의 새로운 텐서를 만듦
- 표기법: $\otimes$

$$a = \begin{pmatrix} a_1 \\ a_2 \end{pmatrix}$$

$$b = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$

$$C = a \otimes b = \begin{pmatrix} a_1 \times \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \\ a_2 \times \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} a_1 \times b_1 & a_1 \times b_2 \\ a_2 \times b_1 & a_2 \times b_2 \end{pmatrix}$$

(3) 행렬 분해(Matrix Decomposition, Matrix Factorization)

- 보다 복잡한 행렬 연산을 더 쉽게 계산할 수 있도록 행렬을 구성 부분으로 줄이는 방법
- 선형 방정식 시스템 해 구하기, 역 계산 및 행렬의 행렬식 계산 등의 기본 작업에 사용
- 예)

$$\begin{bmatrix} 2 & 6 & 2 \\ -3 & -8 & 0 \\ 4 & 9 & 2 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \\ 3 \end{bmatrix}$$

만약 다음과 같이 분해 가능하다면(LU분해)

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

식은,

$$\begin{bmatrix} 2 & 0 & 0 \\ -3 & 1 & 0 \\ 4 & -3 & 7 \end{bmatrix} \cdot \begin{bmatrix} 1 & 3 & 1 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \\ 3 \end{bmatrix}$$

식의 중간 부분을 다음과 같이 정의하면,

$$\begin{bmatrix} 1 & 3 & 1 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

전체 식은,

$$\begin{bmatrix} 2 & 0 & 0 \\ -3 & 1 & 0 \\ 4 & -3 & 7 \end{bmatrix} \cdot \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \\ 3 \end{bmatrix}$$

따라서 행렬식을 쉽게 풀 수 있다.

고유값 분해 (Eigendecomposition)

- 정방행렬을 고유 벡터 및 고유 값의 집합으로 분해
- 벡터는 다음 방정식을 충족하는 경우 행렬의 고유 벡터(eigenvector)임
- $A \cdot v = \lambda \cdot v \Rightarrow Av = \lambda v$
  - A: 분해 대상 부모 정방행렬
  - v: 행렬의 고유벡터
  - λ: 고유값(eigenvalue, 스칼라)
- 행렬은 부모 행렬의 각 차원에 대해 하나의 고유 벡터와 고유 값을 가질 수 있음
  - 모든 정방행렬이 고유 벡터와 고유 값으로 분해 될 수있는 것은 아니며 일부는 복소수가 필요한 방식으로만 분해 될 수 있음
- 부모 행렬은 고유 벡터와 고유 값의 곱으로 표시 될 수 있음
- $A = Q \cdot \Lambda \cdot Q^{-1} \Rightarrow A = Q\Lambda Q^{-1}$
  - Q: 고유벡터로 구성된 행렬
  - Λ: 고유값으로 구성된 대각행렬
- 고유값 분해는 머신러닝에서 데이터의 차원을 줄이는 데 사용할 수 있는 주성분 분석 방법(PCA: Principal Component Analysis) 또는 PCA에서 행렬의 주성분을 계산하는 데 사용할 수도 있음

(1) 행렬 A 정의

다음과 같은 행렬 A가 있다고 가정하겠습니다.

$$A = \begin{bmatrix} 4 & -2 \\ 1 & 1 \end{bmatrix}$$

이 행렬의 고유값과 고유벡터를 구해 보겠습니다.

(2) 고유값 찾기

고유값을 찾으려면 **특성 방정식**을 풀어야 합니다.

$$\det(A - \lambda I) = 0$$

행렬 A에서 단위 행렬 I에 고유값 λ을 곱한 행렬을 빼고, 행렬식(Determinant)을 계산합니다.

$$A - \lambda I = \begin{bmatrix} 4-\lambda & -2 \\ 1 & 1-\lambda \end{bmatrix}$$

이 행렬의 행렬식을 계산하면,

$$(4-\lambda)(1-\lambda) - (-2)(1) = 0$$

전개하면,

$$4 - 4\lambda - \lambda + \lambda^2 + 2 = 0$$

$$\lambda^2 - 5\lambda + 6 = 0$$

因式分解(인수분해)하면,

$$(\lambda - 3)(\lambda - 2) = 0$$

즉, 고유값은

$$\lambda_1 = 3, \quad \lambda_2 = 2$$

(3) 고유벡터 찾기

각 고유값에 대해 $A\mathbf{v} = \lambda\mathbf{v}$을 만족하는 고유벡터 $\mathbf{v}$를 찾아야 합니다.

고유값 $\lambda_1 = 3$에 대한 고유벡터

$$(A - 3I)\mathbf{v} = 0$$

$$\begin{bmatrix} 4-3 & -2 \\ 1 & 1-3 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 & -2 \\ 1 & -2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

첫 번째 식:

$$x - 2y = 0 \quad \Rightarrow \quad x = 2y$$

즉, 고유벡터는

$$\mathbf{v}_1 = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

고유값 $\lambda_2 = 2$에 대한 고유벡터

$$(A - 2I)\mathbf{v} = 0$$

$$\begin{bmatrix} 4-2 & -2 \\ 1 & 1-2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 2 & -2 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

첫 번째 식:

$$2x - 2y = 0 \quad \Rightarrow \quad x = y$$

즉, 고유벡터는

$$\mathbf{v}_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

(4) 고유값 분해

고유값과 고유벡터를 사용하여 행렬 A를 분해할 수 있습니다.

- **고유벡터 행렬 P**

$$P = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix}$$

- **대각행렬 D (고유값을 대각성분으로 가지는 행렬)**

$$D = \begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix}$$

이제 고유값 분해 공식을 적용하면,

$$A = PDP^{-1}$$

P^{-1}을 계산하여 확인하면 원래의 행렬 A가 나옵니다.

1. 3×3 행렬에서 고유값 분해 적용

우선, 다음과 같은 3×3 행렬 A를 예제로 사용하겠습니다.

$$A = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}$$

이제 이 행렬을 **고유값 분해**해 보겠습니다.

고유값 찾기

특성 방정식을 계산하기 위해 행렬식(Determinant)을 구합니다.

$$\begin{vmatrix} 2-\lambda & -1 & 0 \\ -1 & 2-\lambda & -1 \\ 0 & -1 & 2-\lambda \end{vmatrix} = 0$$

행렬식을 전개하면,

$$(2-\lambda)\begin{vmatrix} 2-\lambda & -1 \\ -1 & 2-\lambda \end{vmatrix} - (-1)\begin{vmatrix} -1 & -1 \\ 0 & 2-\lambda \end{vmatrix} = 0$$

$$(2-\lambda)\left((2-\lambda)(2-\lambda) - (-1)(-1)\right) - (-1)\left((-1)(2-\lambda) - (-1)(0)\right) = 0$$

$$(2-\lambda)\left((2-\lambda)^2 - 1\right) + (2-\lambda) = 0$$

$$(2-\lambda)\left(\lambda^2 - 4\lambda + 4 - 1\right) + (2-\lambda) = 0$$

$$(2-\lambda)(\lambda^2 - 4\lambda + 3) = 0$$

인수분해하면,

$$(2-\lambda)(\lambda-3)(\lambda-1) = 0$$

따라서 **고유값은**

$$\lambda_1 = 3, \quad \lambda_2 = 2, \quad \lambda_3 = 1$$

고유벡터 찾기

각 고유값에 대해 $A\mathbf{v} = \lambda\mathbf{v}$을 만족하는 고유벡터를 찾습니다.

고유값 $\lambda_1 = 3$에 대한 고유벡터

$$(A - 3I)\mathbf{v} = 0$$

$$\begin{bmatrix} -1 & -1 & 0 \\ -1 & -1 & -1 \\ 0 & -1 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

이 방정식을 풀면, 하나의 고유벡터는

$$\mathbf{v}_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

고유값 $\lambda_2 = 2$에 대한 고유벡터

$$(A - 2I)\mathbf{v} = 0$$

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & -1 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

이를 풀면, 하나의 고유벡터는

$$\mathbf{v}_2 = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$$

고유값 $\lambda_3 = 1$에 대한 고유벡터

$$(A - I)\mathbf{v} = 0$$

$$\begin{bmatrix} 1 & -1 & 0 \\ -1 & 1 & -1 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

이를 풀면, 하나의 고유벡터는

$$\mathbf{v}_3 = \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}$$

고유값 분해

이제 행렬 A를 **고유값 분해** 형태로 나타낼 수 있습니다.

- **고유벡터 행렬 P**

$$P = \begin{bmatrix} 1 & -1 & 1 \\ 1 & 0 & -2 \\ 1 & 1 & 1 \end{bmatrix}$$

- **대각행렬 D (고유값을 대각성분으로 가지는 행렬)**

$$D = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

따라서,

$$A = PDP^{-1}$$

(4) 행렬과 통계

- 통계는 머신러닝에서 데이터를 더 잘 이해하기 위한 유용한 도구

기대값(Expected Value)과 평균(Mean)

- 확률에서 어떤 확률변수(random variable) X의 평균 값을 기대값(expected value, expectation)이라 함
- 기대값은 $E(X)$로 표기하며 확률 가중 합계로 계산

$$E(X) = \sum x_1 \times p_1, x_2 \times p_2, x_3 \times p_3, \cdots, x_n \times p_n$$

- 각 이벤트의 확률이 비슷할 경우 기대값은 모든 값의 합계를 이벤트의 수로 나누어 계산

$$E(X) = \frac{1}{n} \times \sum x_1, x_2, x_3, \cdots, x_n$$

- 통계에서 산술 평균 또는 표본 평균은 도메인에서 가져온 예제 샘플에서 추정 가능하며 이때 평균은 μ로 표시

$$\mu = \frac{1}{n} \times \sum x_1, x_2, x_3, \cdots, x_n = P(x) \times \sum x$$

(x: 관측값 벡터, $P(x)$ 각 관측값에 대해 계산된 확률)

- x와 같은 특정 변수에 대해 계산할 때 평균은 $\bar{x}$로 표시

$$\bar{x} = \frac{1}{n} \times \sum_{i=1}^{n} x_i$$

분산(Variance)과 표준편차(Standard Deviation)

- 확률에서 랜덤변수 X의 분산은 분포의 값이 평균에 대해 평균적으로 얼마나 달라지는지의 정도: $Var[X]$
- 분포에서 각 값의 평균 제곱 차이로 계산

$$Var[X] = E[(X - E[X])^2]$$

- 변수의 기대값이 계산되었다고 가정하면($E[X]$), 확률변수의 분산은 다음과 같이 계산

$$Var[X] = \sum p(x_1) \times (x_1 - E[X])^2, p(x_2) \times (x_2 - E[X])^2, \cdots, p(x_n) \times (x_n - E[X])^2$$

- 분포에 있는 각 값의 확률이 같으면

$$Var[X] = \frac{1}{n} \times \sum (x_1 - E[X])^2, (x_2 - E[X])^2, \cdots, (x_n - E[X])^2$$

- 통계에서 분산은 도메인에서 가져온 예제 샘플(표본)에서 추정 할 수 있으며 표본 분산은 σ^2로 표기
- 차이 제곱의 합에 표본의 편향(표본은 일부이므로 분산을 과소평가할 수 있음)을 수정하기 위해 예제 수에서 1을 뺀 값의 역수로 곱함

$$\sigma^2 = \frac{1}{n-1} \times \sum_{i=1}^{n} (x_i - \mu)^2$$

공분산(Covariance)과 상관관계(Correlation)

- 확률에서 공분산은 두 확률 변수에 대한 결합 확률의 척도로 두 변수가 함께 어떻게 변하는 지 설명함
- X와 Y가 확률변수일때 공분산은 $cov(X,Y)$로 표기
- 공분산은 각 랜덤 변수와 기대값의 차이 곱의 평균 또는 기대값으로 계산

$$cov(X,Y) = E[(X - E[X]) \times (Y - E[Y])]$$

- X와 Y의 기대값이 계산되었다고 가정하면,

$$cov(X,Y) = \frac{1}{n} \times \sum_{i=1}^{n} (x_i - E[X]) \times (y_i - E[Y])$$

- 통계에서 표본 공분산은 분산과 동일한 편향 교정을 통해 계산 가능

$$cov(X,Y) = \frac{1}{n-1} \times \sum_{i=1}^{n} (x_i - \mu_x) \times (y_i - \mu_y)$$

- 공분산의 부호는 두 변수가 같이 증가하는지 (양수) 또는 서로 다르게 증감하는지 (음수)로 해석 될 수 있음
- 공분산의 크기는 쉽게 해석하기 어려움. 공분산 값이 0이면 두 변수가 완전히 독립적임을 나타냄

상관계수(Correlation Coefficient)

- 공분산을 X와 Y의 표준 편차로 나누어 -1과 1 사이의 정규화된 크기로 해석 가능하도록 만들 수 있음
- Pearson 상관계수(correlation coefficient): r

$$r = \frac{cov(X,Y)}{s_Y \times s_Y}$$

공분산 행렬(Covariance Matrix)

- 둘 이상의 랜덤 변수 간의 공분산을 설명하는 정방 대칭 행렬
- 공분산 행렬의 대각선은 분산-공분산 행렬이라고 불리는 각 랜덤 변수의 분산
- 두 변수의 공분산을 일반화 한 것으로 데이터 세트의 모든 변수가 함께 변경 될 수있는 방식을 포착

$ \Sigma = E[(X-E[X]\times(Y-E[Y])] $, 이때 $\sum_{i,j} = cov(X_i, X_j) $

- 공분산 행렬은 랜덤 변수 행렬에서 구조화 된 관계를 분리하는 데 유용

(5) 주성분 분석 (PCA: Principle Component Analysis)

- 데이터의 차원을 줄이는 방법
- m열(특징)이 있는 데이터를 m개 이하의 열이 있는 부분 공간에 투영하면서 원본 데이터의 본질을 유지하는 프로젝션 방법

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \\ a_{3,1} & a_{3,2} \end{bmatrix}$$

$$B = PCA(A)$$

- 단계 1: 각 열의 평균 값 계산

$$M = mean(A)$$

- 단계 2: 평균 열 값을 빼서 각 열 값을 중앙에 배치

$$C = A - M$$

- 단계 3: 중심 행렬 C의 공분산 행렬 계산 - 두 열이 함께 변경되는 양과 방향(양수 또는 음수)의 정규화 된 값

$$V = cov(C)$$

- 단계 4: 공분산 행렬 V의 고유값 분해 계산

$$values, vectors = eig(V)$$

- 고유벡터($vectors$): B의 축소 된 부분 공간에 대한 방향 또는 구성 요소
- 고유값($values$): 방향의 크기
- 고유벡터는 A에 대한 새 부분 공간의 구성 요소 또는 축의 순위를 제공하기 위해 고유값을 기준으로 내림차순으로 정렬 할 수 있음
- 모든 고유값이 비슷한 값을 갖는 경우 기존 표현이 이미 합리적으로 압축되거나 조밀 할 수 있음
- 0에 가까운 고유값은 버릴 수있는 B의 구성 요소 또는 축을 나타냄
- 가장 큰 k개의 고유값을 갖는 k개의 고유벡터(주성분) 선택

$$B = select(values, vectors)$$

- 특이값 분해(SVD)와 같은 다른 행렬 분해 방법도 사용 가능
- 고유값과 고유벡터가 선택되면 데이터는 다음 행렬곱을 통해 부분공간으로 투영(projection)됨

$$P = B^T \cdot A$$

선형 회귀란?

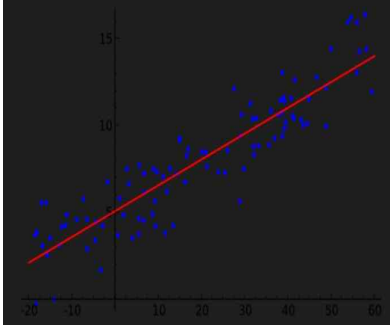
- 입력 변수 x와 출력 변수 y의 두 스칼라 값 간의 관계를 모델링하는 방법
- 모델은 y가 선형 함수 또는 입력 변수의 가중 합이라고 가정

$$y = f(x) = b_0 + b_1 \times x_1$$

- 모델은 다음과 같은 여러 입력 변수가 주어질 때 출력 변수를 모델링하는 데 사용할 수도 있음

$$y = f(x) = b_0 + (b_1 \times x_1) + (b_2 \times x_2) + \cdots$$

- 선형 회귀 모델을 만드는 목적은 출력 변수 y의 예측에서 오류를 최소화하는 값 (b)를 찾는 것임



- 회귀(regression): 옛날 상태로 돌아가는 것
  - 영국의 유전학자 프랜시스 골턴이 부모의 키와 아이들의 키 사이의 연관 관계를 연구하면서 부모와 자녀의 키사이에는 선형적인 관계가 있고 키가 커지거나 작아지는 것보다는 전체 키 평균으로 돌아가려는 경향이 있다는 가설을 세웠으며 이를 분석하는 방법을 "회귀분석"이라고 함

- 방정식 수(y_1, y_2, y_3, y_4)가 미지수(b_1, b_2, b_3)보다 많은 초과 결정(over-determined) 시스템
  - 방정식이 서로 모순되서(inconsistent) 해를 가지지 않는 경우가 많음
- 약간의 오차를 허용하는 해결책 필요: 제곱 오차를 최소화(minimize squared error) $\rightarrow$ 최소제곱법(Least Linear Squares)

$$||X \cdot b - y||^2 = \sum_{i=1}^{m} \sum_{j=1}^{n} (X_{i,j} \cdot b_j - y_i)^2$$

- 정규방정식(normal equation)을 사용한 공식

$$X^T \cdot X \cdot b = X^T \cdot y$$

$$b = (X^T \cdot X)^{-1} \cdot X^T \cdot y$$

$$\hat{y} = X \cdot b$$