리드오프 3주차 후반부

SVD(Singular Value Decomposition)

PCA를 진행하기 위해서는 데이터의 공분산 행렬을 구해야 하는데, 데이터가 클수록 공분산 행렬을 구하기 위한 계산량이 매우 늘어난다. 또한 위에서 배웠던 고유값 분해는 **정사각 행렬(Square Matrix)**에 대해서만 사용 가능한데 보통의 경우, 직사각 행렬에 대해서도 행렬 분해가 필요한 경우가 많을 것이다. 즉 고유값 분해를 직사각 행렬에 대해 일반화하여 대각화를 진행해주는 것이 필요하고, 이것이 특이값 분해 SVD(Singular Value Decomposition)다.

개념

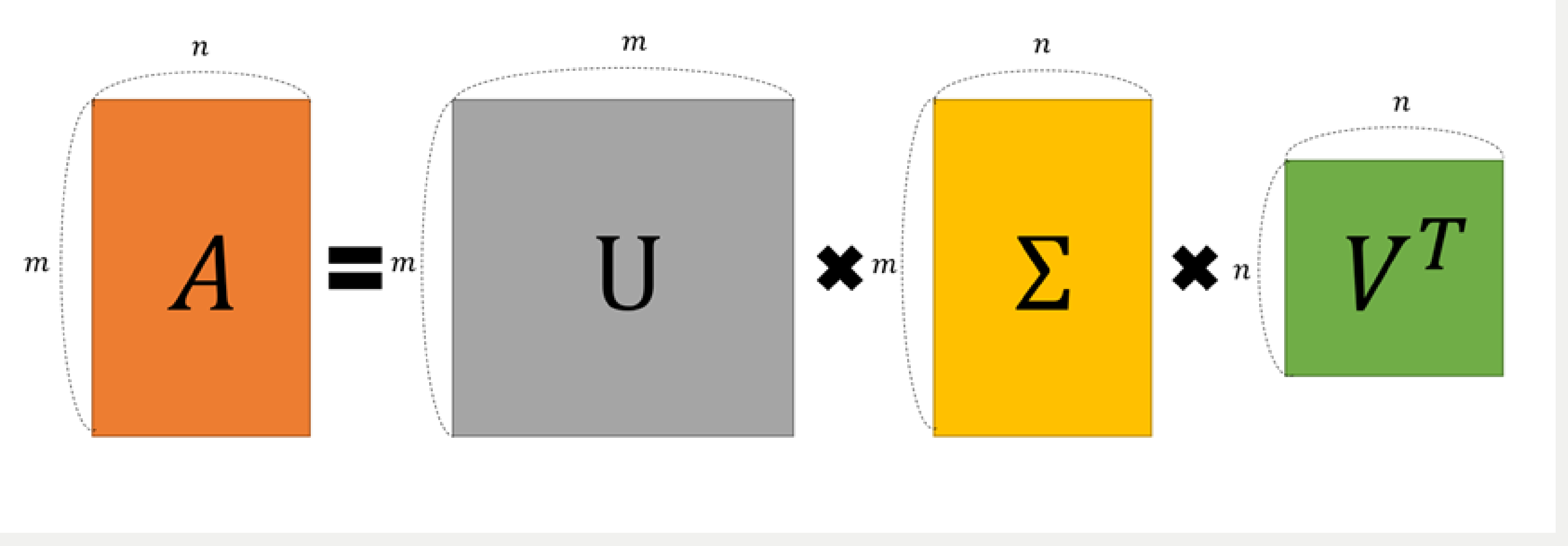
📌 $A = U\Sigma V^T$

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

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

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

이 때, U와 V는 직교행렬



우선, 모든 행렬 A에 대해 AA^T와 A^TA는 정방행렬이며, symmetric하다는 것을 짚고 넘어가자. 따라서 직교행렬로 고유값 분해가 가능한데, AA^T의 경우를 살펴보면, U는 이를 고유값 분해 $(AA^T = U(\Sigma\Sigma^T)U^T)$ 해서 얻어진 직교행렬로, U에 담겨있는 벡터를 A의 left singular vector라 부른다. V는 A^TA를 고유값 분해 $(A^TA = V(\Sigma^T\Sigma)V^T)$ 해서 얻어진 직교행렬로 V에 담겨있는 벡터를 A의 right singular vector라 부른다. 마지막으로 Σ는 AA^T, A^TA 를 고유값 분해해서 나오는 고유값들의 square root를 대각원소로 하는 $m \times n$ 직사각 대각행렬로, 그 대각 원소들을 A의 특이값이라 부른다.

이렇게 행렬을 분해하면

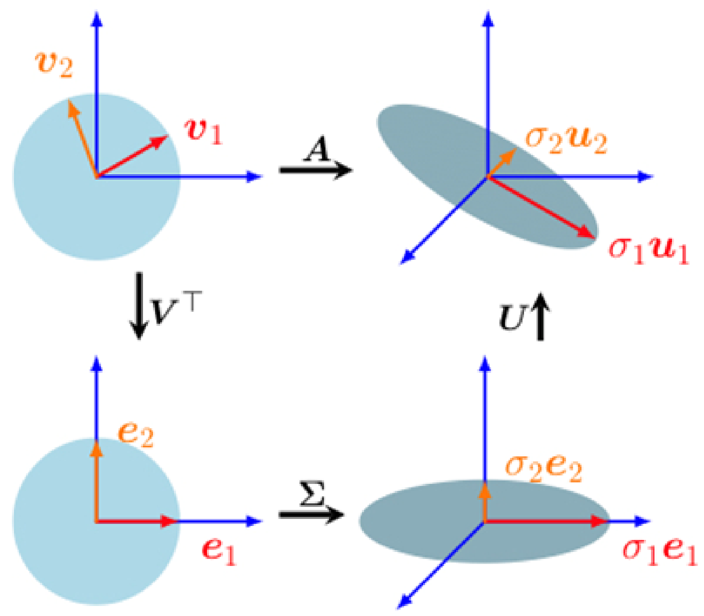
$$
\begin{aligned}
A &= U\Sigma V^T \\
&= \begin{pmatrix} | & | & & | \\ \vec{u}_1 & \vec{u}_2 & \cdots & \vec{u}_m \\ | & | & & | \end{pmatrix}
\begin{pmatrix} \sigma_1 & & & 0 \\ & \sigma_2 & & 0 \\ & & \ddots & 0 \\ & & \sigma_m & 0 \end{pmatrix}
\begin{pmatrix} - & \vec{v}_1^T & - \\ - & \vec{v}_2^T & - \\ & \vdots & \\ - & \vec{v}_n^T & - \end{pmatrix} \\
&= \sigma_1 \vec{u}_1 \vec{v}_1^T + \sigma_2 \vec{u}_2 \vec{v}_2^T + \cdots + \sigma_m \vec{u}_m \vec{v}_m^T
\end{aligned}
$$

우리가 3주차 전반부 symmetric 행렬의 고유값 분해에서 봤던 것처럼, $u_k v_k^T$ 행렬들의 합으로 A를 표현할 수 있다는 것이고 이를 이용하면 임의의 행렬 A에 대해서도 정보량에 따라 여러 layer로 쪼개서 생각할 수 있게 해준다.

기하학적 해석

특이값 분해를 기하학적으로 해석해보자. 모든 행렬은 선형 변환을 의미한다. 행렬은 표준기저벡터의 변환을 의미했고, 또한, 대각 행렬은 기저벡터를 스케일링하는 변환을 의미한다.

$A = U\Sigma V^T$에서 행렬 A의 선형 변환을 해보자.. 먼저 V^T로 표준기저벡터를 돌리고 Σ 로 특이값(σ_i)만큼 스케일링한 뒤, 다시 U로 그 스케일링된 기저벡터를 돌려주는 것을 볼 수 있다. 그림으로 표현하자면 아래와 같다.



즉 정리하면 다음과 같다.

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

Σ : 새로운 기저에서 값 스케일링(Scaling; 크기 변환)

U : 다시 **Codomain(공역)에서** 기저 변환(Basis Change)

이정도까지만 이해해도 괜찮지만, 더 나아가면 이러한 특이값 분해는 다음과 같은 의미를 갖는다고도 요약할 수 있다.

> "직교하는 벡터 집합 $V = (v_1, v_2, \dots)$에 대하여 선형 변환 후에도 그 크기는 변하지만 여전히 직교하게 만드는 그 직교 벡터 집합은 무엇이고, 변경 후의 벡터 집합은 무엇인가?"

잘 와닿지 않겠지만, 아래의 예시를 통해 차근차근 보자.

이해를 위해 2×2 행렬에 대해 생각해보도록 하자. 우리는 2차원 실수 벡터 공간에서 하나의 벡터가 주어지면 언제나 그 벡터에 직교하는 벡터를 찾을 수 있다. 그렇지만 직교하는 두 벡터에 대해 동일한 선형 변환 A를 취해준다고 했을 때, 그 변환 후에도 여전히 직교한다고 보장할 수는 없다.

그렇다면, 직교하는 두 벡터에 대해 동시에 선형 변환을 시켜본다면, 선형 변환 후의 결과가 직교하는 경우를 찾을 수 있을까?

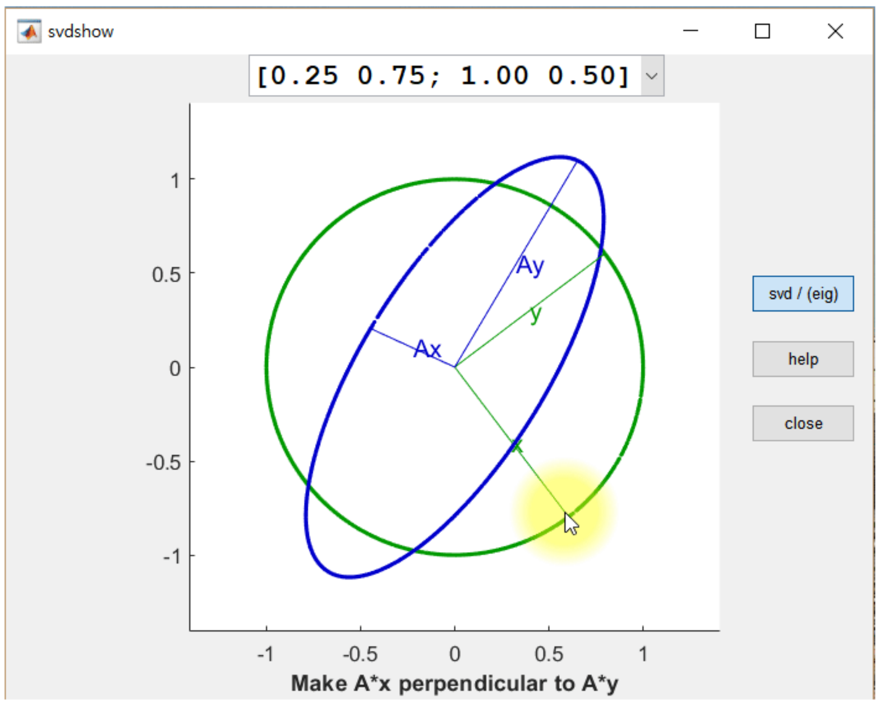


그림 2. 임의의 벡터 x와 x에 직교하는 벡터 y, 그리고 x, y를 선형변환한 결과인 Ax와 Ay

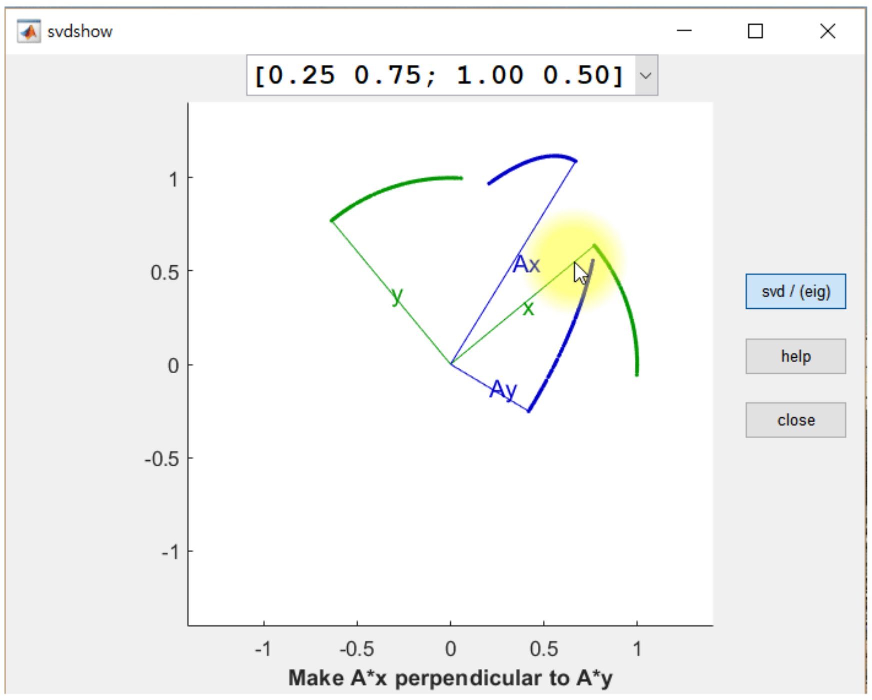


그림 2. 임의의 벡터 x와 x에 직교하는 벡터 y, 그리고 x, y를 선형변환한 결과인 Ax와 Ay

위 그림에서 주목할 것은 크게 두 가지이다.

1. $A\vec{x}, A\vec{y}$가 직교하게 되는 경우는 단 한번만 있는 것이 아님을 확인할 수 있다.
2. $\vec{x}, \vec{y}$ 가 행렬(즉, 선형변환)을 통해 변환되었을 때, 길이가 조금씩 변했다는 것이다. 이 값들을 scaling factor라고 할 수 있지만, 일반적으로는 singular value라고 하고 크기가 큰 값부터 $\sigma_1, \sigma_2, \cdots$ 등으로 부른다.

처음으로 돌아가, 임의의 $m \times n$ 행렬 A에 대해 $A = U\Sigma V^T$ 로 분해할 수 있다고 했다.

위의 예시에서 보여준 선형 변환 전의 직교하는 벡터 $\vec{x}, \vec{y}$는 다음과 같이 열벡터의 모음으로 생각할 수 있으며 이것이$A = U\Sigma V^T$ 에서 V에 해당된다.

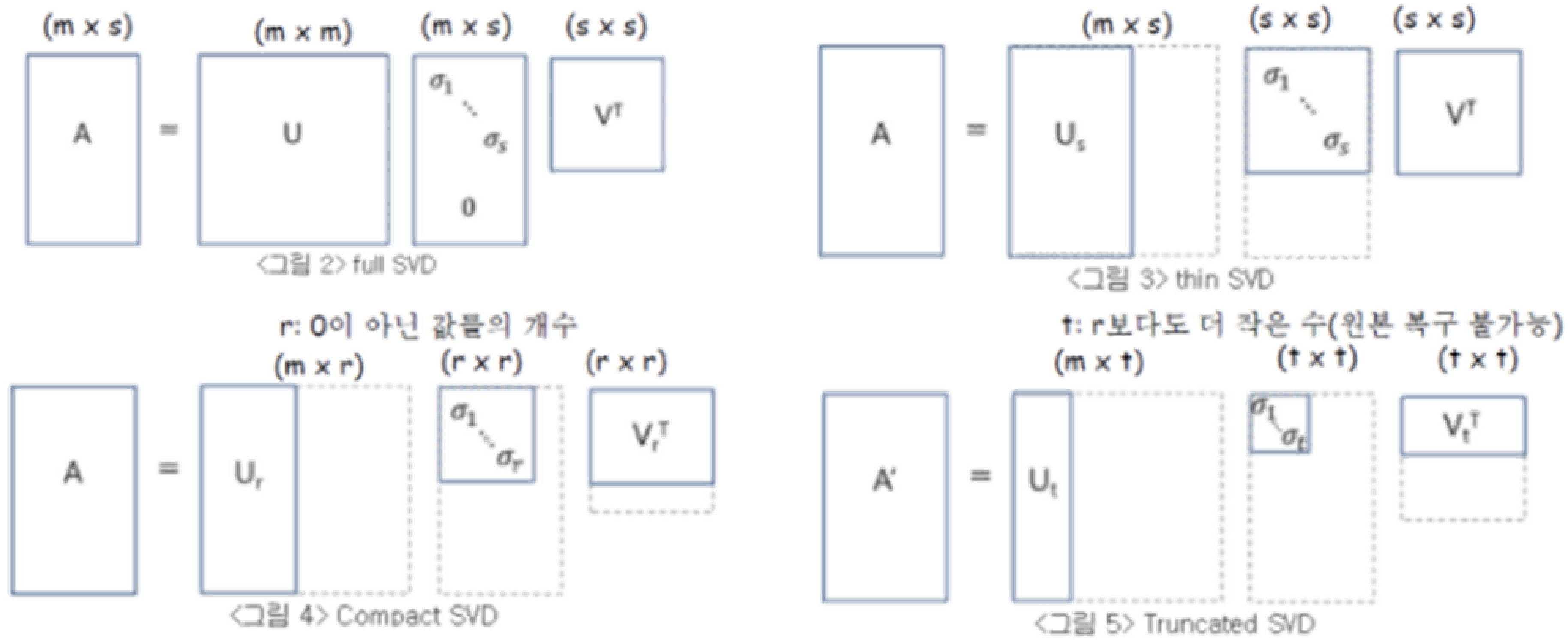
$$V = \begin{pmatrix} | & | \\ \vec{x} & \vec{y} \\ | & | \end{pmatrix} \ / \ U = \begin{pmatrix} | & | \\ \vec{u}_1 & \vec{u}_2 \\ | & | \end{pmatrix} \ / \ \Sigma = \begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix}$$

양변의 우측에 V를 곱해주면, $AV = U\Sigma$ 가 된다. U와 V에 속한 벡터는 서로 직교하는 성질을 가진다고 했다.(symmetric matrix를 고유값 분해한 행렬이므로) 따라서, 서로 직교하는 벡터로 구성된 행렬 V에 선형 변환 A를 해준 뒤에도 서로 직교하는 벡터로 구성된 행렬 U가 만들어질 수 있다. 다만 그 크기가 Σ만큼 스케일링되어있다. 즉

즉, 특이값 분해를 한다는 것은 V에 있는 열벡터($\vec{x}, \vec{y}$)를 행렬 A를 통해 선형변환 할 때(AV), 그 크기는 σ_1, σ_2만큼 변하지만, 여전히 직교하는 벡터들($\vec{u}_1, \vec{u}_2$)을 찾을 수 있는가($U\Sigma$)" 라는 것

특이값 분해의 변형

지금까지 우리가 다룬 특이값 분해는 특이값 분해의 기본적인 개념이며, Full SVD라고 불린다. 하지만 실제로는 축약된 버전인 Reduced SVD를 더 많이 활용하는 편이다.



<그림 2> full SVD
<그림 3> thin SVD
<그림 4> Compact SVD
<그림 5> Truncated SVD

Reduced SVD에는 다음과 같은 종류들이 존재한다.

- **Thin SVD** : Σ 행렬의 아랫부분(비대각 파트)과 U에서 여기에 해당하는 부분을 모두 제거하는 분해. 해당 부분은 연산을 진행해도 항상 0이기 때문에 A를 복원할 수 있다.
- **Compact SVD** : Σ 행렬에서 비대각 파트뿐만 아니라 특이값이 0인 부분도 모두 제거한 형태. 여기에 대응하는 U, V^T의 요소 또한 제거한다. 즉, 특이값이 양수인 부분만 골라낸다는 의미. 이 역시 원본 행렬 A를 복원할 수 있다.
- **Truncated SVD** : Σ 행렬의 특이값 가운데 상위 t개만 골라낸 형태. 이렇게 하면 행렬 A를 완전히 복원할 수는 없지만, 데이터 정보를 상당히 압축하여 행렬 A를 근사할 수 있게 된다. 이렇게 구한 유사 행렬 A'는 원래 행렬 A보다 연산이 더 빠르다는 장점도 있겠지만, 유사 행렬 A'는 데이터의 핵심적인 부분만 사용한다는 특성 때문에 데이터 압축, 불필요한 노이즈 제거 등에 활용될 수 있다. 아래의 이미지 압축이 그 예시이다.

SVD 활용 예제

(1) Truncated SVD 이미지 압축



맨 왼쪽 이미지의 픽셀값을 원소값으로 하는 600 * 367 행렬 A를 잡고 truncated SVD를 이용하여 근사행렬 A'를 구한 후 이를 다시 이미지로 표시하면 다음과 같다.

Full SVD 상태에서는 367개의 특이값을 갖고 있었지만 그 중 100개만을 사용하였더니 사진이 약간 흑백으로 변한 것을 확인할 수 있다. 그러나 사진의 대부분이 원본과 크게 다르지 않다. 반면 20개의 특이값을 사용한 경우, 원본 이미지의 중요한 특성들(창문 밖 여성, 손님과 이발사)은 인식 가능하지만 화질이 떨어진 것을 볼 수 있다.

Truncated SVD의 중요한 포인트는 이미지 데이터에서 중요한 부분(일부 Singular Values)만을 남기고 필요없는 정보를 일부 제거함으로써 정보량을 줄이고 그에 따른 용량과 연산량을 줄이는 것에 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 진행

하게 되면 원본만큼은 아니더라도 원본에 버금가는 이미지로 분석하여 좋은 결과를 더 효율적인 방법으로 계산이 가능하다는 것이다.

(2) 토픽 모델링 - 잠재요인분석(LSA)

토픽 모델링의 목적은 전체 문서의 주제를 연구자가 지정한 개수만큼 압축하여 각 문서들이 어떤 주제를 가지는지 확인하는 것이라고 할 수 있다. 그런 토픽모델링의 시초 모델인 LSA가 어떻게 Truncated SVD를 사용하는지 알아보자.

문장 1: pizza	문장 4: ramen
문장 2: pizza hamburger cookie	문장 5: sushi
문장 3: hamburger	문장 6: ramen sushi

이러한 문서가 있다고 할 때, 단어를 행으로, 문장을 열로 하는 단어-문서행렬 A를 만든 후 특이값분해를 진행하면

	문장1	문장2	문장3	문장4	문장5	문장6
Pizza	1	1	0	0	0	0
Hamburger	0	1	1	0	0	0
Cookie	0	1	0	0	0	0
Ramen	0	0	0	1	0	1
sushi	0	0	0	0	1	1

$A =$

	T1	T2	T3	T4	T5
W1	0.6	0	0	0.7	-0.3
W2	0.6	0	0	-0.7	-0.3
W3	0.5	0	0	0	0.9
W4	0	0.7	-0.7	0	0
W5	0	0.7	0.7	0	0

U (Word Matric for Topic)

	T1	T2	T3	T4	T5	T6
T1	1.9	0	0	0	0	0
T2	0	1.7	0	0	0	0
T3	0	0	1	0	0	0
T4	0	0	0	1	0	0
T5	0	0	0	0	0.5	0

Σ (Topic Strength)

	D1	D2	D3	D4	D5	D6
T1	0.3	0.9	0.3	0	0	0
T2	0	0	0	0.4	0.4	0.8
T3	0	0	0	-0.7	0.7	0
T4	0.7	0	-0.7	0	0	0
T5	-0.6	0.5	-0.6	0	0	0
T6	0	0	0	-0.6	-0.6	0.6

V^T (Document Matrix for Topic)

기존 A 행렬은 (Term X Document)의 크기를 가진 행렬이었는데 LSA를 통해 SVD를 진행한다는 것은 문서와 단어 간의 관계에 어떤 topic이 내재되어있다고 가정하고 이를 특이값 분해를 통해 찾고자 하는 것이다. 따라서 $A = U\Sigma V^T$ 연산은 Term X Document의 관계를 다음과 같이 표현하는 것과 같다.

$$\text{Term} \times \text{Document} = (\text{Term} \times \text{Topic})\ (\text{Topic} \times \text{Topic})(\text{Topic} \times \text{Document})$$

따라서 각 행렬을 확인하는 것으로 단어와 토픽의 관계, 토픽의 영향력(어떤 주제가 이 문서 집합 내에서 중요하게 작용하는 지), 토픽과 문서의 관계를 확인할 수 있다. 이제 이 문서당 하나씩 나타나고 있는 주제들을 압축해보기 위해 여기에서 가장 큰 순으로 2개의 토픽값만을 사용하여 Truncated SVD를 진행해보자.

U

	T1	T2	T3	T4	T5
W1	0.6	0	0	0.7	-0.3
W2	0.6	0	0	-0.7	-0.3
W3	0.5	0	0	0	0.9
W4	0	0.7	-0.7	0	0
W5	0	0.7	0.7	0	0

X

Σ

	T1	T2	T3	T4	T5	T6
T1	1.9	0	0	0	0	0
T2	0	1.7	0	0	0	0
T3	0	0	1	0	0	0
T4	0	0	0	1	0	0
T5	0	0	0	0	0.5	0

X

V^T

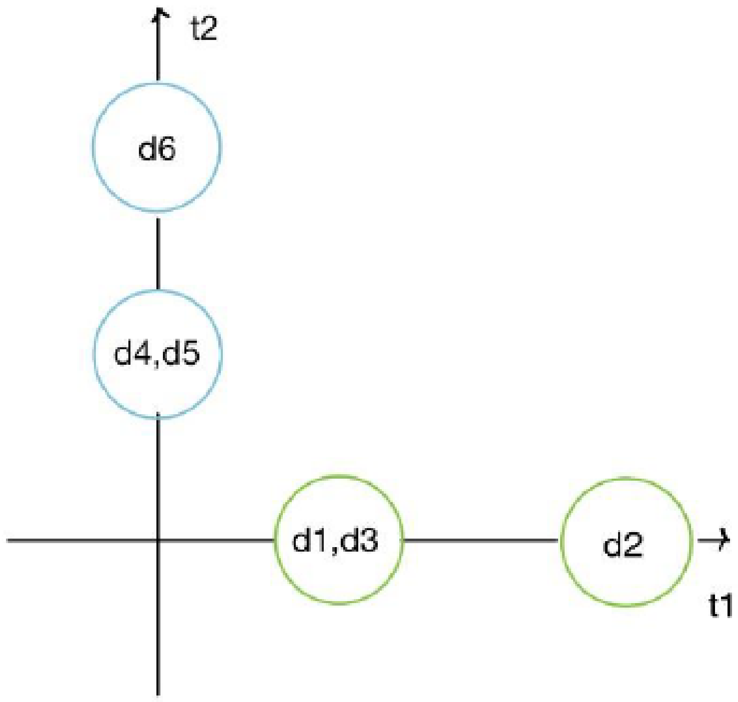
	D1	D2	D3	D4	D5	D6
T1	0.3	0.9	0.3	0	0	0
T2	0	0	0	0.4	0.4	0.8
T3	0	0	0	-0.7	0.7	0
T4	0.7	0	-0.7	0	0	0
T5	-0.6	0.5	-0.6	0	0	0
T6	0	0	0	-0.6	-0.6	0.6

그리고 이렇게 만든 유사행렬 A'가 원래 행렬 A와 얼마나 비슷한지 확인해보면

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \qquad \begin{bmatrix} 0.342 & 1.026 & 0.342 & 0 & 0 & 0 \\ 0.342 & 1.026 & 0.342 & 0 & 0 & 0 \\ 0.285 & 0.855 & 0.285 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.476 & 0.476 & 0.952 \\ 0 & 0 & 0 & 0.476 & 0.476 & 0.952 \end{bmatrix}$$

비교 결과, 오차가 있기는 하지만 A의 경향성을 유지하고 있는 것을 확인할 수 있다.

현재 우리가 알고자 하는 것은 문서와 토픽의 관계(Document Matrix for Topic)이다. 즉 행렬 V^T를 통해 잠재되어 있는 주제가 무엇인지 알 수 있으며, 특히 행을 통해 토픽에 대한 각 단어의 영향력을 확인할 수 있다. 그리고 토픽과 문서 간의 관계를 알아보기 위해 토픽의 영향력과 계산하여 정리하면(ΣV^T)



	D1	D2	D3	D4	D5	D6
T1	0.57	1.71	0.57	0	0	0
T2	0	0	0	0.68	0.68	1.36

다음과 같은 결과를 얻을 수 있다. d1, d2, d3는 t1로 묶여 '양식'을, d4, d5, d6는 t2로 묶여 '일식'을 의미함으로 해석할 수 있을 것이다. 물론 이 예시는 이해를 위해 아주 간단하게 만들어진 것이다. 현재는 LDA, Word2vec 등의 다른 방법들에 밀려 잘 사용되지는 않지만, SVD가 어떻게 응용될 수 있는지 볼 수 있었을 것이다.

이처럼 SVD는 다양한 분야에서 활용이 되고 있다. 추가적으로 의사역행렬(Peseudo-Inverse), 그리고 추천시스템에서 사용자들의 평점 행렬을 분해하여 사용자-아이템 간 상호작용을 표현할 수도 있다. (추천시스템에 대한 내용은 데마팀 클린업 예정)

Additional Topic - Positive Definite(양의 정부호 행렬)

선형대수학이나 최적화를 공부하다보면 Positive Definite, 또는 Positive semi-definite라는 말을 종종 볼 수 있습니다. 저 역시 공부를 하면서 잘 와닿지 않았던 개념이었기 때문에, 추가적인 토픽으로 다루어보고자 합니다. 정의부터 살펴보면,

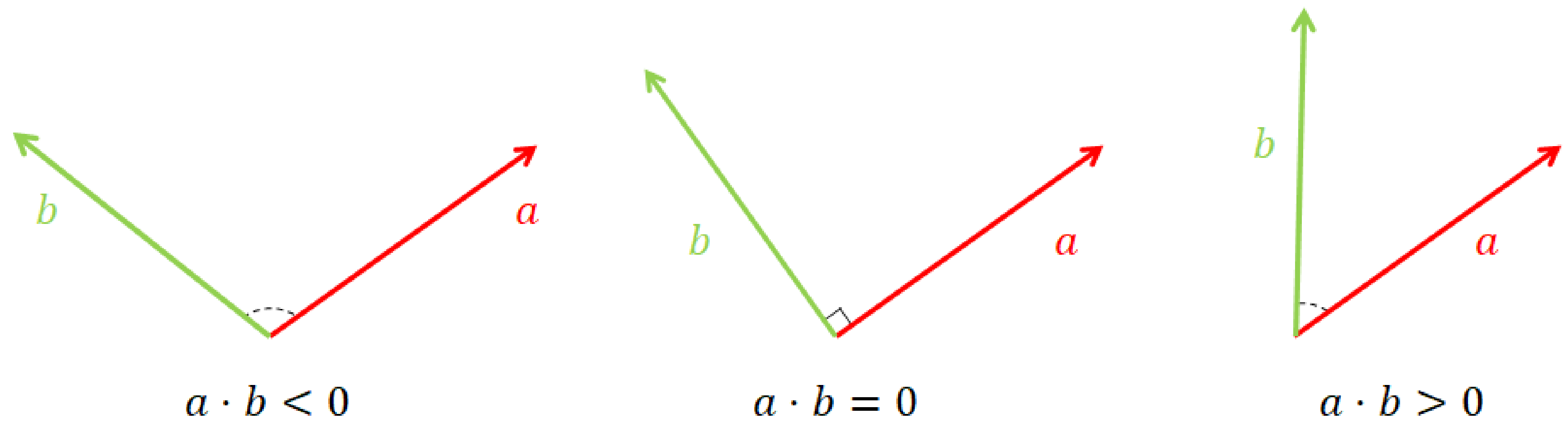
📌 영벡터가 아닌 임의의 열벡터 x와 대칭 행렬 A에 대해 다음이 성립한다면 A는 양의 정부호(positive definite) 행렬이다.

$x^TAx > 0$

라고 되어 있습니다. 저는 처음 접하고 2가지 정도의 질문이 떠올랐던 것 같은데요, 1. 왜 앞뒤에 x^T, x가 붙는 것일까? 2. 그래서 이게 무엇을 의미하고, 왜 갑자기 최적화에서 등장하는 것일까? 차근차근 살펴보며 질문에 답할 수 있도록 해봅시다.

우선 'positive definite' 라는 것은 부호와 관련이 있다. 즉 행렬이 positive definite라면 양수가 작동하는 방식이 그대로 적용되어 작동하는 것과 유사함을 의미한다. c가 양수라면, x에 이를 곱했을 때 x의 부호를 바꾸지 않는 것처럼..$(2 \times -1 = -2, \quad 2 \times 3 = 3)$

그러면 a, b 벡터를 생각해보자. 그리고 이 두 벡터에 대한 내적을 생각해보면, 다음과 같이 계산될 수 있다. $a^Tb = |a||b|\cos\theta$



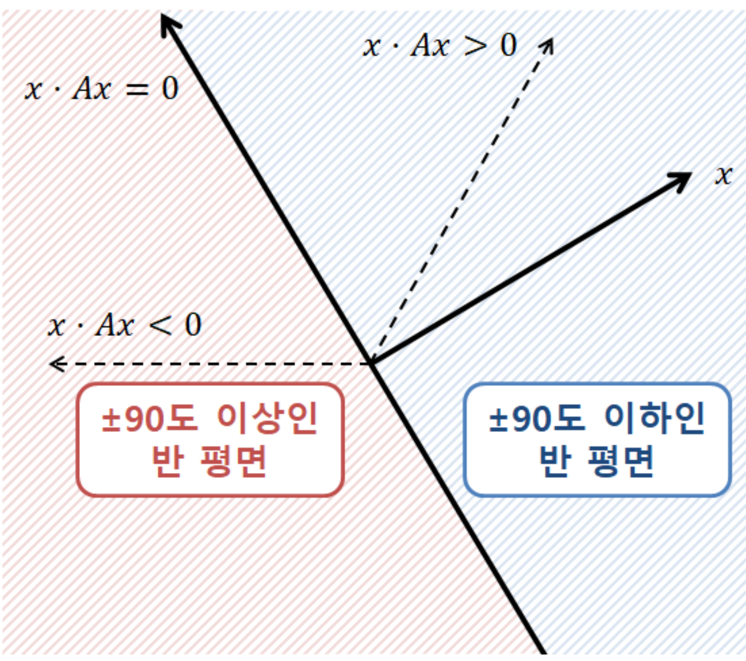
두 벡터의 사잇각 θ가 $-\pi/2 < \theta < \pi/2$ 를 만족한다면, $a^T b > 0$ 이 될 것이다.

이를 A에 의해 선형변환된 벡터 x, 즉 Ax의 경우에서 생각해보면, 영벡터가 아닌 열벡터 x에 대해 다음을 만족해야 임의의 대칭 행렬 A는 양의 정부호 행렬이 된다고 할 수 있을 것이다. 괄호로 묶어본다면 x^T와 x가 각각 어떤 의미인지 생각해볼 수 있을 것 같다.

$$x^T(Ax) > 0$$

즉, 위 식은 임의의 영벡터가 아닌 벡터 x에 대해 선형 변환 A를 취해준 다음 원래의 x와 내적을 취해준 것으로 해석할 수 있다. 앞서 말했던 것 처럼 두 벡터를 내적해서 양의 값이 나오기 위해서는 두 벡터 간의 사잇각이 $\pi/2 < \theta < \pi/2$ 을 만족해야 한다. 그러므로 결국 x에 선형 변환을 시켜줬을 때 변환 전 후의 각도 변화가 -90도에서 90도 사이에서 변하게 된다는 뜻으로 볼 수 있다.

즉, 양의 실수처럼 양의 정부호 행렬을 이용한 선형변환은 입력 벡터를 뒤집어주지는 않는 것이다.



양의 정부호 행렬과 고유값의 부호는 연관성이 깊다.

$$Ax = \lambda x$$
$$x^T Ax = x^T \lambda x = \lambda x^T x = \lambda |x|^2$$

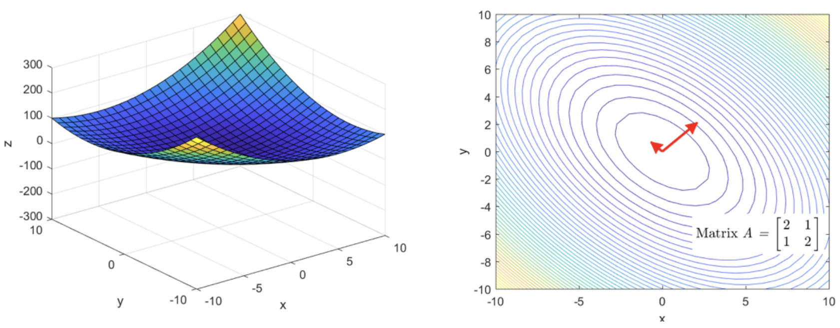
고유값에 대해서는 위 식과 같이 생각해볼 수 있을 것인데, 정의상 $x^T Ax > 0$ 이므로 고유값 역시 모두 양수가 된다는 것이다.

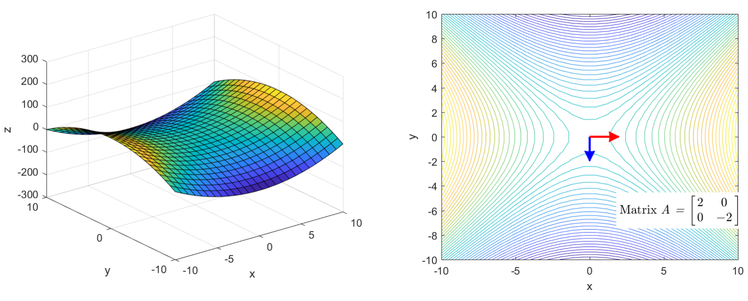
또 고유값의 의미를 다시 한번 생각해보면 고유벡터의 방향으로 얼마만큼 행렬이 변하는지를 보여주는 것인데, 고유값이 양수라는 말은 그 고유벡터의 방향으로 scaling을 수행하기는 하지만 뒤집어지지 않는 변환임을 얘기해준다. 또 고유값이 모두 양수라는 것은 최적화와 연관이 깊다고 할 수 있다.

1주차 후반부에서 헤시안 행렬을 고윳값 분해(Eigen Decompostion) 하여 gradient 변화율이 가장 잘 나타나는 방향(고유벡터)과 크기(고유값)를 찾아 함수의 곡률 방향과 정도를 파악할 수 있다고 했었다.

💡 **Hessian 행렬을 이용한 임계점 판정**

1. 특정 고유벡터에 대해 고유값의 크기가 클 수록 해당방향으로 더 볼록하다.
2. **헤시안 행렬의 고유값이 모두 양수라면 함수는 아래로 볼록하며, 이것이 임계점이라면 극솟값이다.**
3. 헤시안 행렬의 고유값이 모두 음수라면 함수는 위로 볼록할 것이며, 이것이 임계점이라면 극댓값이다.
4. 헤시안 행렬의 교유값에 양수와 음수가 섞여있는 경우라면 함수는 안장의 형태를 갖고, 이것이 임계점이라면 안장점이다.





헤시안 행렬은 대칭행렬이고, 그 헤시안 행렬이 양의 정부호 행렬이라는 것은 헤시안 행렬의 고윳값이 모두 양수라는 것을 의미한다.

다시 말해, 두 번 미분 가능한 함수 f에 대해 헤시안 행렬이 positive definite라면 이 함수의 전체적인 형태는 아래로 볼록하고, 반드시 극솟값을 갖는다는 것을 알 수 있다. 따라서 gradient를 구해 0인 지점을 찾는다면 그 임계점이 바로 극솟값임을 확신할 수 있는 것이다.

positive semi-definite는 $x^T(Ax) \geq 0$ 로, 0이 포함되느냐 아니냐의 차이일 뿐 개념은 동일하고, 이 역시 함수가 아래로 볼록하다고 할 수 있다. 공분산 행렬 역시 positive semi-definite임도 추가로 알아두자!