Linear Algebra Basic

염은지

Vector

- **Vector**
  - 크기와 방향을 가지고 있는 양을 나타내는 개념
  - 좌표 상에서 화살표로 표현
  - 힘, 속도, 위치와 같은 양은 모두 벡터로 나타낼 수 있음
  - v_1과 v_2는 벡터의 성분으로, 크기와 방향을 결정
  - v_1은 가로로, v_2는 세로로 위치한 값
- **Scalar**
  - 하나의 수치만으로 완전히 표시되는 양
  - 방향이 없음
  - 질량, 에너지, 밀도, 전기량 등
- **Dimension**
  - 벡터의 성분의 개수를 의미
  - 2차원 벡터의 경우 성분이 2개이므로 2차원 벡터

$$\mathbf{v} = \begin{bmatrix} v_1 \; v_2 \end{bmatrix}$$

Vector

- 예시
  - 만약 내가 남쪽으로 5미터, 동쪽으로 3미터를 걸어간다면?
  - 이동한 거리와 방향을 다음과 같이 표현

$$\mathbf{v_{2D}} = [3 \ -5]$$

Vector

- **벡터의 덧셈과 뺄셈**
  - 곱같은 위치에 있는 성분들을 더하거나 빼서 새로운 벡터를 생성
- **벡터의 스칼라 곱**
  - 벡터에 스칼라 를 곱하면 각 성분에 스칼라가 곱해지게 됩

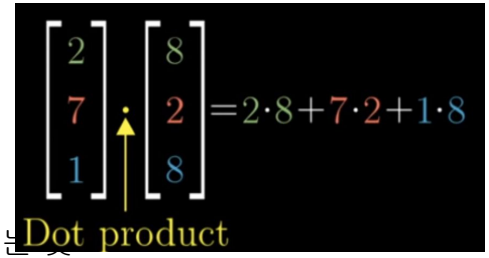
- 벡터 $\mathbf{u}$: $[u_1, u_2, \ldots, u_n]$
- 벡터 $\mathbf{v}$: $[v_1, v_2, \ldots, v_n]$
- 벡터의 덧셈: $\mathbf{u} + \mathbf{v} = [u_1 + v_1, u_2 + v_2, \ldots, u_n + v_n]$
- 벡터의 뺄셈: $\mathbf{u} - \mathbf{v} = [u_1 - v_1, u_2 - v_2, \ldots, u_n - v_n]$
- 벡터의 스칼라 곱: $c \cdot \mathbf{u} = [c \cdot u_1, c \cdot u_2, \ldots, c \cdot u_n]$

Vector

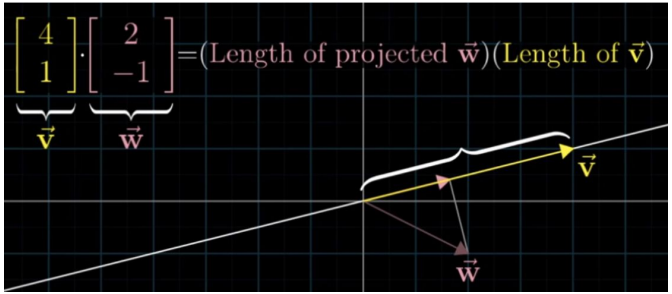
- 벡터 내적: $\mathbf{u} \cdot \mathbf{v} = u_1 \cdot v_1 + u_2 \cdot v_2 + \ldots + u_n \cdot v_n$
- 벡터 외적: $\mathbf{u} \times \mathbf{v} = [u_2v_3 - u_3v_2, u_3v_1 - u_1v_3, u_1v_2 - u_2v_1]$

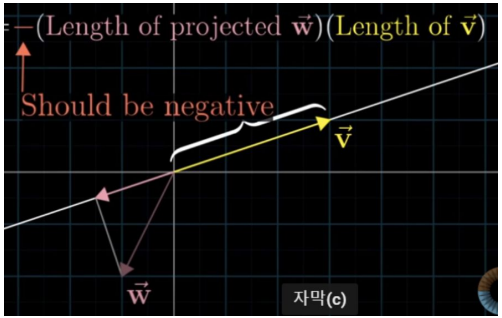
- **내적 (Dot Product)**
  - 벡터의 각 성분을 곱한 후 그 결과를 모두 더하는 연산
  - 두 벡터 사이의 상관관계를 나타냄
    - 유사도 측정
      - 두 벡터가 얼마나 유사한 방향을 가지고 있는지를 나타내는 척도로 사용
      - 두 벡터가 비슷한 방향을 가지고 있을수록 내적 값은 크고, 서로 다른 방향을 가지고 있을수록 내적 값은 작아짐
    - 투영 (Projection)
      - 한 벡터를 다른 벡터 방향으로 투영한 길이를 계산하는 데 활용
      - 벡터를 다른 벡터에 대해 분해하고 관련된 계산에 사용
- **외적 (Cross Product)**
  - 두 벡터의 성분을 활용하여 새로운 벡터를 생성하는 연산
  - 두 벡터 사이의 수직인 벡터를 생성 (면의 법선 벡터 생성)
  - 평면의 면적과 3차원 공간의 부피 계산에 활용
  - 물체의 회전 운동 계산 시 회전 축 계산 시 사용

Vector



- **내적 (Dot Product)을 Projection 관점에서 다시 한번 살펴보자**
  - 2차원 벡터 W를 1차원 공간인 수선(직각이 되도록)의 한 점으로 변환하는 것
  - 이미지에서는 V 벡터를 지나는 직선 위를 투영하는 상황
  - 결국, (벡터 V를 지나는 직선에투영된 W벡터의 길이) X (벡터 V의
  - W벡터의 투사체가 v 벡터 방향과 반대이면 내적 값은 음수
  - 이를 통해 파악학 수 있는 내적의 속성!
    - 같은 방향 : V*W > 0
    - 반대 방향 : V*W < 0
    - 서로 직각 : V*W = 0





Vector

- **단위 벡터 (Unit Vector)**
  - norm이 1인 벡터
  - 임의의 영벡터가 아닌 벡터 v를 선택하여 v에 그의 norm의 역수를 곱하면 단위 벡터를 만들 수 있음 (정규화 과정)
  - **norm?**
    - 벡터의 길이
    - 일반적으로 || v || 또는 | v | 로 나타내며, 벡터 v에 대한 놈(norm), 길이(distance), 크기(magnitude)라고 불름
    - $\vec{v} = (v_1, v_2, \cdots, v_n)$ 일 때, $\|v\| = \sqrt{v_1^2 + v_2^2 + \cdots + v_n^2}$

$$\vec{u} = \frac{1}{\|\vec{v}\|}\vec{v}$$

Matrix

- 숫자들을 격자 형태로 배열한 것으로, 주로 데이터나 변환을 표현하는 데 사용
- 행(가로 방향)과 열(세로 방향)로 구성
- 크기(size) = (행의 크기 x 열의 크기)
- 정사각행렬, 정방행렬(square matrix) = 행의 크기와 열의 크기가 같은 행렬 (N * N)

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

Matrix

$$\mathbf{A}+\mathbf{B}=\begin{bmatrix} a_{11}+b_{11} & a_{12}+b_{12} & \cdots \\ a_{21}+b_{21} & a_{22}+b_{22} & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix}$$

$$\mathbf{A}=\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

$$\mathbf{B}=\begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & \cdots & b_{mn} \end{bmatrix}$$

- **행렬의 합**
  - 행렬의 덧셈과 뺄셈은 각 요소들을 같은 위치끼리 더하거나 빼는
  - 행렬의 크기가 같아야 같은 위치의 element들끼리 연산이 가능
- **행렬의 스칼라 곱**
  - 행렬에 스칼라(숫자)를 곱하면 모든 요소에 그 스칼라 값을 곱한 결과를 얻

$$2\cdot\mathbf{A}=\begin{bmatrix} 2\cdot 2 & 2\cdot 3 \\ 2\cdot 1 & 2\cdot 4 \end{bmatrix}=\begin{bmatrix} 4 & 6 \\ 2 & 8 \end{bmatrix}$$

- **행렬간의 곱**
  - 두 개의 행렬 A와 B의 곱셈은 첫 번째 행렬의 행과 두 번째 행렬의 열을 조합하여 새로운 행렬을 만드는 연산
  - 행렬 A의 열 수와 행렬 B의 행 수가 일치해야 곱셈이 가능
  - 만약 A의 크기가 m * n 인 행렬이고, B의 크기가 n * p 인 행렬이라면, 두 행렬의 곱 C는 크기가 m * p 인 행렬

$$C=AB \qquad \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & \ddots & \cdots & \vdots \\ \vdots & \cdots & \cdots & \vdots \\ a_{m1} & \cdots & \cdots & a_{mn} \end{bmatrix}\begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1r} \\ b_{21} & \ddots & \cdots & \vdots \\ \vdots & \cdots & \cdots & \vdots \\ b_{n1} & \cdots & \cdots & b_{nr} \end{bmatrix}$$

$$c_{ij}=a_{i1}b_{1j}+a_{i2}b_{2j}+\cdots+a_{in}b_{nj}=\sum_{k=1}^{n}a_{ik}b_{kj}$$

$$\underset{(m\times n)}{A}\quad\underset{(n\times r)}{B}\;=\;\underset{(m\times r)}{C}$$

(A의 열의 개수 n = B의 행의 개수 n)

Matrix

- **항등행렬 (Identity Matrix)**
  - 항등원의 개념을 행렬에 적용한 것
  - 행렬의 곱셈에서 곱해도 변화하지 않는 역할을 하는 행렬
  - **항등원?**
    - 항등원은 어떤 연산에 대해 그 연산을 수행해도 값이 변하지 않는
    - 실수에 대한 덧셈 연산의 항등원은 0
    - 수에 대한 곱셈 연산의 항등원은 1

$$I = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}$$

- **역행렬 (Inverse Matrix)**
  - 역원의 개념을 행렬에 적용한 것
  - A와의 곱셈 결과가 항등행렬이 되는 행렬
  - **역원?**
    - 어떤 연산에 대해 해당 원소와 연산을 했을 때 항등원이 되는 원소
    - 실수에 대한 덧셈의 역원은 해당 실수의 음수
    - 실수에 대한 곱셈의 역원은 1을 그 실수로 나눈 값

$$\mathbf{A} \cdot \mathbf{A}^{-1} = \mathbf{A}^{-1} \cdot \mathbf{A} = I$$

- 항등행렬과 역행렬은 선형방정식의 해를 구하는 데에 많이 사용

Matrix

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{bmatrix}$$

- **전치행렬 (Transpose Matrix)**
  - 행렬의 행과 열을 뒤바꾼 행렬
  - 행렬의 곱셈 연산에서 유용하게 사용
  - **전치행렬의 특징**

1. $(A^T)^T = A$: 전치행렬의 전치행렬은 원래 행렬과 같습니다.
2. $(A + B)^T = A^T + B^T$: 두 행렬의 합의 전치는 각각의 전치행렬의 합과 같습니다.
3. $(kA)^T = kA^T$: 상수 k를 곱한 행렬의 전치는 상수를 전치한 행렬에 곱한것과 같습니다.
4. $(AB)^T = B^T A^T$: 두 행렬의 곱의 전치는 각각의 전치행렬을 역순으로 곱한 것과 같습니다.

$$A^T = \begin{bmatrix} a_{11} & a_{21} & \cdots & a_{n1} \\ a_{12} & a_{22} & \cdots & a_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1m} & a_{2m} & \cdots & a_{nm} \end{bmatrix}$$

- **직교 행렬(Orthogonal Matrix)**
  - A와 A의 전치 행렬을 곱했을 때 단위 행렬이 된다면, A를 직교 행렬이라고 함

$$\boldsymbol{A}^\intercal \boldsymbol{A} = \boldsymbol{A}\boldsymbol{A}^\intercal = \begin{bmatrix} \frac{2}{3} & -\frac{2}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{2}{3} & \frac{2}{3} \\ \frac{2}{3} & -\frac{1}{3} & -\frac{2}{3} \end{bmatrix}^\intercal \begin{bmatrix} \frac{2}{3} & -\frac{2}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{2}{3} & \frac{2}{3} \\ \frac{2}{3} & -\frac{1}{3} & -\frac{2}{3} \end{bmatrix} = \boldsymbol{I}_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

Matrix

- **대칭행렬 (Symmetric Matrix)**
  - 전치(transpose)한 행렬과 원래 행렬이 같은 행렬
  - 행렬의 대각선(↘)을 기준으로 대칭되는 원소들이 같은 값을 가지는 행
- **대각행렬 (Diagonal Matrix)**
  - 대각행렬은 대각선 상의 원소 이외의 원소가 모두 0인 행렬
  - 수학적 연산과 변환을 표현할 때 주로 활용
  - 행렬 분해에 활용

$$A = \begin{bmatrix} a & b & c \\ b & d & e \\ c & e & f \end{bmatrix}$$

$$D = \begin{bmatrix} d_1 & 0 & 0 \\ 0 & d_2 & 0 \\ 0 & 0 & d_3 \end{bmatrix}$$

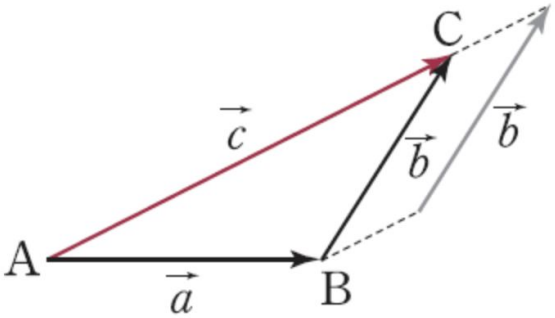
Linear transformation

- **선형관계 (Linear Relationship)**
  - 두 개 이상의 변수 사이에서 일정한 패턴이나 규칙에 따라 형성되는 관계
  - 변수들 간에 직선의 형태로 표현될 수 있음
  - 만약, 두 변수 X와 Y 간에 선형 관계가 있다면, X의 값이 변할 때 Y의 값도 일정한 비율로 변화하게 됨
  - 이 비율은 변화하는 두 변수 사이의 상수 계수로 나타낼 수 있음
  - 예를 들어, $Y = 2X + 3$. 이 식에서 X와 Y는 선형 관계에 있다고 할 수 있음
    - X의 값이 증가하면 Y의 값도 2배의 비율로 증가
    - 그래프로 나타내면 직선으로 표현

Linear transformation

- **선형결합 = 선형조합 (linear combination)**
  - 벡터 a와 가 b존재할 경우 각 벡터의 element를 상수배 한 후 결과를 더하는 연산
  - 각 벡터들이 가지는 element를 변화시키면 벡터 덧셈을 통해 생성된 벡터로 해당 차원의 모든 점들을 만들 수 있음
  - 선형결합을 통해 생성된 공간을 우리는 생성공간(span) 혹은 벡터공간(vector space)라고 부름
  - **벡터공간 (vector space)**
    - 일정한 조건을 만족하는 벡터들의 집합
    - 다음 조건을 만족해야 함

$$c = \alpha_1 a + \alpha_2 b$$



벡터의 덧셈

1. **덧셈에 대한 닫힘(Closure under Addition)**: 모든 벡터 $\mathbf{u}$와 $\mathbf{v}$에 대해, $\mathbf{u} + \mathbf{v}$도 벡터 공간에 속해야 합니다.
2. **스칼라 곱에 대한 닫힘(Closure under Scalar Multiplication)**: 모든 스칼라 c와 벡터 $\mathbf{v}$에 대해, $c\mathbf{v}$도 벡터 공간에 속해야 합니다.
3. **영벡터(Zero Vector) 존재**: 영벡터는 덧셈에 대한 항등원입니다. 즉, $\mathbf{0} + \mathbf{v} = \mathbf{v}$가 모든 벡터 $\mathbf{v}$에 대해 성립해야 합니다.
4. **가역 덧셈(Additive Inverse) 존재**: 모든 벡터 $\mathbf{v}$에 대해, $\mathbf{v}$와 더해져서 영벡터가 되는 벡터 $-\mathbf{v}$가 존재해야 합니다.

Linear Transformation

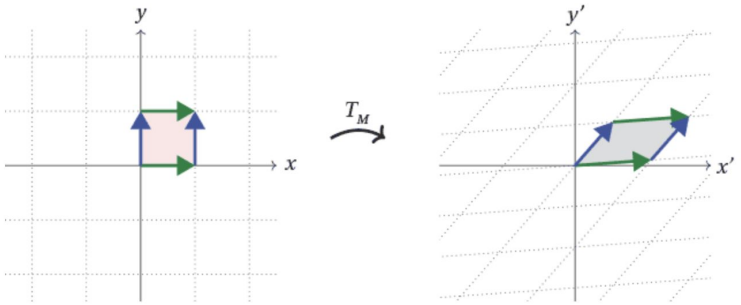
- **선형변환 (Linear Transformation)**
  - 벡터 공간 내에서 벡터를 다른 벡터로 mapping하는 것을 의미 (함수 = 행렬 / 모든 행렬은 선형 변환)
  - 벡터의 크기, 방향, 또는 위치를 동시에 변환하면서도 그들의 상대적인 관계를 유지하는 연산으로 이해
  - 아래와 같은 성질을 가짐

1. **덧셈 보존성**: $T(\mathbf{u}+\mathbf{v}) = T(\mathbf{u}) + T(\mathbf{v})$

   벡터 u와 v에 대해서, 선형 변환 T가 적용된 결과인 T(u)와 T(v)의 합은 T(u + v)와 같습니다.

2. **스칼라 곱 보존성**: $T(\mathbf{cu}) = cT(\mathbf{u})$

   벡터 u에 대해서, 선형 변환 T가 적용된 결과인 T(u)에 스칼라 c를 곱한 것과 cT(u)가 같습니다.



Linear Independence

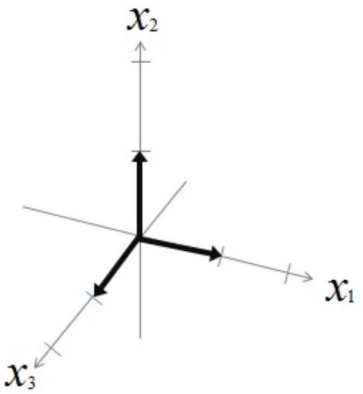
- 선형결합이 0벡터가 되는 경우로 선형 독립과 선형 종속을 판

$$c_1x_1 + c_2x_2 + \cdots + c_Nx_N = 0$$

- **선형 종속 (linearly dependent)**
  - 어떤 벡터가 다른 벡터들의 선형 조합으로 만들어질 수 있는 경우
  - 위 식을 만족하는 모두 0이 아닌 스칼라 값이 존재한다면-> 선형 종속
  - 스칼라와 벡터를 잘 조합하면 0 만들 수 있음
- **선형 독립 (linearly independent)**
  - 어떤 벡터가 다른 벡터들의 선형 조합으로 만들어질 수 없는 경우
  - 위 식을 만족하는 모두 0이 아닌 스칼라 값이 없으면 -> 선형 독립
  - 선형조합이 영벡터가 되려면 모든 스칼라가 0이어야 한다는 의미 (0을 사용하는거 빼고는 다른 벡터와 동일해질 수 없음)
  - 이는 해에 대한 유일한 값을 가진다는 의미임 (해를 찾을 수 있음)
  - 이 성질을 이용해 공간을 만들 수 있는 집합들을 생성하고 이를 통해 차원을 줄여도 데이터가 보존됨

Linear Independence

R^3의 세벡터 $\begin{pmatrix}1\\0\\0\end{pmatrix}, \begin{pmatrix}0\\1\\0\end{pmatrix}, \begin{pmatrix}0\\0\\1\end{pmatrix}$



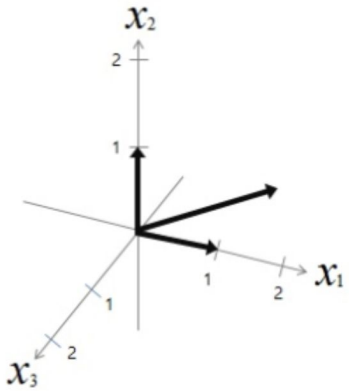
$$\begin{pmatrix}0\\0\\0\end{pmatrix} = c_1\begin{pmatrix}1\\0\\0\end{pmatrix} + c_2\begin{pmatrix}0\\1\\0\end{pmatrix} + c_3\begin{pmatrix}0\\0\\1\end{pmatrix}$$ 의 해는

유일하게 $\begin{cases} c_1 = 0 \\ c_2 = 0 \\ c_3 = 0 \end{cases}$ 뿐이므로

R^3의 세벡터 $\begin{pmatrix}1\\0\\0\end{pmatrix}, \begin{pmatrix}0\\1\\0\end{pmatrix}, \begin{pmatrix}0\\0\\1\end{pmatrix}$ 은

선형독립 (Linearly independent) 임

R^3의 세벡터 $\begin{pmatrix}1\\0\\0\end{pmatrix}, \begin{pmatrix}0\\1\\0\end{pmatrix}, \begin{pmatrix}2\\1\\0\end{pmatrix}$



$$\begin{pmatrix}0\\0\\0\end{pmatrix} = c_1\begin{pmatrix}1\\0\\0\end{pmatrix} + c_2\begin{pmatrix}0\\1\\0\end{pmatrix} + c_3\begin{pmatrix}2\\1\\0\end{pmatrix}$$ 의 해는

$\begin{cases} c_1 = 0 \\ c_2 = 0 \\ c_3 = 0 \end{cases}$ 외에 $\begin{cases} c_1 = 2 \\ c_2 = 1 \\ c_3 = -1 \end{cases}$ $\begin{cases} c_1 = -2 \\ c_2 = -1 \\ c_3 = 1 \end{cases}$

등이 추가로 더 있으므로

R^3의 세벡터 $\begin{pmatrix}1\\0\\0\end{pmatrix}, \begin{pmatrix}0\\1\\0\end{pmatrix}, \begin{pmatrix}2\\1\\0\end{pmatrix}$ 은

선형종속, 1차 종속 (Linearly dependent) 임

Linear Independence

- **벡터 공간(vector space) 또다른 특징**
  - 서로 독립인 벡터를 선형 조합하여 만들어지는 모든 벡터의 집합
- **기저 벡터(basis vector)**
  - 벡터 공간을 만드는 서로 독립인 벡터 (선형종속인 벡터는 기저 벡터가 될 수 없음)
- **표준 기저 벡터(standard basis vector)**
  - 기저 벡터 중 원소의 하나만 값이 1인 벡터 (단위행렬 I를 이루는 벡터들)
  - 표준 기저 벡터를 열로 가지는 행렬이 항등행렬

$$e_1 = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 1 \end{bmatrix}, e_2 = \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}, \dots, e_n = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Eigen Value & Eigen Vector

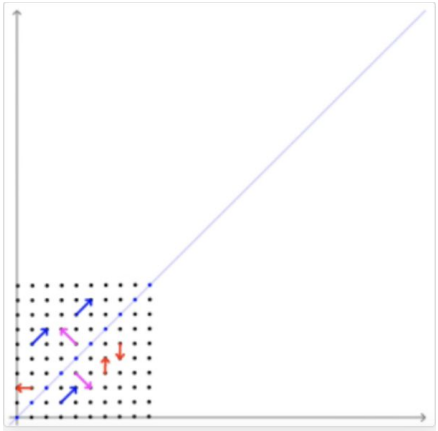
- **고유벡터 (Eigenvector)**
  - 정방행렬 A를 선형변환으로 봤을 때, 선형변환 A에 의한 변환 결과가 자기 자신의 상수배가 되는 0이 아닌 벡터
- **고유값 (Eigenvalue)**
  - 위 설명에 대한 상수배 값
- 하단 식에서 λ는 '행렬 A의 고유값', v는 '행렬 A의 λ에 대한 고유벡터

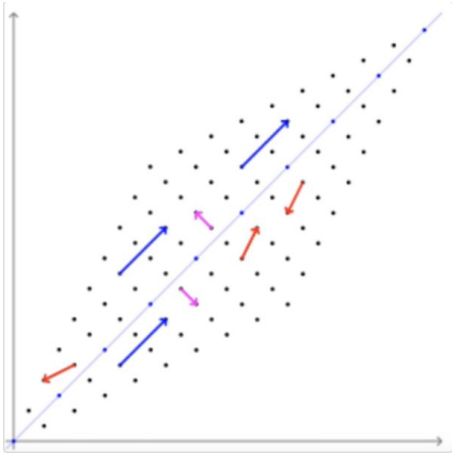
$$A\mathbf{v}=\lambda\mathbf{v} \quad \text{---(1)}$$

$$\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix}\begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix}=\lambda\begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix} \quad \text{--- (2)}$$

Eigen Value & Eigen Vector

- 열 벡터 v에 행렬 A를 곱하는 것을 '열 벡터 v에 선형 변환 A를 해주었다는 의미
- v라는 열 벡터에 선형 변환 A를 해준 결과가 열 벡터 v의 상수 배(λ)와 동일한 경우
- Av = λv이 만족한다는 것은 벡터 v에 대해 선형 변환 A를 해주었을 때, 벡터 v의 방향은 변하지 않고 크기만 변했다는 뜻
- 선형 변환을 하면 보통 방향이 바뀜, 하지만 그렇지 않은 경우도 존재 -> 이 경우만 말하는 것
- 하단의 예시에서, 선형 변환을 진행
    - 파란색과 분홍색은 고유 벡터 O / 빨간색은 방향도 변했으므로 고유 벡터 X





Eigen Value & Eigen Vector

- 방향은 보존되고 스케일만 번형되는 것
- 예를 들어 지구의 자전의 경우 (3차원 회전 변환) 고유 벡터는 회전 변환에 영향을 받지 않는 회전 축으로 볼 수 있음 (고유값은 1)

Eigen Decomposition

- 방향의 변화가 없다는 것? -> 벡터의 각 element에 대해서 independent한 벡터를 곱해 스케일의 변화만 준다는 것 (대각행렬)
- 행렬 A의 고유값을 λi, 고유벡터를 vi, i = 1, 2,..., n이라고 할 경우
- 아래와 같이 식을 전개할 수 있음
- 그리고 행렬 A의 고유 벡터들을 열 벡터로 하는 행렬을 P, 고유값을 대각원소로 가지는 대각 행렬을 Λ라 하면 다음 식이 성립
- 을 행렬 A에 대한 고유값 분해했다고 함 (행렬 A를 고유 벡터와 고유값으로 분해한 것)

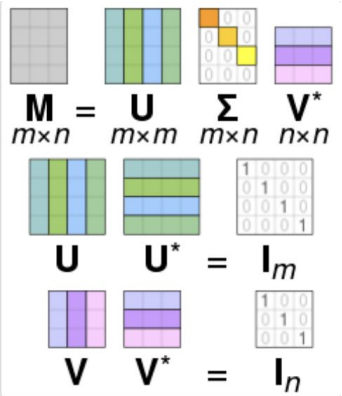
$$A\boldsymbol{v}_1=\lambda_1\boldsymbol{v}_1$$
$$A\boldsymbol{v}_2=\lambda_2\boldsymbol{v}_2$$
$$\vdots$$
$$A\boldsymbol{v}_n=\lambda_n\boldsymbol{v}_n$$

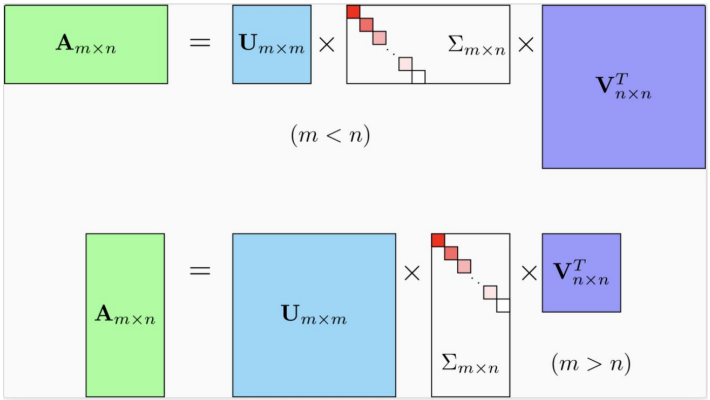
$$A[\ \boldsymbol{v}_1\ \boldsymbol{v}_2\ \cdots\ \boldsymbol{v}_n\ ]=[\ \lambda_1\boldsymbol{v}_1\ \lambda_2\boldsymbol{v}_2\ \cdots\ \lambda_n\boldsymbol{v}_n\ ]$$
$$=[\ \boldsymbol{v}_1\ \boldsymbol{v}_2\ \cdots\ \boldsymbol{v}_n\ ]\begin{bmatrix}\lambda_1 & & 0\\ & \lambda_2 & \\ & & \ddots\\ 0 & & \lambda_n\end{bmatrix}$$

$$AP=P\Lambda$$
$$A=P\Lambda P^{-1}$$

Singular Vector Decomposition

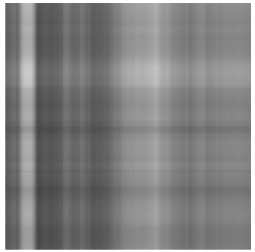
- 고유값 분해는 정방 행렬(행과 열의 크기가 같은 행렬)에 대해서만 가능
- 하지만 특이값 분해는 행과 열의 크기가 다른 모든 직각 행렬에 대해서도 적용 가능
- m x n 크기의 행렬 A는 다음과 같이 분해될 수 있음
- m x n 크기의 행렬 A는 m x m 크기의 행렬 U와 m x n 크기의 ∑ 그리고 n x n 크기의 Vt로 나뉨
- 행렬 U와 V에 속한 벡터를 특이 벡터(Singular Vector)라고 함
    - 모든 특이 벡터는 서로 직교하는 성질을 가짐
- ∑는 직사각 대각 행렬이며, 행렬의 대각에 위치한 값만 0이 아니고 나머지 위치의 값은 모두 0
- ∑의 0이 아닌 대각 원소값을 특이값(Singular Value) 라고 함

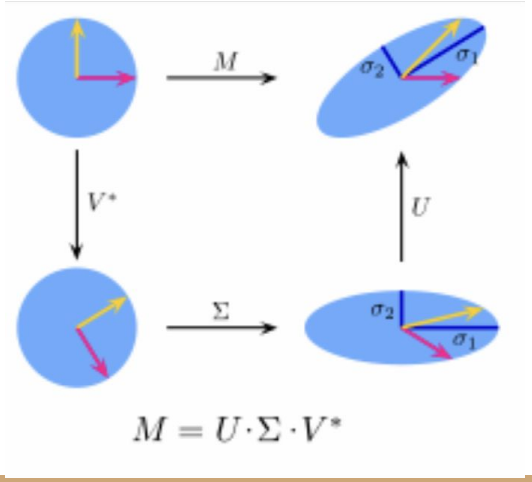




Singular Vector Decomposition

- 모든 행렬은 선형 변환
- 특이값 분해를 통해 선형 변환의 절차를 3가지로 기하학적으로 분리하는 효과가 있음
- 직교 행렬은 선형 변환 중 회전 변환
  a. Vt로 회전 변환(Rotate)
  b. ∑로 스케일 변환(Stretch)
  c. 다시 U로 회전 변환(Rotate)
- 결국 행렬의 특이값(Singular Value)은 이 행렬로 표현되는 선형 변환의 스케일 변환(Stretch)을 나타내는 값
- PCA의 일반화된 버전
- 압축 및 차원 축소 기술로 많이 활용



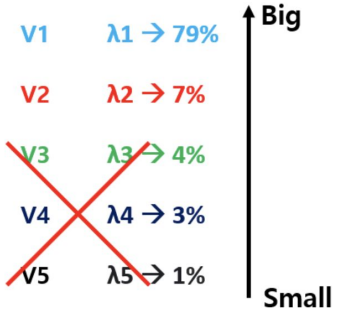


Principal Component Analysis

- 고차원 데이터의 차원을 축소하는 기법
- 과정
  a. 데이터의 각 차원들 간의 공분산 (covariance)를 산출
     - covariance를 구하는 방식은 x값과 y값을 각각 x, y값의 평균의 차 곱하여 더하고(= 내적) n으로 나눠줌으로써 구할 수 있음
  b. covariance matrix 생성
     - 수식은 2차원 데이터의 covariance matrix
  c. Eigenvector와 Eigenvalue를 찾음
  d. Eigenvalue가 작은 순으로 elements 정렬
  e. 원하는 만큼 Eigenvector를 쳐냄으로써 차원 축소 진행

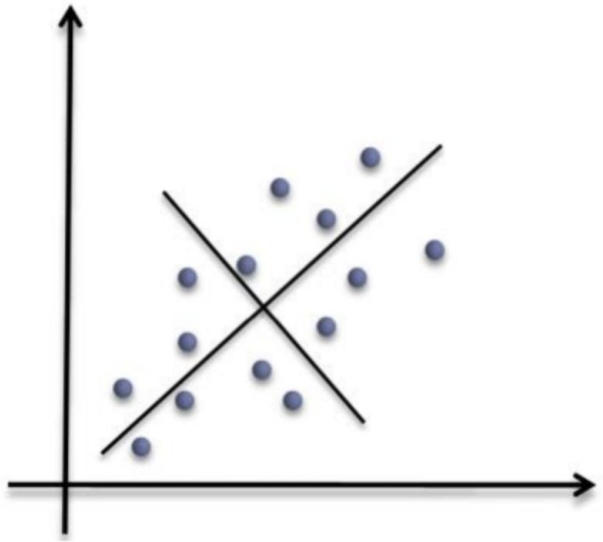
$$Cov(x,y) = \frac{\sum (x-\bar{x})(y-\bar{y})}{n}$$

$$\Sigma = \begin{pmatrix} Var(X) & Cov(X,Y) \\ Cov(X,Y) & Var(Y) \end{pmatrix}$$



Principal Component Analysis

- 데이터의 구조를 가장 잘 반영하는 eigen value와 eigen vector를 찾는 것이 목적
- eigenvalue는 중심 축(기저), 행렬은 공분산 행렬
- 즉 각 eigenvalue 중에 가장 긴 값이 가장 넓은 데이터의 분산을 갖고 있는 것



끝!