선형대수학 클린업 2주차

[Week2 - Contents]

1. 선형대수의 가하학적 접근
    1) 노름(Norms)
    2) 내적(Inner Products)
    3) 각도(Angle)
    4) 직교성(Orthogonality)
    5) 직교기저(Orthogonal Basis)와 정규직교기저(Orthogonal Basis)
    6) 직교여공간(Orthogonal Complement)
2. 기하학적 접근의 응용
    1) 투영(Projections)
    2) 회전(Rotations)
3. 행렬의 분해/인수화(Matrix Decompositions)
    1) 행렬식(Determinant) 및 대각합(Trace)
    2) 고유값(Eigenvalues)과 고유벡터(EigenVector)
    3) 촐레스키분해(Cholesky Decomposition)
    4) 고유값 분해(EigenDecomposition)
    5) 대각화(Diagonalization)
    6) 특이값 분해(Singular Value Decomposition)
4. Linear Algebra in Machine Learning(응용편)
    1) 웹 크롤링 in Python
    2) Regression Analysis in Python
    3) SVD in Python

1. 선형대수의 기하학적 접근

1) 노름(Norms)

벡터의 노름(Norm)을 간단하게 말하면 벡터의 '크기'를 말한다.
벡터의 크기를 구하는 방법이라 하면 다들 익히 알고 있는 공식이 있을 것이다.

하지만, 이는 사실 노름의 종류 중 하나일 뿐이다.

노름의 공식은 다음과 같다.

$$L_p = \sqrt[p]{|x_1|^p + |x_2|^p + \cdots + |x_n|^p}$$

여기서 p는 노름의 차수로 p가 1이면 L1 Norm, p가 2이면 L2 Norm, … 과 같이 정의된다.

참고로, 가장 많이 쓰는 Norm으로는 L1 Norm과 L2 Norm이 있으며 수식은 다음과 같다.

L1 Norm: $L_1 = |x_1| + |x_2| + \cdots + |x_n|$

L2 Norm: $L_2 = \sqrt[2]{|x_1|^2 + |x_2|^2 + \cdots + |x_n|^2}$

L1 Norm은 맨허튼 노름(Manhattan norm)이라고도 하는데, 벡터의 요소에 대한 절댓값의 합으로, 좌표평면에서 각 좌표 축을 따라 움직이는 거리를 표현한다.

L2 Norm은 유클리드 노름(Euclidean Norm)이라고도 하는데, 원점에서 x라는 벡터에 연결된 직선 거리를 의미하며, 많은 사람들이 벡터의 크기로 알고 있는 만큼 가장 대중적으로 쓰인다.

2) 내적(Inner Products)

내적(Inner Products)이란 어떤 두 벡터의 각 성분끼리의 곱의 합을 의미한다.

즉, $x = (x_1, x_2, \cdots, x_n)$, $y = (y_1, y_2, \cdots, y_n)$ 일 때, x와 y의 내적 $x \cdot y = x_1y_1 + x_2y_2 + \cdots + x_ny_n$ 이다.

이 때, 두 벡터의 내적을 행렬의 곱연산처럼 나타낼 수도 있는데,

x와 y가 크기가 같은 열벡터일 때, 내적 연산에서 앞 벡터를 전치(Transpose)시킨 후 뒤 벡터와 곱연산을 하면 내적의 결과와 같다.

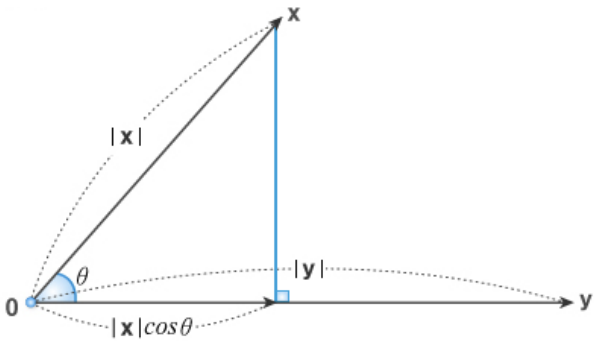
예를 들면 다음과 같다.

$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$, $y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$ 일 때, $x \cdot y = x^T y = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = x_1y_1 + x_2y_2 + x_3y_3$ 이 되는 것을 볼 수 있다.

또한, 내적을 성분이 아니라 벡터 자체를 가지고 계산하면 다음과 같은 수식을 가진다.

x·y = |x| |y| cosθ, 여기서 θ는 벡터 x와 y 사이의 각도를 말한다.

이를 기하학적으로 해석하면 다음과 같다.



그림에서 보면 알 수 있듯이, 내적 연산은 어떤 벡터 x를 다른 벡터 y에 투영시켰을 때 생기는 크기와 벡터 y의 크기를 곱한 것과 결과가 같다. 이 때 방향을 고려해야 한다.

3) 각도(Angle)

벡터의 각도를 구하는 방법에는 여러가지가 있는데, 바로 위에서 소개한 내적을 이용한 방법을 소개하겠다.

당연하게도, 각 벡터가 어떤 모양을 가지고 있는지 모른다면 두 벡터 사이의 각도를 구한다는 것은 불가능하다.
그러므로 두 벡터 사이의 각도를 구하기 위해서는 두 벡터가 어떤 모양을 가지고 있는지 아는 게 필수적인데, 이는 곧 두 벡터의 성분을 알고 있는 것과 같다. 따라서, 두 벡터의 성분을 안다는 가정 하에 설명을 진행하겠다.

벡터 x, y에 대해 x·y = |x| |y| cosθ 이므로
cosθ = $\frac{x \cdot y}{|x||y|}$ 임을 알 수 있다.

따라서, θ = $\cos^{-1}(\frac{x \cdot y}{|x||y|})$ 이 됨을 알 수 있다.

즉, 우리는 x·y, |x|, |y|을 알고 있다면 각도를 구할 수 있는 것이다.
또한, x·y, |x|, |y|은 벡터 x, y의 성분을 알고 있으면 구할 수 있는 값들이므로 두 벡터의 성분을 알고 있을 때 우린 내적을 이용하여 각도를 구할 수 있는 것이다.

4) 직교성(Orthogonality)

■ 직교(Orthogonal)

벡터의 각도를 설명할 때 특별한 경우가 존재한다.

바로 각도가 90°(직각)일 때가 그 경우에 해당한다.

이 경우가 특별한 이유는 바로 내적의 값에서 알 수 있는데,

다음 식을 살펴보자.

$\cos\theta = \frac{x \cdot y}{|x||y|}$ 에서 각도가 직각이면 cosθ의 값은 0이 된다.

|x|, |y|는 0보다 큰 값일 때를 가정하므로 이는 곧 내적 값 x·y가 0이 된다.

즉, **"두 벡터 x,y가 직교한다(Orthogonal하다)" 는 "내적 값 x·y가 0이다" 라는 말과 같다.**

■ 정규 직교(Orthonormal)

Orthonormal한 경우는 Orthogonal한 경우에서 한 가지 조건이 추가되는데,

바로 "벡터들의 크기가 1"이라는 조건이다.

따라서, **"벡터 x, y가 Orthonormal하다"라는 말은 "내적 값 x·y가 0이며, |x| = |y| = 1이다" 라는 말과 같다.**

■ 직교행렬(Orthogonal Matrix)

직교행렬(Orthogonal Matrix)이란 $AA^T = I = A^TA$를 만족하는 정방행렬(Square Matrix)을 말한다.

이 때, 이런 성질을 만족하는 행렬의 이름이 직교행렬인 이유는 위와 같은 행렬은 열벡터들이 Orthonormal한 경우이기 때문이다.

직교행렬은 중요한 성질이 있는데, 바로 이 **직교행렬을 이용한 선형 변환(Linear Transformation)은 input vector x의 크기가 변하지 않는다는 것**이다.

증명은 매우 간단하다.

$||Ax||^2 = (Ax)^T(Ax) = x^TA^TAx = x^TIx = x^Tx = ||x||^2$

따라서, $||Ax||^2 = ||x||^2$ 이므로 $||Ax|| = ||x||$ 이다.

즉, A가 직교행렬일 때 선형 변환은 input vector의 크기를 변화시키지 않는다.

■ 직교집합(Orthogonal Set)과 정규직교집합(Orthonormal Set)

위에서는 두 벡터 사이의 직교만을 다뤘는데, 이를 일반화시킨 것이 직교집합이다.

즉, 직교집합(Orthogonal Set)이란 벡터들의 집합 $\{x_1, x_2, \cdots, x_n\}$에서 각각의 x_i가 서로 직교하는 집합을 말한다.
→ $x_i \cdot x_j = 0$ where $i \neq j$

또한, 정규직교집합(Orthonormal Set)이란 벡터들의 집합 $\{x_1, x_2, \cdots, x_n\}$에서 각각의 x_i가 서로 정규직교하는 집합을 말한다.

→ $x_i \cdot x_j = 0$ where $i \neq j$ & $||x_i|| = 1$

5) 직교기저(Orthogonal Basis)와 정규직교기저(Orthonormal Basis)

■ 직교기저(Orthogonal Basis)

직교기저(Orthogonal Basis)란 기저 $B = \{b_1, b_2, \cdots, b_n\}$에서 기저벡터 b_i들이 서로 직교하는 기저를 말한다.

→ $b_i \cdot b_j = 0$ where $i \neq j$

■ 정규직교기저(Orthonormal Basis)

정규직교기저(Orthonormal Basis)란 기저 $B = \{b_1, b_2, \cdots, b_n\}$에서 기저벡터 b_i들이 서로 정규직교하는 기저를 말한다.

→ $b_i \cdot b_j = 0$ where $i \neq j$ & $||b_i|| = 1$

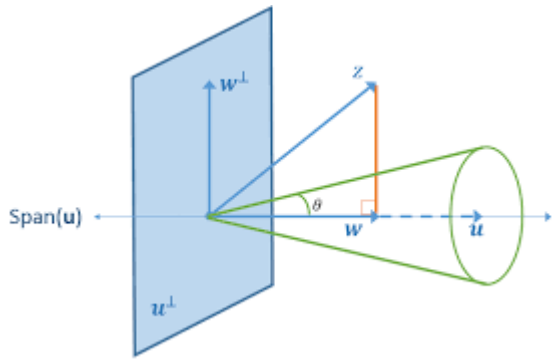
참고로, Standard Basis $\{e_1, e_2, \cdots, e_n\}$은 정규직교기저임을 쉽게 알 수 있다.

6) 직교여공간(Orthogonal Complement)

직교여공간(Orthogonal Complement)의 정의는 다음과 같다.

집합 S가 공집합이 아닌 $\mathbb{R}^n$에 속하는 벡터의 집합일 때, S의 Orthogonal Complement는 S에 속하는 모든 벡터들에 대하여 직교하는 벡터들의 집합이다. 이 때, 벡터들은 $\mathbb{R}^n$에 속하며, S의 Orthogonal Complement를 $S^{\perp}$로 표기한다.

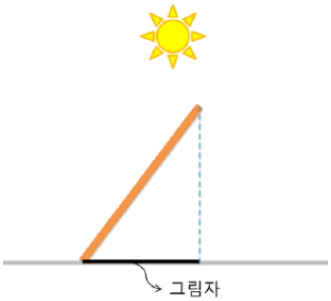
예를 들어, 어떤 직선의 Orthogonal Complement는 그 직선을 법선으로 하는 평면이 된다.

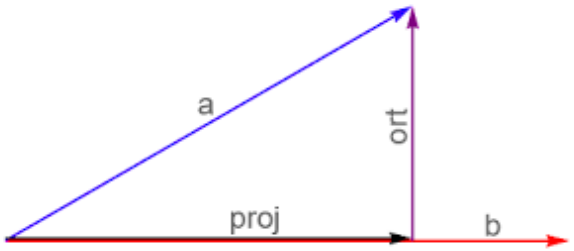


2. 기하학적 접근의 응용

1) 정사영(Orthogonal Projections)

투영(Projections)이란 어떤 벡터를 다른 벡터로 옮겨 표현하는 것을 말한다.
일반적으로 벡터를 다른 벡터로 투영시킬 때는 여러가지 각도로 투영이 가능한데,
정사영은 직각을 이루게 하여 투영시킬 때를 말한다.
이는 다음 그림을 보면서 쉽게 이해할 수 있다.





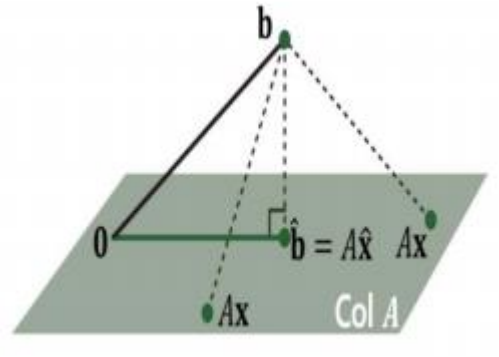
왼쪽 그림을 보면, 어떤 막대기가 있을 때 햇빛에 의해 그림자가 땅에 생긴다. 이 때 막대기를 어떤 벡터 x라고 하고 땅을 벡터 y라고 하면, 땅에 생기는 그림자가 x의 y에대한 투영벡터가 되는 것이다.

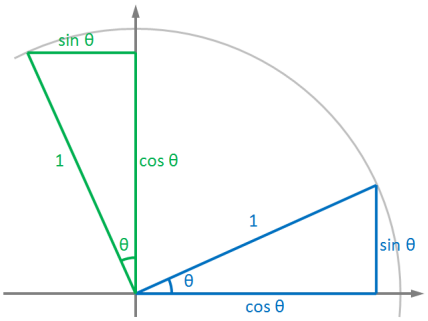
오른쪽 그림은 실제로 벡터들의 투영에 관해 표기한 것인데, a의 b에 대한 투영 벡터를 **$\mathrm{proj}_b(a)$**라고 표기한다.

이 때, $\mathrm{proj}_b(a)$는 다음과 같이 구할 수 있다.

$$\mathrm{proj}_b(a) = tb = \frac{a \cdot b}{\|b\|^2} b$$

참고로, 정사영은 직각을 이루며 투영시킨 벡터이므로 $\mathrm{proj}_b(a)$는 벡터 a와 b사이의 최소거리에 위치함이 자명하다.
다음 그림을 보면 이해가 빠를 것이다.





정사영을 일반화하기 위해 평면일 때의 그림을 가져와봤다.
그림을 보면 사선으로 이루어진 거리보다는 직각일 때의 거리가 짧은 것이 자명하다.

이 개념은 선형대수학을 응용함에 있어서 매우 중요한 개념이므로 제대로 이해하고 넘어갔으면 한다.

2) 회전(Rotations)

벡터의 회전(Rotations)을 선형 변환을 통해 설명할 수 있어서 간단하게 언급하고 넘어가겠다.

■ Rotations in $\mathbb{R}^2$

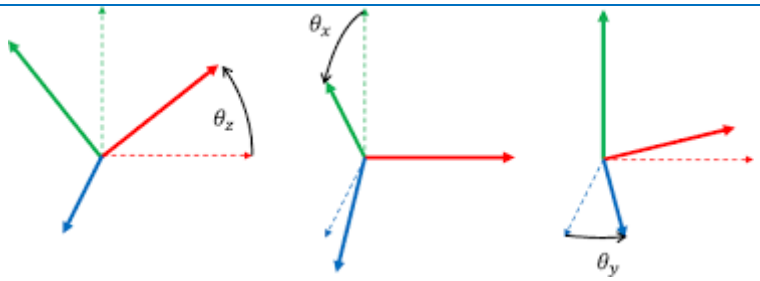
2차원에서 회전은 행렬 R = $\begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix}$ 를 곱함으로써 구할 수 있다.

이 때, θ는 회전하고 싶은 각도이다.

■ Rotations in $\mathbb{R}^3$

3차원에서의 회전은 2차원과는 조금 다른 형태를 가진다.

이는 3차원에서의 회전은 어떤 평면에서 회전하느냐에 따라 회전 방향이 달라지기 때문인데,
다음 그림을 보면서 이해하자.



위 그림에서 x축을 고정시키고 yz평면에서 회전시킨 각도를 θ_x, y축을 고정시키고 xz평면에서 회전시킨 각도를 θ_y, z축을 고정시키고 xy평면에서 회전시킨 각도를 θ_z라 정의한 것을 볼 수 있다.

이처럼 3차원에서의 회전은 어떤 평면에서 회전하는 지에 따라 행렬의 모양이 달라진다.

그 행렬들은 다음과 같다.

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\theta & -sin\theta \\ 0 & sin\theta & cos\theta \end{bmatrix},\ R_y(\theta) = \begin{bmatrix} cos\theta & 0 & sin\theta \\ 0 & 1 & 0 \\ -sin\theta & 0 & cos\theta \end{bmatrix},\ R_z(\theta) = \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

더 나아가, 일반화시켜 $\mathbb{R}^n$에서의 회전도 행렬을 정의할 수 있는데, 이는 본 교안에서는 다루지 않기로 한다.

3. 행렬의 분해/인수화(Matrix Decompositions)

1) 행렬식(Determinant) 및 대각합(Trace)

■ 행렬식(Determinant)

행렬식(Determinant)은 선형대수학에서 매우 중요한 개념이다.

행렬식은 함수인데, 정사각행렬이 실수에 대응되는 함수이다.
또한, 행렬식은 행렬에 A에 대해 det(A)로 표기하며,

행렬 A가 $\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$로 주어진다면, det(A) = $\begin{vmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{vmatrix}$ 로 표기한다.

행렬식의 계산 방법은 매우 다양하다.

우선, 가장 기본적인 2x2 행렬식은 다음과 같다.

A = $\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$일 때, det(A) = $a_{11}a_{22} - a_{12}a_{21}$

2x2 행렬의 행렬식은 매우 간단하며, 활용도도 높기 때문에 외우고 있는 것을 권장한다.

이를 n차원으로 확장했을 때, 즉 n x n 행렬의 행렬식을 계산할 때는 계산 방법이 다양하게 존재한다.

본 교안에서는 이에 대해 자세히 다루지 않고 **Laplace Expansion**이라고 불리는 행렬식 계산 방법만을 간단히 소개하고 넘어가겠다. 그 내용은 다음과 같다.

임의의 정사각행렬 $A \in \mathbb{R}^{n\,x\,n}$에 대해 행렬식은 다음과 같다.

(i) 열을 따라서 계산할 때 : $\det(A) = \sum_{k=1}^{n}(-1)^{k+j} a_{kj} \det(A_{k,j})$

(ii) 행을 따라서 계산할 때 : $\det(A) = \sum_{k=1}^{n}(-1)^{k+j} a_{jk} \det(A_{j,k})$

■ 행렬식과 역행렬

행렬식의 가장 중요한 성질 중 하나는 역행렬과 관련이 있다.

임의의 n x n 정사각행렬 A에 대해 "A의 역행렬이 존재하지 않는다"의 필요충분조건은 "det(A) = 0"이다.

즉, A의 역행렬이 존재한다면 $\det(A) \neq 0$이다.

■ 대각합(Trace)

어떤 정사각행렬 A의 대각합(trace)은 행렬의 대각성분들의 합을 의미하며 이를 tr(A)로 표기한다.

즉, $A \in \mathbb{R}^{n\,x\,n}$에 대해 $\mathrm{tr}(A) := \sum_{i=1}^{n} a_{ii}$ 이다.

2) 고유값(Eigenvalues)과 고유벡터(EigenVectors)

■ 고유값(Eigenvalues)과 고유벡터(EigenVectors)

고유값(Eigenvalues)이란 $A \in \mathbb{R}^{n\,x\,n}$에 대해 $\exists x \in \mathbb{R}^n \backslash \{0\}$ $Ax = \lambda x$일 때 λ를 말한다. 이 때, λ는 실수이다.

또한, 이 때 벡터 x를 고유벡터(EigenVector)라고 한다.

■ 고유공간(EigenSpaces)

고유공간(EigenSpace)란 어떤 고유값 λ에 대응하는 모든 고유벡터들을 모아놓은 집합을 말한다.

■ 고유값, 고유벡터, 고유공간 구하기

다음 수식을 보자.

$A \in \mathbb{R}^{n\,x\,n}$이며 벡터 x가 0이 아닌 벡터일 때, A에 대한 고유값 λ, 고유벡터 x는 다음과 같은 방식으로 구할 수 있다.

$Ax = \lambda x = \lambda Ix$

$Ax - \lambda Ix = (A - \lambda I)x = 0$

이 때, **x가 0이 아닌 벡터를 가지기 위해서는 반드시 $A - \lambda I$의 역행렬이 존재하지 않아야 한다**.
(∵ $A - \lambda I$의 역행렬이 존재한다면, $x = (A - \lambda I)^{-1}0 = 0$ 이 되므로 x=0이 된다)

따라서, 위의 조건들을 통해 **$\det(A - \lambda I) = 0$** 이라는 방정식을 얻을 수 있고, 이 방정식을 풀어서 고유값과 고유벡터를 구할 수 있다.

고유값과 고유벡터는 중요한 개념인 만큼, 예를 들어보자.

행렬 $A = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$의 고유값과 고유벡터를 구해보자.

$$\det(A - \lambda I) = \begin{vmatrix} 2-\lambda & 1 \\ 1 & 2-\lambda \end{vmatrix} = (2-\lambda)^2 - 1 = \lambda^2 - 4\lambda + 3 = (\lambda - 1)(\lambda - 3) = 0$$

$$\therefore \lambda = 1 \; or \; 3$$

(i) $\lambda = 1$일 때

$$Ax = x, \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$x_1 + x_2 = 0, x_2 = -x_1, x = x_1\begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

즉, $a\begin{bmatrix} 1 \\ -1 \end{bmatrix}, a \in \mathbb{R}$인 임의의 벡터가 고유값 $\lambda = 1$에 대한 고유벡터가 된다.

또한, 이 때 $\text{span}\{\begin{bmatrix} 1 \\ -1 \end{bmatrix}\}$이 고유공간이 된다.

(ii) $\lambda = 3$일 때

$$Ax = 3x, \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$x_1 - x_2 = 0, x_2 = x_1, x = x_1\begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

즉, $a\begin{bmatrix} 1 \\ 1 \end{bmatrix}, a \in \mathbb{R}$인 임의의 벡터가 고유값 $\lambda = 3$에 대한 고유벡터가 된다.

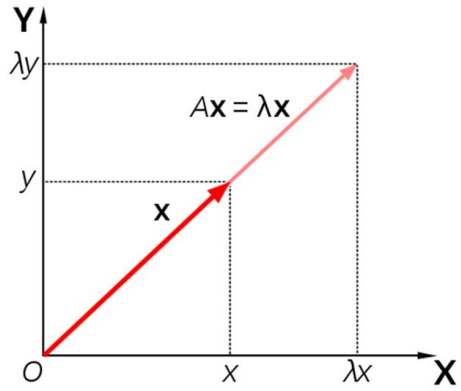
또한, 이 때 $\text{span}\{\begin{bmatrix} 1 \\ 1 \end{bmatrix}\}$이 고유공간이 된다.

■ 고유값과 선형변환

행렬 A와 벡터 x에 대해 Ax라는 값은 A에 의한 벡터 x의 선형변환으로도 해석할 수 있다.

이러한 관점에서 $Ax = \lambda x$가 의미하는 것은 **x의 A에 의한 선형변환이 x의 λ 에 의한 스칼라곱과 같은 결과**라는

것이다. 벡터의 스칼라곱은 방향을 변화시키지 않고 크기만이 변화하는 것이므로, 다시 해석하면 A에 의한 벡터 x의 선형 변환 Ax가 **"x의 방향을 그대로 두고 크기만을 변화시키는 선형 변환"**이라고 말할 수 있다.



그림처럼 벡터 x가 주어졌을 때, 방향은 변하지 않고 크기만 변화한다.
이 때, 행렬 A에 의한 선형변환 전과 후가 평행한 벡터를 **고유벡터**, 크기의 변화정도를 **고유값**이라고 한다.

3) 촐레스키분해(Cholesky Decomposition)

■ 양확정행렬(Positive-Definite Matrix)

본격적으로 촐레스키 분해에 들어가기 앞서, 양확정행렬이라는 용어의 정의에 대해서 언급하고 가겠다.

양확정행렬의 이름은 "양수를 확실하게 만드는 행렬이다"라는 것에서 유래된 것으로 보인다.

어떤 n x n 크기의 행렬 A와 임의의 n x 1크기의 벡터 x에 대해 x^TAx를 계산한다고 해보자.

이 때, 영벡터가 아닌 임의의 x에 대해 $x^TAx>0$ 을 만족할 때, 행렬 A를 Positive-Definite Matrix라고 한다.

예를 들어, $A = \begin{bmatrix} 2 & 6 \\ 6 & 20 \end{bmatrix}$이라 하자.

임의의 벡터 $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$로 표기하면, $x^TAx = \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} 2 & 6 \\ 6 & 20 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 2x_1^2 + 12x_1x_2 + 20x_2^2$이 된다.

이 때, $2x_1^2 + 12x_1x_2 + 20x_2^2 = 2(x_1 + 3x_2)^2 + 2x_2^2 > 0$ 이다. (∵x가 영벡터가 아니므로)

따라서, A는 Positive-Definite Matrix이다.

■ 촐레스키분해(Cholesky Decomposition)

우리는 어떤 실수 a에 대해, a를 작은 수의 곱으로 표현할 수 있는 것을 안다.

특히, 제곱근을 통해서 계산하면 같은 수 2개로 나누어 곱연산을 할 수 있다.
예를 들어, 9 = 3·3처럼 말이다.

행렬에서도 마찬가지로 하나의 행렬을 두개의 행렬로 분해할 수 있다.
그게 지금 다루게 되는 촐레스키분해(Cholesky Decomposition)이다.

그 내용은 다음과 같다.

Symmetric하며 Positive definite한 행렬 A에 대해 다음을 만족하는 0이 아닌 성분이 양수로 구성된 하삼각(Lower-Triangular) 행렬 L이 존재한다.

$A = LL^T$

즉, $\begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix} = \begin{bmatrix} l_{11} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ l_{n1} & \cdots & l_{nn} \end{bmatrix} \begin{bmatrix} l_{11} & \cdots & l_{1n} \\ \vdots & \ddots & \vdots \\ 0 & \cdots & l_{nn} \end{bmatrix}$

이 때, L을 A의 **Choleksy factor**라고 하며, 유일하게 존재한다.

촐레스키 분해를 통해서 행렬의 분해를 진행할 경우, 큰 실수를 작은 2개의 실수로 분해하는 거와 달리 n x n 행렬을 n x n 행렬 2개로 분해하는 것이기 때문에 굳이 촐레스키분해를 진행하는 이유에 대해 궁금할 수 있다.

이는 행렬식과 관련이 있다.

머신러닝 등 다양한 수단에서 행렬의 Numerical한 계산이 포함되어 있는데,
이 때 행렬식도 계산에 필요한 요소 중 하나가 될 수 있다.

일반적으로, n x n 행렬의 경우 n이 크게 되면 행렬식의 계산이 매우 복잡해진다.

하지만, 예외적으로 **삼각행렬의 경우에는 행렬식의 계산이 매우 간단**하다(∵ 대각성분의 곱 = 행렬식)

촐레스키 분해를 했을 때, $A = LL^T$라 하면 행렬식 $\mathbf{det(A) = det(LL^T) = det(L)\cdot det(L^T) = det(L)^2}$이며 det(L)은 삼각행렬이므로 매우 쉽게 계산이 된다.

즉, 촐레스키 분해를 통해 계산하기 힘든 행렬의 행렬식을 계산하기 쉽게 분해할 수 있는 것이다.

4) 대각화(Diagonlization)

다음과 같이 대각선에만 0이 아닌 성분이 있고, 그 외의 성분은 모두 0인 행렬을 **대각행렬(Diagonal Matrix)**이라 한다.

$D = \begin{bmatrix} c_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & c_n \end{bmatrix}$

대각행렬은 행렬식, 거듭제곱, 역행렬 계산 등에서 굉장한 이점을 가진다.

대각행렬 $D \in \mathbb{R}^{n \times n}$와 어떤 행렬 $A \in \mathbb{R}^{n \times n}$에 대해 $D = P^{-1}AP$인 역행렬이 존재하는 행렬 $P \in \mathbb{R}^{n \times n}$가 존재할 때, A를 **대각화 가능(Diagonalizable)**하다고 하며, 위의 과정을 **대각화(Diagonalization)**이라고 한다.

5) 고유값 분해(EigenDecomposition; EVD)

어떤 정사각행렬 $A \in \mathbb{R}^{n \times n}$에 대해 A의 고유벡터들이 $\mathbb{R}^n$의 기저를 형성할 때 다음과 같은 식을 만들 수 있다.

$A = PDP^{-1}$, 이 때 $P \in \mathbb{R}^{n \times n}$이며 D는 각 성분이 A의 고유값으로 구성되어 있는 대각행렬(Diagonal Matrix)이다.

이 과정을 **고유값 분해(EigenDecomposition)**이라고 부른다.

예를 들어, $A = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$로 주어졌을 때, 고유값과 고유벡터는 $\lambda = 1$일 때 $\begin{bmatrix} 1 \\ -1 \end{bmatrix}$, $\lambda = 3$일 때 $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$이다.

따라서, 고유값 분해는 다음과 같이 이루어진다.

$$A = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} = PDP^{-1} = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} 0.5 & -0.5 \\ 0.5 & 0.5 \end{bmatrix}$$

6) 특이값 분해(Singular Value Decomposition; SVD)

■ 특이값 분해의 개념

특이값 분해(Singular Value Decomposition)는 앞서 설명했던 분해들과 달리 n x m 크기의 일반적인 행렬A에 대해서 분해가 가능한데, 이 때문에 선형대수학에서 매우 중요한 분해 방법으로 자리잡고 있다.
심지어, "Fundamental Method of Linear Algebra"라고 불릴 정도로 중요한 개념이다.

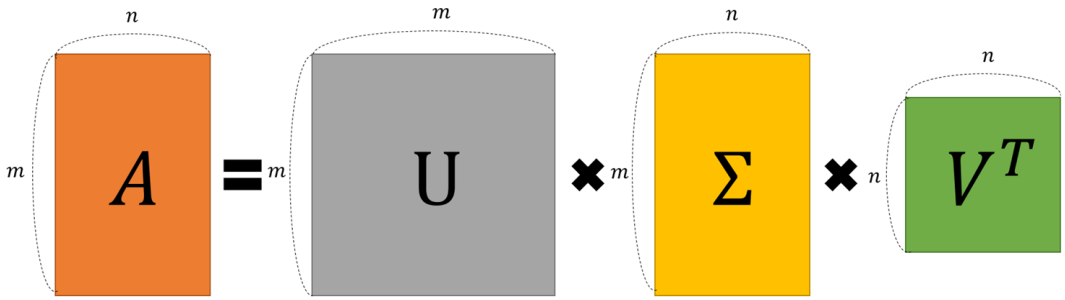
그 내용은 다음과 같다.

A : m x n 행렬이며 계수(rank)가 0≤ rank(A) ≤ min(m,n)

V^T : n x n 행렬, U : m x m 행렬, Σ : m x n 행렬

이 때, U와 V는 직교행렬이며

Σ는 성분이 $\Sigma_{ii} = 0$, $\Sigma_{ij} = 0 (i \neq j)$인 행렬



사실, 직교 행렬의 정의에 따라 $A = U\Sigma V^T = U\Sigma V^{-1}$과 같다.

또한, **Σ는 대각 행렬을 부분집합으로 가지는 m x n크기의 행렬**이라고 표현하는데, 이는 성분이 $\Sigma_{ii} = 0$, $\Sigma_{ij} = 0 (i \neq j)$인 행렬로 다음 그림처럼 표현되는 행렬을 말하기 때문이다.

m > n

$$\begin{pmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ & & \ddots & \\ 0 & 0 & \cdots & \sigma_n \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix}$$

m<n

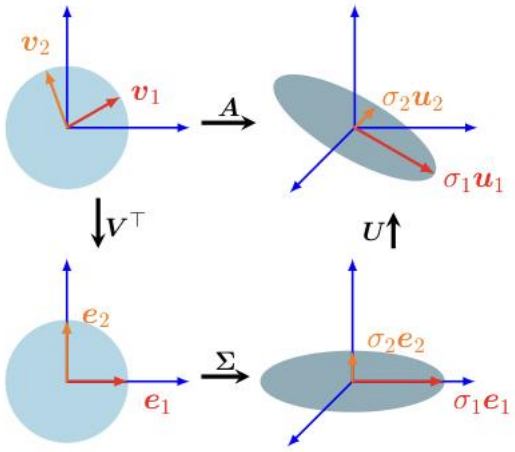
$$\begin{pmatrix} \sigma_1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 & 0 & \cdots & 0 \\ & & \ddots & & & & \\ 0 & 0 & \cdots & \sigma_m & 0 & \cdots & 0 \end{pmatrix}$$

■ 특이값 분해의 기하학적 해석

특이값 분해를 기하학적으로 해석해보자.

앞서, 행렬 A를 $A = U\Sigma V^T$로 표현하는 것을 특이값 분해라고 했었다.

이 때, 이 과정을 기하학적으로 나타내면 다음과 같다.



어떤 선형 변환 행렬 A가 존재하는데, A는 3개의 행렬의 곱연산으로 표현할 수 있고,
이는 일련의 3개의 다른 변환을 수행하는 것으로 나타낼 수 있다.

이 과정을 하나하나 해석하면 다음과 같다.

V^T : **Domain(정의역)에서** 표준 기저에서 다른 기저로 기저 변환(Basis Change)

Σ : 새로운 기저에서 값 스케일링(Scaling; 크기 변환)

U : 회전(Rotation)을 통해 **Codomain(공역)에서** 기저 변환(Basis Change)

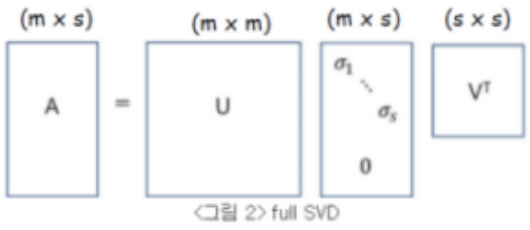
즉, SVD의 경우 **Domain과 Codomain에서 동시에 기저 변환이 일어나는 것이 특징**이다.

■ 특이값 분해의 활용

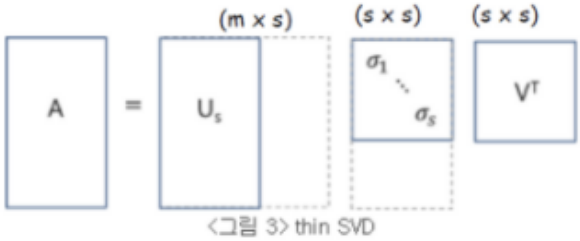
지금까지 우리가 다룬 특이값 분해는 특이값 분해의 기본적인 개념이며, Full SVD라고 불린다.

하지만 실제로는 축약된 버전인 reduced SVD를 더 많이 활용하는 편이다.

Reduced SVD에는 다음과 같은 종류들이 존재한다.

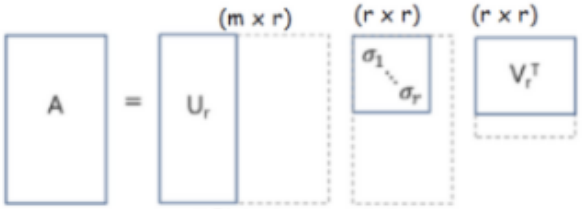


<그림 2> full SVD



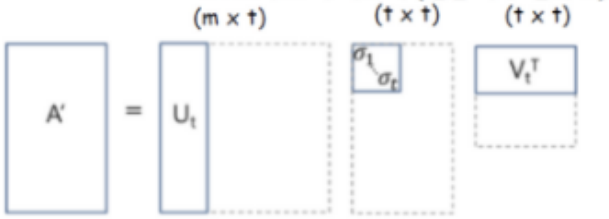
<그림 3> thin SVD

r: 0이 아닌 값들의 개수



<그림 4> Compact SVD

t: r보다도 더 작은 수(원본 복구 불가능)



<그림 5> Truncated SVD

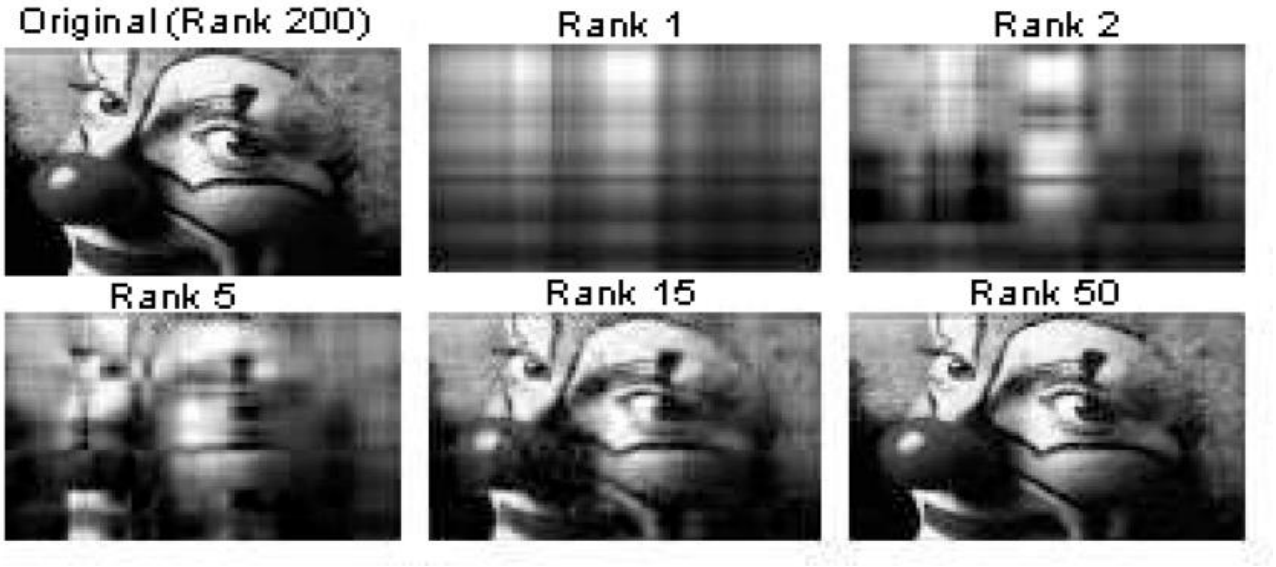
(i) <그림 3> Thin SVD : Σ에서 비대각파트(대각파트가 아닌 0으로 구성된 부분)를 삭제. 또한, U에서 이에 대응되는 열벡터들을 제거한 형태

(ii) <그림 4> Compact SVD : Σ에서 비대각파트 뿐 아니라 Singular Values 중 0인 부분까지 제거한 형태이며, 이에 따라 U와 V^T를 조정한 형태

(iii) <그림 5> Truncated SVD : Σ에서 비대각파트와 0인 Singular Values 뿐 아니라 일부 Singular Values만 남기고 제거하는 방식이며, 이에 따라 U와 V^T를 조정한 형태

주목할 점은, Thin SVD와 Compact SVD는 계산 결과 원래의 A와 동일한 행렬을 얻는 반면, Truncated SVD는 원래의 행렬 A가 보존되지 않고 A에 대한 근사행렬 A'을 얻게 된다. 이렇게 얻게 된 A'은 Matrix Norm ||A − A'||을 최소화시키는 rank t 행렬로 데이터 압축, 노이즈 제거 등에 활용된다.

아래의 그림을 보자.



Truncated SVD의 중요한 포인트는 **이미지 데이터에서 중요한 부분(일부 Singular Values)만을 남기고 필요없는 정보를 일부 제거함으로써 정보량을 줄이고** 그에 따른 **용량과 연산량을 줄이는 것**에 있다. 이미지 분석에서 이러한 기법을 사용하게 되면 더 적은 정보로 효율적이고 빠른 분석이 가능해진다.

물론, Truncated SVD에서 남길 Singular Values의 수(Rank t에서 t)를 현저히 줄이면, 우리 눈에 알아볼 수 없을 정도가 된다. 하지만 중요한 것은 결국 데이터를 처리하는 것은 사람이 아닌 컴퓨터라는 것이고, 핵심 정보를 남겨놓는 한에서 SVD를 진행하게 되면 원본만큼은 아니더라도 원본에 버금가는 이미지로 분석하여 좋은 결과를 더 효율적인 방법으로 계산이 가능하다는 것이다.