



수학 2팀

김현우

다들 내용:

2.5. 특이값 분해

2.6. 특이값 분해와 그 응용 : LSA(와 토픽모델링), 협업 필터링 & 협업 필터링 적용(Web)

2.5. 특이값 분해(Singular Value Decomposition, SVD)

우리는 2.3.과 2.4.에서 행렬들을 분해하여 고유값을 찾아내고, 이를 바탕으로 차원 축소(PCA)를 진행하는 방법들에 대해서 알아보았다.

그러나 이런 고유값 분해는 정사각 행렬(Square Matrix)에서만 사용 가능하다는 단점이 있다. 그러나 우리가 마주친 대부분의 데이터 셋들은 대부분 직사각행렬이었다. 또한 데이터셋을 행렬로 표현할 때, 행은 보통 관측된 값일 것이고, 열은 변수를 나타내줄텐데, 관측된 값의 개수와 변수의 개수가 같은 상황은 행렬을 분해하여 정보를 얻기에만 좋지, 나머지 머신러닝 기법을 통해 분석하기에는 좋은 상황은 아닐것이다.

따라서, 일반적인 상황, 즉 직사각 행렬에 대해서도 행렬분해를 할 수 있도록 일반화를 해줘야 할 것이다. 이렇게 고유값 분해를 직사각 행렬에 대해 일반화하여 행렬 분해를 진행해주는 것이 바로 특이값 분해이다.

2.5.1. 특이값 분해 과정

먼저 특이값 분해는 $m \times n$ 행렬 A 에 대해서, 다음과 같이 행렬을 분해한다.

$$A = U\Sigma V^T$$

이렇게만 적어놓으면 기존 고유값 분해와 별반 다른 것이 없어보이지만, U, Σ, V^T 행렬을 어디서 얻어야 될 지 알 수 없다. U, Σ, V^T 행렬을 얻기 위해서는, 우리가 분해하려는 A 를 자기 자신과 내적, 그리고 외적하여 이 값을 고유값 분해하는 과정을 거쳐야 한다.

이때, 우리가 사용할 성질은 다음과 같다.

1. $m \times n$ 행렬을 내적, 혹은 외적한 행렬은 대칭행렬(Symmetric Matrix)이다.
2. 대칭행렬을 고유값 분해하면, 그 고유벡터로 이루어진 행렬은 직교행렬(Orthogonal Matrix)이다.
3. 직교행렬에 대해서, $A^T = A^{-1}$ 이 성립한다. (Properties of Orthogonal Matrix)

다음과 같은 성질에 유의하여, $m \times n$ 행렬 A 를 자기 자신에 대해서 내적, 그리고 외적하고, 이를 고유값 분해하면 다음과 같은 결과를 얻을 수 있다.

(1) 행렬 A 를 자기 자신과 외적

$$\begin{aligned}
 AA^T &= U\Sigma V^T (U\Sigma V^T)^T \\
 &= U\Sigma V^T (V\Sigma^T U^T) \\
 &= U\Sigma V^T V\Sigma^T U^T \\
 &= U\Sigma\Sigma^T U^T \quad (\because V \text{ is orthogonal, } V^T V = I) \\
 &= U\Sigma^2 U^T \quad (\because \Sigma\Sigma^T = \Sigma^2) \\
 &= U\Lambda_U U^T \\
 U &\in \mathbb{R}^{m \times m} \quad (\because AA^T \in \mathbb{R}^{m \times m})
 \end{aligned}$$

(2) 행렬 A 를 자기 자신과 내적

$$\begin{aligned}
 AA^T &= (U\Sigma V^T)^T U\Sigma V^T \\
 &= (V\Sigma^T U^T) U\Sigma V^T \\
 &= V\Sigma U^T U\Sigma^T V^T \\
 &= V\Sigma\Sigma^T V^T \quad (\because V \text{ is orthogonal, } U^T U = I) \\
 &= V\Sigma^2 V^T \quad (\because \Sigma^T \Sigma = \Sigma^2) \\
 &= V\Lambda_V V^T \\
 V^T &\in \mathbb{R}^{n \times n} \quad (\because A^T A \in \mathbb{R}^{n \times n})
 \end{aligned}$$

이와 같이 $m \times n$ 행렬 A 를 특이값 분해 할 때 나오는 행렬 U, V^T 는 행렬 A 를 자기 자신과 외적, 내적하는 것으로 얻어 낼 수 있었고, 이 과정에서 $U \in \mathbb{R}^{m \times m}, V^T \in \mathbb{R}^{n \times n}$ 라는 사실도 알 수 있었다.

이제 좀 더 살펴봐야 할 것은 $m \times m$ 행렬 U 와 $n \times n$ 행렬 V^T 의 중간에 있는 행렬 Σ 이다. 일단, 행렬 U, V^T 를 얻어내는 과정과 U, V^T 의 크기를 통해 행렬 Σ 는 크기가 $m \times n$ 인 행렬이고, $\Sigma \Sigma^T = \Lambda_U$ 이고, $\Sigma^T \Sigma = \Lambda_V$ 라는 사실 또한 알 수 있었다. 따라서, 행렬 분해를 통해 나오는 행렬 Σ 가 모든 특이값들을 표현해주기 위해서는 m 이 클때와 n 이 클 때의 경우를 나누어서 생각해줘야 한다는 것을 알 수 있다. 각 경우에 따른 행렬 Σ 는 다음과 같다.

$$\Sigma = \begin{pmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_s \\ & & & 0 \end{pmatrix} (m > n) \text{ or } \Sigma = \begin{pmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_s & 0 \end{pmatrix} (m < n)$$

이렇게 정의하는 것으로, 모든 행렬(정사각 행렬과 직사각행렬 모두 $A = U \Sigma V^T$ 형태로 분해할 수 있게 된다.

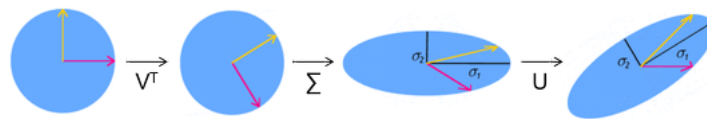
다음으로 넘어가기 전에, 이 Σ 행렬에 대해 좀 더 살펴보자. 먼저 Σ 행렬은 $m = n$ 인 부분은 $\sigma_i, 1 \leq i \leq \min(m, n)$ 를 원소로 가지는 대각행렬의 형태를 띄고 있고, $m \geq n$ 일때는 나머지 행벡터가 $\vec{0}$, $m \leq n$ 일때는 나머지 열벡터가 $\vec{0}$ 인 행렬 형태를 띄고 있다.

또한 $\Sigma \Sigma^T = \Lambda_U$ 이고, $\Sigma^T \Sigma = \Lambda_V$ 에서 확인할 수 있듯이, $(\Lambda_V)^{\frac{1}{2}} = \Sigma$, $(\Lambda_U)^{\frac{1}{2}} = \Sigma$ 이기 때문에, Λ 의 i 번째 주대각 원소 λ_i 에 대해서 $\lambda_i \geq 0$ 이 성립하고, Σ 의 i 번째 주대각 원소 σ_i 에 대해서 $\sigma_i = \sqrt{\lambda_i}$ 가 성립하기 때문에, $\sigma_i \geq 0$ 도 성립한다.

대부분의 경우에, 행렬 Σ 의 원소 σ_i 에 대해서 $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_{\min(m, n)}$ 가 성립하게 특이값분해를 진행한다. 이는 이후 진행할 Truncated SVD 등에서 편의를 위한 것이다.

2.5.2. 특이값 분해의 기하학적 의미

이제 특이값 분해의 기하학적인 특성을 살펴보는 것으로, 특이값 분해를 통해 어떤 정보들을 얻어낼 수 있는지 살펴보고자 한다.



위 그림은 $x \rightarrow Ax$ 의 연산과정을 그림으로 나타낸 것이다. $A = U \Sigma V^T$ 라는 걸 생각한다면, 이 과정은 $x \rightarrow V^T x \rightarrow \Sigma V^T x \rightarrow U \Sigma V^T x (Ax)$ 라고 생각할 수 있을 것이다.

이 때, V^T, U 행렬의 성질에 대해서 생각해보자. V^T, U 는 각각 $A^T A, A A^T$ 를 행렬 분해한 것에서 나왔다. 이 과정에서 V^T, U 는 직교 행렬인 것도 알 수 있었다. 이때, V^T 는 $\mathbb{R}^n \rightarrow \mathbb{R}^n$ 인 Matrix Operator이고, U 또한 $\mathbb{R}^m \rightarrow \mathbb{R}^m$ 인 Matrix Operator인데, $\det(U) = \det(V^T) = 1$ 이 항상 성립하기 때문에 행렬 V^T, U 는 회전변환이다. 따라서, V^T, U 를 곱해주는 것은 어떤 회전을 의미하게 된다.

그렇다면 Σ 행렬은 어떤 것을 의미할까? 위의 그림을 확인해보면, σ_i 의 값에 따라서 공간이 해당 방향으로 수축/팽창하고 있음을 확인할 수 있다. 또한 V^T, U 행렬이 직교행렬이라는 것을 생각해보면, V^T, U 행렬의 열벡터, 행벡터 모두 그 길이가 1이 된다. 즉, A 라는 선형 변환을 할 때, 벡터의 크기는 Σ 행렬에 의해서만 결정되는 것이다.

따라서 x 에 A 라는 선형변환을 하는 과정 $x \rightarrow Ax$ 은 다음과 같이 표현해줄 수 있다.

1. x 를 V^T 방향으로(V 의 반대방향으로) 회전
2. Σ 를 통해 특이값(σ_i)만큼 벡터의 크기를 변경
3. 다시 U 만큼 회전

이렇게, SVD는 행렬 A 로 인해 이루어지는 변환의 과정을 회전과 크기변환으로 설명할 수 있게 해준다.

2.5.3. 여러가지 특이값분해(SVD)의 변형

$$\boxed{A} = \boxed{U} \begin{pmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_s \\ & & & 0 \end{pmatrix} \boxed{V^T}$$

우리가 앞서 살펴본 SVD는 다음과 같은 형태를 띄고 있었다. 행렬 A 를 자기 자신에 대해서 외적하고, 내적하여 이를 고유값 분해하는 것으로 $U\Sigma V^T$ 의 형태로 분해할 수 있었다. 이 과정을 통해 각 행렬 U, Σ, V^T 를 그대로 분해하는 것을 Full SVD라고 한다. (그림은 $m > n$ 일때, 이후 계속 다음과 같은 방식으로 표현 예정)

그러나 보통 행렬 분해를 하게 될 때, Full SVD를 하는 경우는 잘 없다. 보통은 좀더 간략화 된 형태로 SVD를 진행하게 된다. 간략화 된 형태로는 Thin SVD, Compact SVD, Truncated SVD가 있고, 주로 SVD를 사용한다고 하면, Truncated SVD를 말하는 경우가 많다.

(1) Thin SVD

$$A = U_s \begin{bmatrix} \sigma_1 & & & \\ & \ddots & & \\ & & \sigma_s & \\ & & & \ddots \end{bmatrix} V^T$$

Thin SVD는 Σ 의 비대각 부분(0인 행벡터, $n > m$ 일때는 0인 열벡터)와 이에 해당하는 U 행렬의 부분($n > m$ 일때는 V^T 에서 해당하는 부분)을 제거한 형태로 특이값 분해를 진행한다. 해당 부분은 행렬 연산을 진행해도 항상 0이기 때문에 제거해도 $U_s \Sigma_s V^T$ 를 계산해준다면 A 를 원복할 수 있을 것이다.

(2) Compact SVD

$$A = U_r \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_r \end{bmatrix} V_r^T$$

Compact SVD는 Thin SVD에서 더 나아가 Σ 의 비대각 부분 이외에도 주대각 원소 $\sigma_i = 0$ 인 부분과 이에 대응하는 행렬 U, V^T 의 부분도 모두 제거한 형태로 특이값 분해를 진행한다. Thin SVD때와 마찬가지로, $\sigma = 0$ 인 부분은 행렬 연산을 진행해도 항상 0이되기 때문에, 해당 부분을 제거해도 $U_r \Sigma_r V_r^T$ 를 계산해준다면, 행렬 A 를 그대로 얻어낼 수 있을 것이다.

(3) Properties of SVD - Why Truncated SVD would works

(1),(2)에서 해당한 것처럼, 행렬 Σ 의 비대각 부분과 주대각 원소 $\sigma_i = 0$ 인 부분, 그리고 행렬 U, V^T 에서 대응하는 부분도 제거해주어도 똑같이 행렬 A 를 얻어낼 수 있었다. 어떻게 이게 가능할까? 다시한번 Full SVD를 살펴보자. (헷갈리지만 Notation 통일을 위해 그림에서 $N=m, M=n$ 으로 생각하자)

$$A = \begin{bmatrix} | & | & | & & | \\ u_1 & u_2 & u_3 & \cdots & u_M \\ | & | & | & & | \end{bmatrix} \begin{bmatrix} \sigma_1 & 0 & 0 & \cdots & 0 \\ 0 & \sigma_2 & 0 & \cdots & 0 \\ 0 & 0 & \sigma_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \sigma_M \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} \hline v_1^T \\ \hline v_2^T \\ \hline \vdots \\ \hline v_M^T \end{bmatrix}$$

A 벡터는 행렬 U 의 열벡터 u_i ($i = 1, \dots, m$)와 V^T 의 행벡터 v_j^T ($j = 1, \dots, n$)의 곱에 Σ 의 대각성분 σ_k ($k = 1, \dots, \min(m, n)$)을 곱해줬다고 생각할 수 있을 것이다. 이를 수식으로 나타내면 다음과 같이 나타낼 수 있을 것이다.

$$\begin{aligned}
A &= \sum_{i=1}^m \sum_{j=1}^n \sum_{k=1}^{\min(m,n)} \sigma_k u_i v_j^T \\
&= \left(\sum_{k=1}^{\min(m,n)} \sigma_k \right) UV^T \\
&= \sigma_1 UV^T + \sigma_2 UV^T + \dots + \sigma_{\min(m,n)} UV^T
\end{aligned}$$

따라서, 대각성분이 아닌 부분은 연산이 되지 않기 때문에, 이 부분에 해당하는 부분을 제거해줘도 원래 행렬 A 를 구할 수 있을 것이고, 또한 $\sigma_k = 0$ 인 부분을 제거해줘도 원래 행렬을 구할 수 있을 것이다.

여기서 알 수 있는 것은 이뿐만이 아니다. 앞에서 SVD의 기하학적 의미를 살펴보면서 확인했던 것처럼, 행렬 U 와 V^T 는 어떤 회전에 대해서만 작용하지, 실질적인 크기의 변환은 행렬 Σ 에 의해서 이루어진다는 것을 확인하였다. 따라서 $A = (\sum_{k=1}^{\min(m,n)} \sigma_k) UV^T$ 라고 한다면, σ_k 는 행렬 A 를 설명하는 정보량이라고 생각 할 수 있을 것이다. 이는 Truncated SVD의 기본 아이디어가 된다.

(4) Truncated SVD

(3)에서 우리는 σ_k 는 행렬 A 를 설명하는 정보량이라고 접근할 수 있다는 것을 알게되었다. 앞에서 행렬 Σ 의 원소 σ_i 에 대해서 $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_{\min(m,n)}$ 가 되도록 특이값 분해를 진행한다고 말했는데, 그렇다면 σ_k 를 작은 순으로 줄여나갈 수 있을 것이다. σ_k 가 작은 값이라면, σ_k 를 제외해두더라도 원래 A 를 구하지는 못하겠지만, 원래 A 와 매우 비슷한 행렬 A' 를 얻어낼 수 있을 것이기 때문이다.

$$A' = U_t \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_t \end{bmatrix} V_t^T$$

그래서 Truncated SVD는 Σ 행렬의 0인 부분 이외에도, 매우 작은 σ_k 값과 이 부분에 대응하는 행렬 U, V^T 의 부분도 제거하여 $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_t$ 를 주대각 원소로 가지는 대각행렬 Σ_t 만을 사용하여 특이값 분해를 진행한다. 이 경우에는 $U_t \Sigma_t V_t^T$ 를 계산하여도 원래 행렬 A 는 얻어낼 수 없겠지만, 그와 유사한 행렬 A' 를 구할 수 있게 된다.

이렇게 구한 유사 행렬 A' 는 원래 행렬 A 보다 연산이 더 빠르다는 장점이 있겠지만, Truncated SVD로 구한 유사 행렬 A' 는 Matrix Norm $\|A - A'\|$ 를 최소화하는 rank t 행렬이 된다. 이러한 특성 덕분에, 데이터의 핵심적인 특성만 뽑아내어 데이터 압축이나, 불필요한 노이즈 제거 등에 활용될 수 있다.

이를 확인할 수 있는 예제를 살펴보면, SVD에 대한 소개를 마무리하고자 한다. SVD가 데이터의 핵심적인 특성을 뽑아내는데 활용된다고 하였는데, 이를 통해 이미지의 핵심적인 특성만 유지한채로 데이터를 압축할 수 있다.



이런 600×367 픽셀 이미지가 있다고하면, 이 이미지를 행렬로 변환하여 특이값 분해를 실시하면, Full SVD상태에서는 367개의 특이값을 가지고 있을 것이다.



이 중 100개만을 사용하여 Truncated SVD를 진행하여 이를 다시 이미지로 표시하면, 사진이 좀 더 흑백으로 변한 것을 제외하면 거의 원본 사진과 큰 차이가 없다는 것을 확인할 수 있다.



마지막으로 특이값을 20개만을 사용했을 때는 이미지가 다음과 같이 압축된다. 사진의 화질이 다소 떨어지긴 하였지만, 여전히 원본 이미지의 중요한 특성들(창문 밖 여성, 의자에 앉은 손님과 면도를 하는 이발사)등은 충분히 인식할 수 있음을 확인할 수 있다.

2.6. 특이값 분해(SVD)와 그 응용: LSA(Latent Semantic Analysis, 잠재 의미 분석)와 토픽모델링, 행렬분해 기반 협업필터링(Collaborative Filtering)

앞서 말했던 것처럼, SVD, 특히 Truncated SVD를 통해 우리는 데이터에서 중요한 특성을 뽑아낼 수 있게 되었다. 이를 바탕으로 다양한 데이터 분석을 진행할 수 있게 되었는데, 이 파트에서는 특이값 분해를 바탕으로 하는 분석인 토픽모델링 기법인 LSA와 특이값 분해를 기반으로 한 협업필터링, 그리고 협업필터링을 어떻게 시스템으로 적용할 수 있을지 웹에 대한 간단한 소개와 함께 진행하고자 한다.

▼ Intuition Point 1. Sequential하지 않은 텍스트 분석 :Bag of Words & TF-IDF

지난 스터디시간에 RNN과 LSTM, 그리고 Self-Attention을 활용한 Transformer와 다양한 활용 모델을 공부하였는데, 해당 모델들에서 가장 중요했던 것은 바로 Sequential한 Data를 다룬다는 것이었다. 그래서 해당 모델들에서는 Data의 위치를 표현하기 위해서 Input을 순차적으로 넣어주거나, 단어에 위치에 대한 정보를 encoding하여 합치기도 하였다.

텍스트 데이터는 기본적으로 Sequential한 데이터이다. 텍스트 데이터로 어떤 문장이 주어져도 문장의 위치에 따라 그 의미가 달라지기도 하고, 문법상 어떤 단어가 들어와야 하는지 달라지게 된다. 따라서 텍스트 데이터를 완벽하게 분석하기 위해서는, 이런 위치에 대한 정보까지 고려해줘야 한다.

그러나 이 파트의 제목처럼, 텍스트 데이터를 Sequential하지 않다고 가정하면 어떻게 될까? 물론 기존의 RNN이나 Transformer와 그 활용 모델들처럼 기계번역이나 감성분석, 텍스트 생성 등의 분석들은 불가능하겠지만 텍스트 분류나, 문서를 주제에 따라 묶어주는

토픽모델링(Topic Modeling)은 충분히 가능할 것이다.

따라서 단어가 Sequential하지 않다고 가정하는 것으로도 충분히 괜찮은 분석을 진행할 수 있게 된다. 그렇다면 어떻게 단어가 Sequential하지 않다고 가정해야 할까? 이를 위해서는 2가지 가정이 필요할 것이다.

1. 문장에서 단어는 순서에 상관없이 항상 독립적으로 나타난다.
2. 문장에서 단어의 순서는 의미가 없고, 단어가 나타난 횟수가 중요하다.

이 가정을 토픽모델링에 맞게 바꿔주면 다음과 같이도 표현할 수 있을 것이다.

1. 주제가 비슷한 문장이라면 단어 빈도 역시 비슷할 것이다.
2. 많이 쓰인 단어가 주제와 강한 관계가 있을 것이다.

이런 가정을 마치 단어들이 들어있는 가방에서 단어를 뽑아서 어떤 단어가 나타났는지가 중요하다는 의미로 Bag of Words 가정이라고도 하며, 이 가정에 기반한 텍스트 분석은 RNN등의 모델들이 주요하게 사용되기 이전에 텍스트 분석을 대표하기도 하였다.

IP 1.1. Bag of Words & DTM

그렇다면 실제로 Bag of Words 가정을 바탕으로 텍스트를 한번 분석해보자. 다음과 같은 문장이 주어졌다고 가정하자.

정부가 발표하는 물가상승률과 소비자가 느끼는 물가상승률은 다르다.

이 텍스트 데이터를 처리하기 위해서는 다음과 같은 전처리 과정이 필요할 것이다(텍스트 전처리에 대한 자세한 내용은 여기서 다루지 않는다.)

1. 문장을 단어 단위(영어)/형태소 단위(한글)로 토큰화하여 말뭉치(Corpus) 생성
2. 각 단어에 고유한 정수 인덱스 부여(One-hot Encoding 등)
3. 각 단어가 등장한 횟수(Frequency)를 세서 이를 벡터화 시킴.

주어진 문장에 3번 과정까지 처리를 해주면, 다음과 같이 나타낼 수 있을 것이다.

	정부	가	발표	하는	물가 상승률	과	소비
문장 1	1	2	1	1	2	1	1

이를 바탕으로 '물가상승률'이라는 단어가 많이 나타났기 때문에, 문장 1은 '물가상승률'에 대한 내용임을 쉽게 짐작할 수 있다. 이렇게 문장을 단어와 문서, 그리고 단어가 나타난 횟수를 바탕으로 벡터화시킬 수 있었다.

그렇다면 여러가지 문장/문서가 나타날때는 어떻게 표현해줘야 할까? 이때도 지금 문장 1을 처리했던 것 처럼, 각 문장/문서를 벡터화 시키고, 이를 합쳐 행렬의 형태로 표현해주면 될 것이다.

다음과 같은 문서가 4개 있다고 해보자

- 문서 1: 먹고 싶은 사과
- 문서 2: 먹고 싶은 바나나
- 문서 3: 길고 노란 바나나 바나나
- 문서 4: 저는 과일이 좋아요

그렇다면 이 문서들 또한 다음과 같이 행렬로 표현 가능할 것이다.

	과일이	길고	노란	먹고	바나나	사과	싶은
문서 1	0	0	0	1	0	1	1
문서 2	0	0	0	1	1	0	1
문서 3	0	1	1	0	2	0	0
문서 4	1	0	0	0	0	0	0

이렇게, 단어가 나타난 횟수를 행렬로 표현한 것을 문서 단어 행렬(Document-Term Matrix, DTM)이라고 한다. 이를 문서 단어 행렬이라고 하는 이유는 각 행은 문서에 대하여, 각 열은 단어에 대한 것이기 때문이다. 이와 반대로 각 행은 단어에 대하여, 각 열은 문서에 대하여 행렬을 만든다면, 이를 단어 문서 행렬(Term-Document Matrix, TDM)이라고 한다.

보통 토픽모델링에서는 DTM을 가장 많이사용하기는 하지만, 경우에 따라 TDM을 사용하기도 한다. 그러나 DTM과 TDM은 전치관계이기 때문에, 행렬에서 각 행과 각 열이 나타내는 것이 무엇인지만 신경쓴다면, 둘 중 어느 것을 사용하여도 상관없다.

DTM의 경우에도 아까 문장에 대해서 분석을 진행했던 것처럼 분석을 진행해주면 된다. 그러나 이렇게 단순히 문장/문서에서 단어가 출현한 빈도만을 가지고 분석을 하는 것에는 다음과 같은 문제가 있다.

1. '싫은' 등과 같은 의미 없는 단어(불용어)가 들어갈 때, 자칫 이 단어를 중요한 단어라고 인식할 수 있다.
2. 중요한 것처럼 보이는 단어들도 여러 문서에서 많이 나온다면, 이 단어가 중요한 단어가 아닐 수도 있다.

1.번 같은 경우에는 전처리 과정에서 전체 문서에 들어있는 해당 단어를 제거(불용어 처리)를 하는 것으로 해결 가능하지만, 2번과 같은 경우에는 단순히 단어의 빈도수만 가지고는 알 수 없다. 따라서, 단순히 DTM만 가지고 빈도 분석하는 것으로는 아직 충분한 인사이트를 얻기에 부족하다.

IP 1.2. TF-IDF(단어 빈도-역 문서 빈도, Term Frequency-Inverse Document Frequency)

IP 1.1.에서 확인한 DTM의 문제 2번과 같은 상황을 해결하기 위해서는 각 문서에서 고유하게 나타나는 단어는 크게, 다른 문서에서도 많이 나올때는 단어는 작게 만들어줘야 한다. 이를 위해서 DTM에 어떤 가중치를 매겨주는 방법을 생각해볼 수 있는데, 이 방법이 바로 TF-IDF이다.

DTM에 매기는 가중치 TF-IDF는 다음과 같이 수식으로 표현해줄 수 있다.

$$TF - IDF = TF \times \log \frac{n_D}{1 + n_t}$$

TF : 하나의 문서 내에서 단어 t 가 나온 빈도수
 n_D : 전체 문서 수 n_t : 단어 t 가 나온 문서 수

이때, TF의 경우에는 단어 t 가 나온 빈도수, 즉 DTM에서 우리가 행렬로 표시해주었던 단어들의 등장 횟수이고, n_D 는 DTM의 행의 길이, n_t 은 단어 t 가 나온 문서 수이다.

$$IDF = \log \frac{n_D}{1 + n_t}$$

n_D : 전체 문서 수 n_t : 단어 t 가 나온 문서 수

이때, IDF를 좀 더 살펴보자면, 일단 먼저 전체 식에 \log 가 씌워져 있음을 확인할 수 있다. 만약 \log 를 씌우지 않았다면, n_D 가 커질수록 전체 값이 매우 기하급수적으로 커지기 때문에, \log 를 씌우는 것으로 전체 가중치 값들이 매우 커지는 것을 방지한다.

또한 단순히 $\frac{n_D}{n_t}$ 가 아닌 $1 + n_t$ 를 분모로 사용한 것을 확인할 수 있는데, 이는 어떤 단어가 전체 문장에서 한번도 나오지 않았을 때, IDF의 분모가 0이 되는 것을 막아준다.(이와 비슷한 작업을 뒤에서 토글로 가볍게 언급할 나이브 베이즈(Naive Bayes)에서 어떤 단어가 한번도 등장하지 않았을 때 전체 확률이 0이 되는 것을 막아주기 위해 라플라스 평활법(Laplace Smoothing)을 적용하는 것과 비슷한 맥락에서 생각할 수 있다.(라플라스 평활법에 대한 내용은 다루지 않을 예정))

그렇다면 TF-IDF에 대하여 알아보았으니, 이를 한번 실제로 적용해보자. 예시로 IP 1.1.에서 사용한 4개의 문서 예시를 다시 한번 사용하자. 4개 문서의 DTM은 다음과 같았다.

	과일이	길고	노란	먹고	바나나	사과	싫은
문서 1	0	0	0	1	0	1	1
문서 2	0	0	0	1	1	0	1
문서 3	0	1	1	0	2	0	0
문서 4	1	0	0	0	0	0	0

아까 언급했듯이, TF-IDF에서는 TF는 DTM 행렬의 각 원소이다. 따라서, TF-IDF 가중치를 매기기 위해서는 IDF만 계산하여 IDF를 이 행렬의 각 해당하는 원소에 곱해주면 된다.

각 단어의 IDF는 다음과 같이 계산된다.

단어	IDF(역 문서 빈도)
과일이	$\ln(4/(1+1)) = 0.693147$
길고	$\ln(4/(1+1)) = 0.693147$
노란	$\ln(4/(1+1)) = 0.693147$
먹고	$\ln(4/(2+1)) = 0.287682$
바나나	$\ln(4/(2+1)) = 0.287682$
사과	$\ln(4/(1+1)) = 0.693147$
싫은	$\ln(4/(2+1)) = 0.287682$
저는	$\ln(4/(1+1)) = 0.693147$

단어	IDF(역 문서 빈도)
좋아요	$\ln(4/(1+1)) = 0.69314$

이를 통해 TF-IDF 가중치를 적용한 TF-IDF는 다음과 같이 구할 수 있다.

	과일이	길고	노란	먹고	바나나	사과	싫은
문서1	0	0	0	0.287	0	0.6931	0.28
문서2	0	0	0	0.287	0.2876	0	0.28
문서3	0	0.6931	0.6931	0	0.5753	0	0
문서4	0.6931	0	0	0	0	0	0

TF-IDF 행렬과 기존 DTM 행렬을 비교해보면, 가중치를 적용한 TF-IDF행렬이 좀 더 각 문서에서 중요한 단어가 뭔지 쉽게 확인할 수 있었다. 그러나 TF-IDF에도 다음과 같은 한계가 존재한다.

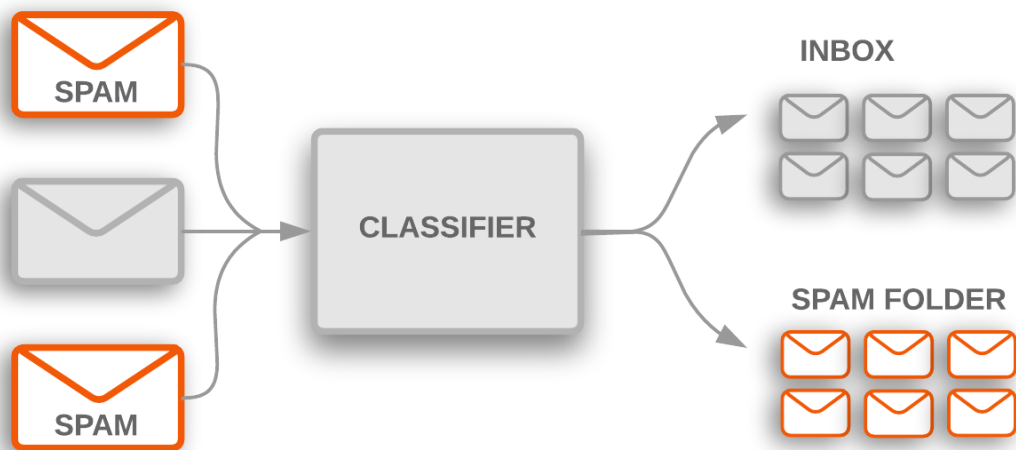
1. 결국 우리가 토픽 모델링을 진행하는 이유는, 각 문서에 대한 대략적인 주제를 알기 위함이 아닌 각 문서를 몇개의 주제로 압축시켜 각 문서들이 대략적으로 어떤 주제에 속하는지 알기 위함인데, TF-IDF만으로는 각 문서들이 어떤 주제를 가지는지 단어를 가지고 판단하고, 이를 묶어주는 과정도 직접 수행해줘야 한다. 토픽 모델링에서 압축된 주제가 어떤 주제인지 판단하는 것에 연구자의 주관이 들어가지만, TF-IDF는 압축된 주제를 정하는 것에도 각 문서가 어떤 주제를 가지는지 주관적으로 판단해줘서 연구자의 주관에 따라 분석 결과에 큰 차이가 날 수 있다.
2. 각 단어가 다른 문서에서 같은 횟수로 언급되고, 문서 내에서도 같은 횟수로 언급된다면, TF-IDF를 사용해도 어떤 단어가 중요한 단어인지 알 수 없다.(예제의 문서 2, 문서 4의 경우)

따라서, Word of Bag 가정 하에 DTM에서 더 많은 인사이트를 도출하기 위해서는 다른 토픽모델링 기법을 활용해줘야 할 것이다. 이런 흐름에서 거의 처음으로 등장한 토픽 모델링 기법이 바로 다음에 다루게 될 LSA(Latent Semantic Analysis, 잠재 의미 분석)이다.

▼ +나이프 베이즈(Naive-Bayes)

Intuition Point 1과 2.6.1.에서 다뤘고 또 다루게 될 Bag of Words 가정에 기반한 토픽 모델링 기법은 모두 비지도 학습이다. 그러나 Bag of Words 가정에 기반한 텍스트 분석이 모두 토픽모델링을 위시로한 비지도학습으로 이루어지는 것은 아니다. 대표적으로 Naive-Bayes 모델은 텍스트 데이터만을 처리하기 위한 모델은 아니지만, 특히 스팸 메일 분류 같이 Bag of Words 가정에 기반한 텍스트 데이터 처리에 탁월한 성능을 보이고 있다.

Naive-Bayes는 이전 스타터들에서 다루었던 Intuition Point들이나 EM Algorithm에서 다루었던 Generative Model에 속하는 지도학습 분류모델이다. Naive-Bayes에 대해서 알아보기 전에, 스팸 메일 분류 예시를 통해 이를 알아보자.



우리는 메일에서 스팸메일을 받았을 때, 이를 자동으로 분류하는 모델을 생성하고자 한다. 이를 위해서, 다음과 같은 데이터를 학습한다고 하자.

	스팸여부(y)	강의(x_1)	광고(x_2)	...	보험(x_{5714})	
이메일 1	1(스팸)	0	1	...	1	...

	스팸여부(y)	강의(x_1)	광고(x_2)	...	보험(x_{5714})	
이메일 2	0(정상)	1	0	...	1	...
⋮	⋮	⋮	⋮	...	⋮	...
이메일 m	0	1	1	...	0	...

m 개의 이메일이 주어지고, 이 이메일에는 스팸 여부(y)가 0과 1로 label되어있다. 우리는 이메일에서 많이 등장하는 단어 10000개를 뽑아 단어에 인덱싱을 해주고, 단어의 출현여부에 따라 단어가 등장하면 1, 등장하지 않으면 0의 값을 줬다고 하자. 그렇다면 인덱스 번호가 j 인 단어의 출현 여부는 다음과 같이 수식으로 표현해줄 수 있을 것이다.

$$x_j = I(j\text{번째 단어가 이메일에서 나타날 때})$$

이제 이 데이터를 어떻게 분류할 것인지 생각해보자. 우리의 목표는 어떤 단어들이 등장할 때, 그 단어가 스팸메일인지 아닌지를 분류하는 것이다. 이를 위해서 만약 어떤 단어들이 등장했을 때 그 단어가 스팸메일일 확률과 아닐 확률을 구해서 스팸메일일 확률이 높다면 스팸메일로 분류하고, 스팸메일이 아닐 확률이 더 높다면 정상 메일로 분류하는 것이다.

이는 다음과 같이 수식으로 표현해줄 수 있을 것이다.

$$P(y|X) = E[I(y|X)], \text{ where } X = [x_1, x_2, \dots, x_{10000}]$$

물론 이를 Logistic Regression등으로 분류 할 수도 있겠지만, 현재 설명변수의 개수가 너무 많기 때문에(10000개) 이를 예측하기 위해서는 매우 많은 파라미터가 필요할 것이기 때문이다. 이렇게 직접 $P(y|X)$ 를 추론하는 방법으로는 효과적으로 스팸메일을 분류하기 어려울 것이다. 따라서, 이 문제를 Generative하게 접근한다면 좀 더 쉽고 효과적으로 분류가 가능할것이다. $P(y|X)$ 는 다음과 같이 표현할 수 있었다.

$$P(y|X) = \frac{P(y)P(X|y)}{P(X)} = \frac{P(X, y)}{P(X)}$$

이렇게 접근한다면, 우리는 쉽게 다음과 같이 $P(y|X)$ 를 최대화하는 것으로 모델을 학습시켜 분류를 진행할 수 있을 것이다.

$$\begin{aligned} \operatorname{argmax}_y P(y|X) &= \operatorname{argmax}_y P(y)P(X|y) = \operatorname{argmax}_y P(X, y) \\ (\because P(X) &= \text{constant}) \end{aligned}$$

그러나 여기에는 문제가 있다. $X = [x_1, x_2, \dots, x_{10000}]$ 였기 때문에, 베이즈 정리에 의해 $P(X|y)$ 는 다음과 같이 표현될 것이기 때문이다.

$$P(X, y) = P(y)P(x_1|y)P(x_2|y, x_1) \cdots P(x_{10000}|x_{9999}, \dots, x_2, x_1, y)$$

이를 계산하는 것은 아까 말했던 Logistic Regression으로 이 이메일들을 분류하는 것처럼 효과적이지 못하고, 매우 복잡한 연산 과정을 요구할 것이다. 그러나 이를 다음과 같이 가정한다면 어떨까?

$$\begin{aligned} P(X, y) &= P(y)P(x_1|y)P(x_2|y) \cdots P(x_{10000}|y) \\ &= P(y) \prod_{j=1}^{10000} P(x_j|y) \end{aligned}$$

이렇게 가정하는 것으로, 기존의 매우 복잡한 연산을 비교적 간단하게 바꿀 수 있었다. 이렇게 가정하는 과정도 간단하다. 단순히 y 가 주어졌을 때, x_j 끼리는 독립이라는 조건부 독립을 가정하면 되기 때문이다. 그렇다면 다시 돌아와서, Bag of Words를 다시 한번 생각해보자. Bag of Words 가정은 다음과 같았다.

1. 문장에서 단어는 순서에 상관없이 항상 독립적으로 나타난다.
2. 문장에서 단어의 순서는 의미가 없고, 단어가 나타난 횟수(여기서는 단어의 등장 여부)가 중요하다.

결국 조건부 독립을 가정하는 것은, 같은 라벨을 가진 데이터끼리 Word of Bag 가정을 하는 것과 같음을 확인할 수 있다. 나이브 베이즈 모델은 이런 조건부 독립 가정을 통해 베이즈 법칙을 간단하게 바꾸어서 해결하기 때문에, Naive(순진한)이름이 모델 이름 앞에 붙게 된 것이다.

이렇게 조건부 독립 가정을 하게 되면 이후 과정은 매우 간단하다. 어떤 이메일이 스팸메일일지 아닐지에 대한 확률($P(y)$)과 스팸 메일일 때 어떤 단어 j 가 등장할 확률($P(x_j = 1|y = 1)$), 스팸메일이 아닐 때 어떤 단어 j 가 등장할 확률($P(x_j = 1|y = 0)$)에 대해서 가정해주자, 이때, 모든 데이터가 0혹은 1로 나타나기 때문에, 베르누이 분포로 가정해주면 된다. 이를 수식으로 표현해 주면 다음과 같이 표현해줄 수 있다.

$$\begin{aligned}
\phi_y &= P(y = 1) \implies y \sim \text{bernoulli}(\phi_y) \\
\phi_{j|y=1} &= P(x_j = 1 | y = 1) \\
\implies (x_j | y = 1) &\sim \text{bernoulli}(\phi_{j|y=1}) \\
\phi_{j|y=0} &= P(x_j = 1 | y = 0) \\
\implies (x_j | y = 0) &\sim \text{bernoulli}(\phi_{j|y=0})
\end{aligned}$$

이를 바탕으로, 다음과 같은 Joint Likelihood Function을 최대화하는 $\phi_y, \phi_{j|y=1}, \phi_{j|y=0}$ 을 구하는 것으로 모델을 학습시킬 수 있다.

$$\begin{aligned}
&\text{argmax}_{\phi_y, \phi_{j|y=1}, \phi_{j|y=0}} L(\phi_y, \phi_{j|y=1}, \phi_{j|y=0}) \\
&= \text{argmax}_{\phi_y, \phi_{j|y=1}, \phi_{j|y=0}} \prod_{i=1}^m P(X^{(i)}, y^{(i)}, \phi_y, \phi_{j|y=1}, \phi_{j|y=0})
\end{aligned}$$

매우 복잡해보이지만, MLE를 통해 추정된 각 파라미터 $\phi_y, \phi_{j|y=1}, \phi_{j|y=0}$ 의 추정량은 간단하게 나온다. 각 파라미터의 MLE 추정량은 다음과 같다.

$$\begin{aligned}
\hat{\phi}_y^{MLE} &= \frac{\sum_{i=1}^m I(y^{(i)} = 1)}{m} \\
\hat{\phi}_{j|y=1}^{MLE} &= \frac{\sum_{i=1}^m I(x_j^{(i)} = 1, y^{(i)} = 1)}{\sum_{i=1}^m I(y^{(i)} = 1)} \\
\hat{\phi}_{j|y=0}^{MLE} &= \frac{\sum_{i=1}^m I(x_j^{(i)} = 1, y^{(i)} = 0)}{\sum_{i=1}^m I(y^{(i)} = 0)}
\end{aligned}$$

식만 복잡잡해 보일뿐, 식을 자세히 들여다보면 각 파라미터의 MLE 추정량은

1. $\phi_y \rightarrow$ 전체 문서에서 스팸메일의 비율
2. $\phi_{j|y=1} \rightarrow$ 스팸메일 중에서 해당 단어 j 가 등장한 비율
3. $\phi_{j|y=0} \rightarrow$ 정상메일 중에서 해당 단어 j 가 등장한 비율

이와 같이 각 파라미터의 추정치가 단순히 ①전체 이메일 수, ②스팸메일 수, ③스팸메일에서 해당 단어 j 가 등장한 이메일 수, ④정상메일에서 해당 단어 j 가 등장한 이메일 수에만 의존하고, 이는 단순히 숫자만 세어주면 되기 때문에 모델의 학습속도도 빠르고, 적용하기도 쉽다는 장점이 있다.

이후, 만약 10000개의 단어중 전체 이메일에서 하나라도 등장하지 않는 단어가 있을 때는, Laplace Smoothing이라는 기법을 적용하여 전체 확률이 0이 되는것을 방지해준다. 이는 Reference를 참고하면 되겠다.

Smoothing 기법: Laplace(add-one) smoothing, Back-off smoothing

통계적 언어 모델 과 N-gram 언어모델 은 희소 문제가 발생하며, 이러한 문제를 해결하기 위한 generalization기법으로 smoothing 기법이 있다. Smoothing기법 중 대표적인 laplace smoothing과 back-off smoothi..

🔗 <https://daanv.tistory.com/28>



이렇게, Words of Bags 가정을 통해서 비지도학습은 물론이고, 지도학습으로 텍스트 데이터가 Sequential하지 않다는 가정에도 충분히 효과적인 분석을 진행할 수 있음을 확인하였다.

2.6.1. LSA(Latent Semantic Analysis, 잠재 의미 분석)와 토픽모델링

Intuition Point 1에서는 Bag of Words 가정을 통해 텍스트 데이터를 Sequential하지 않게 분석할 수 있는지, 그리고 그 분석 방법의 한계는 무엇인지 확인해보았다. 이 파트에서는 토픽 모델링의 시초인 LSA에서 시작하여 최근 토픽 모델링에서 가장 많이 사용되고 있는 LDA 까지 한번 알아보는 시간을 가져보도록 하자.

2.6.1.1. 잠재 의미 분석(Latent Semantic Analysis, LSA)

토픽 모델링의 목적은 전체 문서의 주제를 연구가 지정한 개수만큼 압축하여 각 문서들이 어떤 주제를 가지는지 확인하기 위함이다. LSA는 토픽 모델링의 시초인 모델이고, 문서들의 추상적인 주제를 발견하고, 텍스트 본문에 숨겨진 의미를 찾기 위해서 앞서 보았던 특이값분해를 사용하였다.

앞서 확인하였던 것처럼 SVD에서 행렬 A 를 특이값분해하여 얻은 행렬 Σ 의 대각 원소 σ_k 는 행렬 A 를 설명하는 정보량 내지는 중요도이다. 또한, 이를 압축하여도(Truncated SVD) 충분히 원래 행렬에 근사한 행렬 A' 를 얻을 수 있음을 알 수 있었다.

앞서 Intuition Point 1을 통해서 확인했듯이, 우리는 문서의 집합을 행렬의 형태(DTM)으로 변환할 수 있고, 또한 단어의 중요도에 따른 가중치를 매긴 형태(TF-IDF 행렬)로도 변환할 수 있음을 확인하였다.

따라서, 이렇게 행렬의 형태로 표현된 문서의 집합을 특이값 분해하고, 이 중 영향력있는 t 개의 특이값만 사용한다면, 전체 문서 집합을 최대한 잘 설명하는 행렬 A' 를 얻어낼 수 있고, 앞서 말했듯이 이 행렬은 Matrix Norm $\|A - A'\|$ 를 최소화하는 $rank\ t$ 행렬이기 때문에, 효과적으로 각 문서들의 주제를 효과적으로 t 개의 주제로 압축할 수 있게 된다.

그렇다면 예시를 통해 어떻게 LSA가 진행되는지 알아보자. 다음과 같은 문서가 있고, 이를 DTM으로 변환한다.(만약 TF-IDF 행렬을 구했다면, TF-IDF 행렬을 사용해도 된다.) 그렇다면 다음과 같이 나타나게 될 것이다.

문서 1: 나는 학교에 간다
문서 2: 학교에 가는 영화
문서 3: 나는 영화는 좋다.

DTM

	나	는	학교	에	가	ㄴ	다
문서 1	1	1	1	1	1	1	1
문서 2	0	1	1	1	1	0	0
문서 3	1	2	0	0	0	0	1

이 DTM을 Compact SVD로 특이값 분해하면 다음과 같은 결과를 얻을 수 있다.

$$A = U_r \Sigma_r V_r^T$$

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

$$U_r = \begin{bmatrix} -0.62 & 0.47 & 0.63 \\ -0.47 & 0.41 & -0.78 \\ -0.62 & -0.77 & -0.03 \end{bmatrix}$$

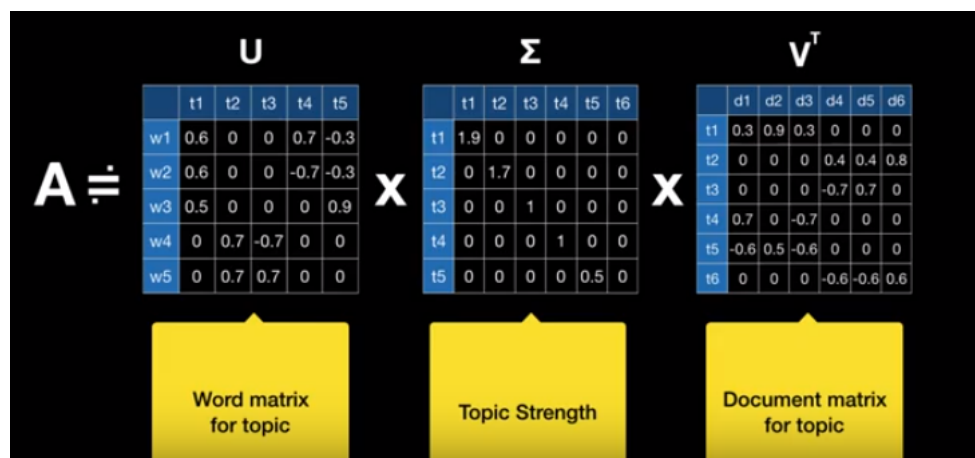
$$\Sigma_r = \begin{bmatrix} 3.76 & 0 & 0 \\ 0 & 1.99 & 0 \\ 0 & 0 & 1.37 \end{bmatrix}$$

$$V_r^T = \begin{bmatrix} -0.33 & -0.62 & -0.29 & -0.29 & -0.29 & -0.16 & -0.33 & -0.29 & -0.17 \\ -0.15 & -0.33 & 0.45 & 0.45 & 0.45 & 0.24 & -0.15 & -0.18 & -0.39 \\ 0.44 & -0.15 & -0.11 & -0.11 & -0.11 & 0.46 & 0.44 & -0.58 & -0.02 \end{bmatrix}$$

현재 우리는 3개의 문서를 단어집합으로 만들어 행렬화 하였고, 이를 Compact SVD를 하여 특이값 분해를 진행하였다.

LSA를 진행하기 전, Compact SVD를 통해 얻어낸 U_r, Σ_r, V_r^T 행렬의 특성을 살펴보자. U_r, Σ_r, V_r^T 행렬은 다음 그림과 같이 특성을 가지게 된다.

(다음 그림은 TDM이므로 그림에서 $U = V_r^T, V^T = U_r$ 로 생각하자)



기존의 DTM행렬 A 는 $Document \times Term$ 의 크기를 가진 행렬이었다. LSA에서 SVD를 진행한다는 의미는 $Document$ 와 $Term$ 간의 관계에 어떤 $Topic$ 이 내재되어있다고 가정하고 이를 특이값 분해를 통해 찾고자 한다. 따라서 $A = U_r \Sigma_r V_r^T$ 연산은

$Document \times Term$ 의 관계를 다음과 같이 표현하는 것과 같다.

$$Document \times Term = (Document \times Topic)(Topic \times Topic)(Topic \times Term)$$

따라서, 각 행렬 U_r, Σ_r, V_r^T 을 확인하는 것으로, 토픽과 문서의 관계, 토픽의 영향력, 단어와 토픽의 관계를 확인할 수 있다. 이는 주제를 압축해도 유효하고, 이를 바탕으로

이제 이 문서당 하나씩 나타나고 있는 주제들을 압축해보자. 여기서 가장 큰 순으로 2개의 특이값만을 사용하여, Truncated SVD를 진행 한다. Truncated SVD 결과를 통해 생성된 행렬 U_t, Σ_t, V_t^T 는 다음과 같다.

$$U_t = \begin{bmatrix} -0.62 & 0.47 \\ -0.47 & 0.41 \\ -0.62 & -0.77 \end{bmatrix}$$

$$\Sigma_t = \begin{bmatrix} 3.76 & 0 \\ 0 & 1.99 \end{bmatrix}$$

$$V_r^T = \begin{bmatrix} -0.33 & -0.62 & -0.29 & -0.29 & -0.29 & -0.16 & -0.33 & -0.29 & -0.17 \\ -0.15 & -0.33 & 0.45 & 0.45 & 0.45 & 0.24 & -0.15 & -0.18 & -0.39 \end{bmatrix}$$

본격적으로 U_t, Σ_t, V_t^T 에 대해 알아보기 전에, U_t, Σ_t, V_t^T 로 만든 유사행렬 A' 가 얼마나 원래 행렬 A 와 비슷한지 확인해보자.

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

$$A' = \begin{bmatrix} 0.62 & 1.13 & 1.09 & 1.09 & 1.09 & 0.60 & 0.62 & 0.50 & 0.01 \\ 0.46 & 0.84 & 0.89 & 0.89 & 0.89 & 0.49 & 0.46 & 0.37 & -0.02 \\ 1.01 & 1.99 & -0 & -0 & -0 & 0.01 & 1.01 & 0.97 & 0.99 \end{bmatrix}$$

비교해본 결과, 오차가 있기는 하지만, 대체로 A' 와 A 가 유사하다는 것을 확인할 수 있다. 즉, 이런 식으로 작은 특이값을 제거하여 주제를 압축해도 충분히 원래 DTM을 잘 설명할 수 있게 된다.

$$U_t = \begin{bmatrix} -0.62 & 0.47 \\ -0.47 & 0.41 \\ -0.62 & -0.77 \end{bmatrix}$$

$$\Sigma_t = \begin{bmatrix} 3.76 & 0 \\ 0 & 1.99 \end{bmatrix}$$

$$V_r^T = \begin{bmatrix} -0.33 & -0.62 & -0.29 & -0.29 & -0.29 & -0.16 & -0.33 & -0.29 & -0.17 \\ -0.15 & -0.33 & 0.45 & 0.45 & 0.45 & 0.24 & -0.15 & -0.18 & -0.39 \end{bmatrix}$$

그렇다면 이제 U_t, Σ_t, V_t^T 행렬을 자세히 살펴보자.

U_t 는 앞에서 말했듯이 $Document \times Topic$ 의 의미를 가진다. 즉 U_t 를 확인하는 것으로, 문서와 토픽의 관계를 확인할 수 있다.

Σ_t 는 토픽의 중요도, 정보량을 나타낸다. 이때, $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_t$ 순으로 정렬되어있기 때문에, 어떤 주제가 이 문서 집합 내에서 중요하게 작용하는 지 알수도 있다.

V_t^T 는 $Topic \times Term$ 의 의미를 가진다. 이제 V_t^T 를 확인하는 것으로, 토픽과, 단어의 관계를 확인할 수 있다. LSA에서 특히 V_t^T 는 중요한데, 이는 V_t^T 행렬이 잠재되어있는 주제가 무엇인지 판단하는데 사용되기 때문이다. V_t^T 행렬의 행을 통해 토픽에 대한 각 단어의 영향력을 확인할 수 있기 때문에, 이를 바탕으로 해당 토픽이 어떤 주제에 관한지 결정하게 된다.

이렇게 LSA를 통해 문서 안에 내재된 잠재된 주제들을 찾을 수 있게 된다. LSA를 통한 토픽모델링은 다음과 같은 장점을 가진다.

1. 단순히 주어진 문서 집합에 대해서 특이값 분해를 진행하고, 원하는 주제의 개수만큼 특이값을 사용하여 분석을 진행하면 되기 때문에, 적용하기 간단하다.
2. 행렬분해의 결과 문서-토픽, 그리고 단어-토픽의 관계에 대한 행렬이 생성되는데(U_t, V_t^T) 이 행렬을 바탕으로 문서 집합내에서 각 문서끼리, 그리고 각 단어끼리의 유사도를 확인할 수 있게 된다. 따라서, 문서 집합 내에 잠재된 단어의 내재적 의미를 좀 더 얻어낼 수 있다.
3. Σ 행렬의 특성($\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_t$ 가 정보량 내지는 중요도를 나타냄) 때문에 전체 문서 내에서의 토픽의 영향력을 확인할 수 있다.

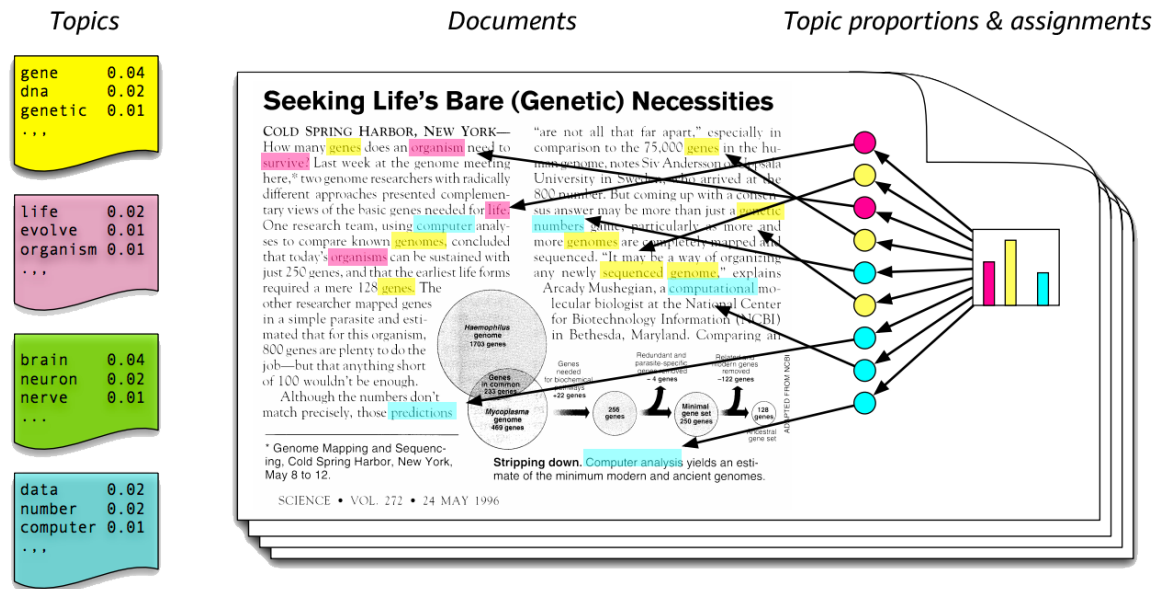
반면, LSA는 다음과 같은 한계 또한 가진다.

1. 행렬분해를 기반으로 진행되기 때문에, 데이터가 계속 추가될 때마다 다시 특이값분해를 진행해줘야 한다.
2. U_t, V_t^T 가 Orthogonal Matrix에서 해당하지 않는 부분의 행 또는 열을 제거한 형태이기 때문에, 값이 -1~1사이에서 나오게 되고, 결국 각 문서가 어느 토픽에 속하는지, 토픽이 어떤 내용인지 해석하기가 굉장히 까다롭다.

이러한 한계 때문에, LSA는 토픽 모델링보다는, Truncated SVD로 인한 차원 축소로 각 문서간, 혹은 각 단어간 관계를 분석하는데 더 많이 사용된다.

2.6.1.2. 잠재 디레클레 할당(Latent Dirichlet Allocation, LDA)

LSA의 단점들을 보완하기 위해서 '단어와 문서의 잠재구조'라는 가정만 유지하고 이를 확률적인 구조로 접근한 pLSA등이 제시되었다. pLSA의 경우에는 문헌 내에 특정 용어가 등장할 확률을 바탕으로 토픽모델링을 진행하였다. 그러나 pLSA는 문헌내에서 주제가 어떻게 분포할지는 고려하지 않기 때문에, 이를 고려하는 모델인 LDA가 개발되었다.



본격적인 설명에 앞서 위의 그림을 통해 LDA를 좀 더 쉽게 이해해보자. 각 토픽은 확률분포형태로 존재하여 문서내에 존재하는 어떤 단어들을 생성한다. 노란색 토픽에서 가장 많이 나타나는 단어 'gene', 'dna', 'genetic'이라는 단어를 통해, 우리는 해당 토픽이 '유전자'에 대한 토픽임을 쉽게 짐작할 수 있다.

또한 문서에 대해서도 보면, 다른 주제들보다 노란색 토픽에 해당하는 단어들이 많이 등장하고 있다. 따라서, 이 문서는 '유전자'에 대한 문서일 가능성이 높음을 쉽게 알 수 있다.

LDA는 각 문서에 어떤 주제들이 존재하는지 확률적으로 나타내주는 모델이다. 이를 위해서, LDA는 문서에서 나타난 단어를 통해 문서별 토픽의 분포, 토픽별 단어의 분포를 추정하게 된다.

LDA는 베이지안 기반의 복잡한 생성 모델이고, 또한 LDA를 적용하는데에는 단순히 관련 모듈을 통해 쉽게 적용할 수 있기 때문에, LDA가 어떻게 문서를 생성한다고 가정하는지, 어떻게 LDA를 통해 토픽모델링을 진행하는지만 간단하게 살펴보고자 한다.

(1) LDA의 문서 생성 과정 가정

LDA는 문서에서 나타난 단어들이 LDA에서 가정한 문서 생성 과정에 의해 생성되었다고 가정하여 문서별 토픽의 분포, 토픽별 단어의 분포를 추정하게 된다.

본격적으로 LDA의 문서 생성 과정을 설명하기전에 다음과 같이 용어를 정의하고 넘어가도록 하자.

- 문서 집합에서 등장하는 모든 단어에 각각 $\{1, \dots, V\}$ 의 인덱스 번호가 붙어있고, 이를 One-Hot Encoding을 해주어 V 차원 벡터로 변형해준다.
- 문서에서는 N 개의 단어가 연속하여 나타나고, 이를 $w = (w_1, w_2, \dots, w_N)$ 으로 표시해준다. 이때, w_n 는 문서에서 나타나는 n 번째 단어이다.
- 말뭉치(문서집합, Corpus)는 M 개 문서의 모임이며, 이를 $D = \{w_1, w_2, \dots, w_M\}$ 로 표시해준다.
- 문서가 가질 수 있는 주제는 총 K 개이며, 이를 $z = (z_1, \dots, z_K)$ 로 표시해준다. 이때 z_n 은 n 번째 주제이다.
- 우리가 알고싶은 문서별 토픽분포는 θ , 토픽별 단어분포는 β 로 표시한다.

LDA는 Corpus에 있는 각 문서 w 에 대해서 다음과 같은 생성 과정을 가정한다.

1. Choose $N \sim \text{Poisson}(\xi)$

:각 문서에서 나타나는 단어의 개수(문서의 길이)가 어떤 포아송 분포를 따른다고 가정한다.

*여기서 ξ 가 어떤 역할을 하는지 궁금할 수 있는데, 실제 논문에서 이 부분에 내용이 $N \sim \text{Poisson}(\xi)$ 부분을 제외하고 한 번도 다시 나오지 않기 때문에(...) 그냥 N 이 포아송 분포를 따른다 정도만 이해하고 넘어가면 된다.

2. Choose $\theta \sim \text{Dir}(\alpha)$

: 각 문서의 토픽 분포가 어떤 디레클레 분포를 따른다고 가정한다. 디레클레 분포는 지난 스터디에서 softmax함수가 어떤 확률분포를 생성했던 것처럼 확률 분포를 생성해주는 역할을 한다. 또한 디레클레 분포의 파라미터 α 는 하이퍼파라미터로, θ 를 조절해주는 역할을 한다. 이는 뒤에서 추가로 살펴보고자 한다.

3. For each of the N word ω_n :

(a) Choose a topic $z_n \sim \text{Multinomial}(\theta)$

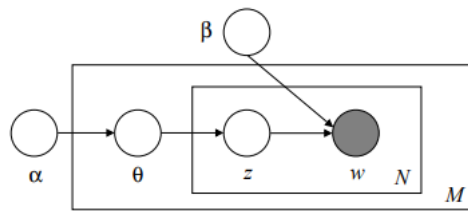
: 어떤 문서의 토픽이 2.에서 생성한 각 문서의 토픽분포를 바탕으로 파라미터로 하는 다항분포를 따른다고 가정한다.

(b) Choose a word ω_n from $P(\omega_n|z_n, \beta)$,

$P(\omega_n|z_n, \beta)$ is a multinomial probability conditioned on the topic z_n

: 3.(a)에서 구한 해당 문서의 토픽 z_n , 각 토픽별 단어 분포 β 가 주어졌을 때 단어 ω_n 이 나타날 다항분포의 확률로 어떤 단어 ω_n 이 Sample되었다고 생각한다.

이 과정을 그래프의 형태로 나타내면 다음과 같이 나타낼 수 있다.



이 과정을 통해 문서가 생성되었다고 생각하면, 우리가 스터디 전반에서 다루었던 것처럼 이를 베이지 정리를 통해 각 파라미터를 최적화할 수 있고, 우리가 알고싶어하는 각 문서의 주제와, 각 주제에서 주요하게 나타나는 단어를 확률분포의 형태로 얻어낼 수 있다.

*LDA를 최적화하는 방법은 매우 복잡하기 때문에, LDA를 최적화하는 과정에서 깃스 샘플링(Gibbs Sampling)이라는 기법을 사용하여 다른 단어들을 전부 고정하고, 한 단어씩 어떤 주제에 속하는지 다시 추정하는 방식으로 최적화를 진행한다. 이를 여러번 반복하다보면, 모든 단어에 대한 토픽 할당정보가 수렴하게 된다. 자세한 내용은 [Reference](#)를 참고 바란다.

(2) LDA 결과 해석 - β, θ 를 바탕으로

이제 우리는 LDA를 통해 ①각 토픽에서의 단어가 나타날 확률 분포(β), ②각 문서의 토픽 분포(θ)를 얻어 낼 수 있었다. 이제 LDA의 결과를 해석하기 위해서

1. β 를 확인하여 각 토픽에서 어떤 단어들이 주요하게 나타나는지(나타날 확률이 높은지) 확인하여 각 토픽들이 어떤 주제에 대한 내용인지 확인한다.
2. 이후 θ 를 확인하여 각 문서들에서 어떤 주제가 주요하게 나타나는지(나타날 확률이 높은지) 확인하여 각 문서가 어떤 주제를 가질지 확인한다.

그렇다면 예시를 통해 한 번 확인해보도록 하자. 6($M = 6$)개의 문서에서 총 10($V = 10$)개의 단어가 나타났다고 가정해보자.

먼저 β 를 확인해보자. β 는 $K \times V$ 크기의 행렬 형태로 나타난다.

Terms	Topic 1	Topic 2	Topic 3
Baseball	0	0	0.2
Basketball	0	0	0.267
Boxing	0	0	0.133
Money	0.231	0.313	0.4
Interest	0	0.312	0
Rate	0	0.312	0
Democrat	0.269	0	0
Republican	0.115	0	0
Caucus	0.192	0	0
President	0.192	0.063	0

먼저 β 가 각 토픽에서의 단어의 확률분포에 대한 행렬이기 때문에, 확률분포의 성질에 따라 β 의 각 열의 합은 항상 1이 된다.

그래서 이 β 의 각 열을 확인하는 것으로 해당 토픽에서 주요하게 나타날, 혹은 나타날 확률이 높은 단어들을 확인하여 해당 토픽이 어떤 내용인지를 쉽게 판단할 수 있다.

예시를 확인해보면 Topic 1에서는 'Democrat(민주당)', 'Money', 'Caucus(전당대회)', 'President'가 주요 단어로 나타나는 것으로 보아 대통령 선출에 따른 경제효과에 대한 주제임을 확인할 수 있을 것이고, 이와 같이 추론하면 Topic 2는 경제 관련 주제, Topic 3는 스포츠와 관련된 주제임을 쉽게 파악할 수 있을 것이다.

다음으로 θ 를 살펴보자. 각 문서의 주제분포 θ_d 를 행으로 결합한다면, 다음과 같이 나타낼 수 있을 것이다.

Docs	Topic 1	Topic 2	Topic 3
Doc 1	0.4	0	0.6
Doc 2	0	0.6	0.4
Doc 3	0.375	0.625	0
Doc 4	0	0.375	0.625
Doc 5	0.5	0	0.5
Doc 6	0.5	0.5	0

아까와 같이, θ_d 가 확률분포이기 때문에 각 행의 합은 1이 된다.

우리는 지난 스터디에서 Attention을 배우면서 어떤 확률분포를 가중치의 형태로도 생각할 수 있다는것을 확인하였다. 이를 바탕으로 생각해보면, 각 문서는 주제와 그 주제가 나타날 확률의 가중합으로 표현할 수 있을 것이다.

이런식으로 생각하면 예시도 다음과 같이 해석할 수 있을 것이다. 예를 들어 $Doc1 = 0.4 \times (Topic1) + 0 \times (Topic2) + 0.6 \times (Topic3)$ 이기 때문에, Doc 1은 Topic 3에 대한 내용을 더 많이 서술하고 있음을 확인할 수 있고, 이를 바탕으로 Doc 1은 Topic 3일 확률이 더 높다고 생각할 수 있다.

이렇게 β, θ 에 대해 살펴보는 것으로 LDA를 통해 효과적으로 문서집합에서 K 개의 토픽에 대하여 그 토픽이 어떤 의미인지, 각 문서가 어떤 주제를 더 많이 다루고 있는지 쉽게 확인할 수 있다.

(3) LDA의 장점과 한계

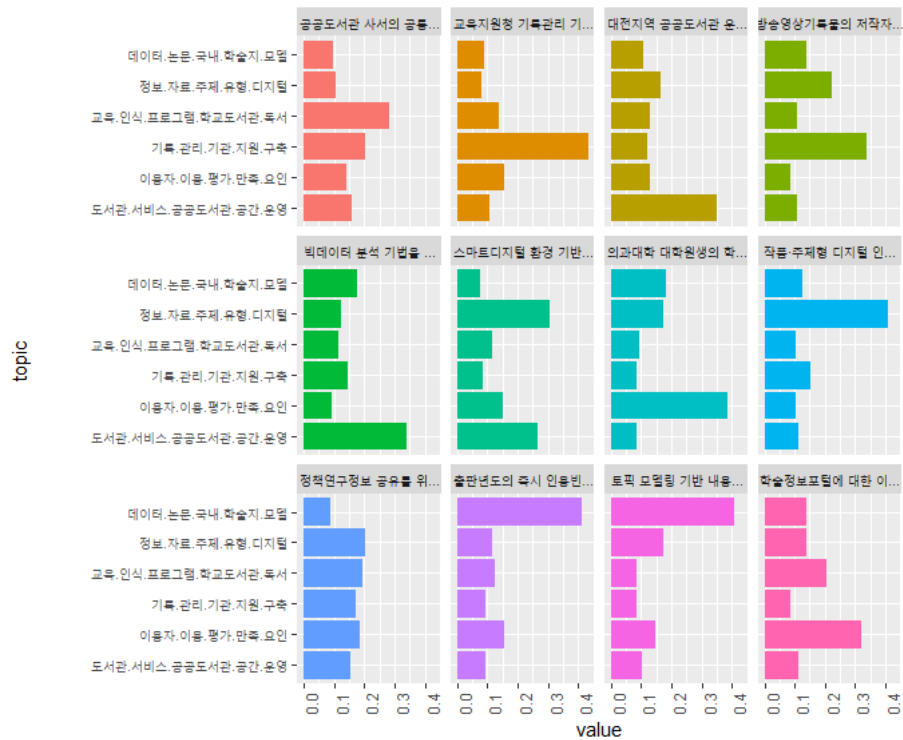
LDA를 통하여 토픽모델링을 진행하는 과정과 어떻게 결과를 해석하는지를 (1)과 (2)에서 살펴보았다. LDA는 매우 효과적인 토픽모델링 기법이고, 다음과 같은 장점을 가진다.

1. 각 토픽에 대한 정보를 쉽게 얻을 수 있다.

LSA에서는 각 특이값에 대응하는 주제가 어떤 것인지 파악하기 어려웠지만, LDA에서는 각 토픽별 단어가 생성될 확률을 β 를 통해 알 수 있었기 때문에, 가장 나타날 확률이 높은 단어가 해당 토픽에서 주요한 단어라는 것을 쉽게 알 수 있기 때문에, 각 토픽이 어떤 내용인지 쉽게 해석할 수 있다.

2. 각 문서의 주제가 가중합의 형태로 나타난다.

LDA에서 각 문서가 어떤 주제를 나타낼지는 확률분포(θ_d)의 형태로 나타나고 있다. 그래서 LDA는 가중합의 형태로 각 문서의 주제를 파악할 수 있고, 이는 다양한 주제를 가지고 있는 문서들의 토픽 모델링에 특히 효과적이다.

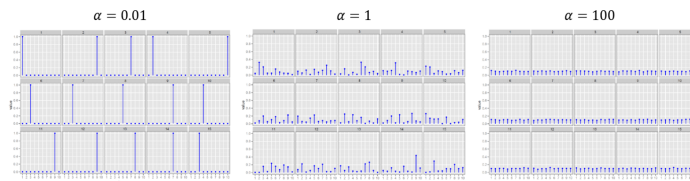


위 그래프들은 문헌정보학 논문 초록 정보를 바탕으로 LDA로 토픽모델을 진행하여 θ 를 시각화한 것이다. 문헌정보학 분야의 특성상 간학문적인 연구가 활발하여 주제를 하나로 표현하기 어려운 경우가 많아 3행 1열에 있는 그래프처럼 나타나는 경우가 많다.

기존 토픽모델링에서는 각 문서를 하나의 주제에 할당시켜 이런 다양한 주제를 가진 문서에 대해서 모델링하기 어려웠지만, LDA는 이런 문서들의 주제를 모델링하는데 특히 효과적이다.

▼ θ 의 하이퍼파라미터 α 의 역할

그렇다고 LDA가 각 문서를 하나의 주제에 할당시키지 못하는 것은 아니다. $\theta \sim \text{dir}(\alpha)$ 이기 때문에, 하이퍼파라미터 α 를 조정해주는 것으로 각 문서의 주제분포를 조정해줄 수 있다. 이때, $\alpha = 0.01$ 와 같이 매우 작은 수라면, 문서가 하나의 주제에 할당될 확률이 한 주제에만 매우 높게 나타나고, $\alpha = 100$ 와 같이 매우 큰 수라면, 문서가 하나의 주제에 할당될 확률이 고르게 나타나게 된다. 다음과 같은 그림을 확인하면 쉽게 이해할 수 있을 것이다.



반면, LDA에도 한계가 존재한다. LDA의 한계는 다음과 같다.

1. 주제의 개수를 직접 설정해주어야 한다.

LDA에서 각 문서가 가질 수 있는 주제의 수 K 는 하이퍼파라미터이기 때문에, 이를 직접 설정해줘야 한다.

만약 해당 문서집합이 가지는 주제의 수가 도메인 지식으로 주어진다면 이를 사용하여 토픽모델링을 진행하면 되지만, 그러한 사전 지식이 없다면 연구자의 재량으로 이를 설정해줘야하기 때문에, 연구자의 주관이 개입될 수밖에 없다.

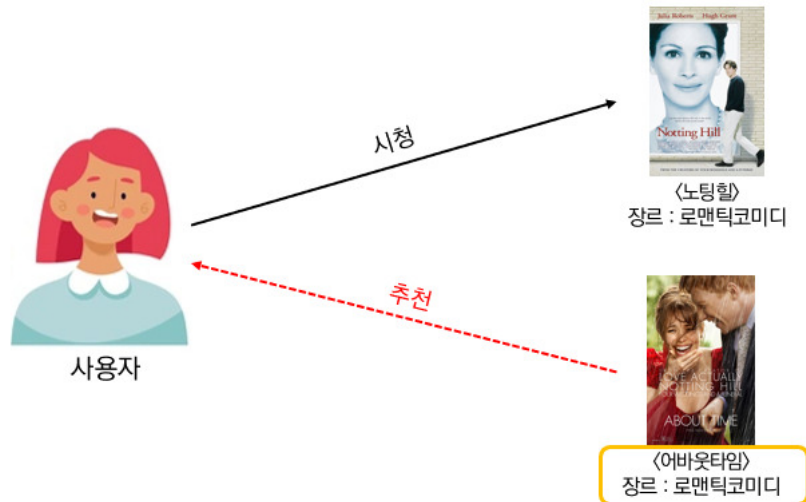
이런 맥락에서 사전지식 없이 LDA에서의 최적의 토픽 수 K 를 찾는 방법이 연구되었고, 최적의 토픽 수를 찾기 위해 Perplexity, Coherence등을 사용하게 된다. 자세한 내용은 Reference를 참조하길 바란다.

이렇게 LDA는 토픽모델링에서 매우 효과적인 기법으로 자리매김하고 있으며, 토픽모델링 분야에서는 바탕으로 가중치를 적용한 LDA 등 LDA를 기반으로 하여 문서집합에 내재되어 있는 주제들에 대해서 효과적으로 분석하려는 등의 연구가 이루어지고 있다.

2.6.2. 협업필터링(Collaborative Filtering)

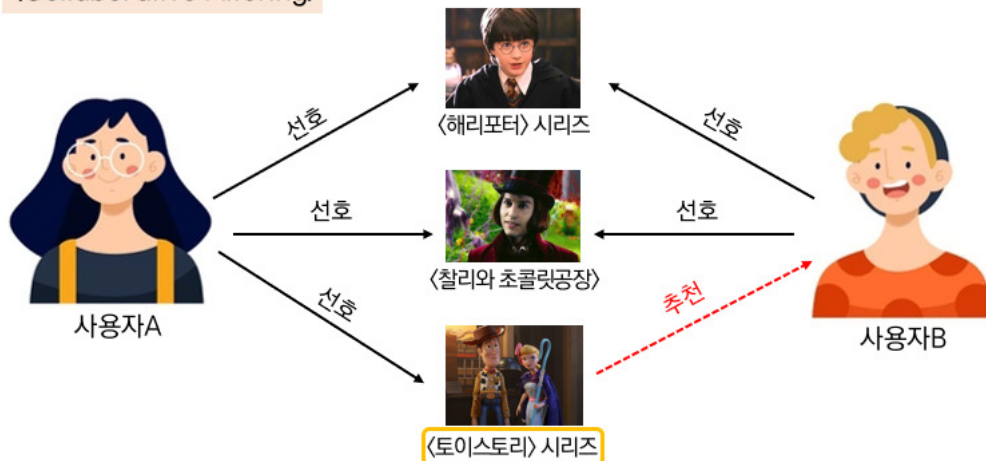
추천시스템은 말 그대로 사용자(User)에게 어떤 콘텐츠 내지는 제품(혹은 item)을 추천해주는 시스템이다. 이런 추천시스템은 이런 콘텐츠를 제공하는 플랫폼을 사용하는 유저가 증가하고, 제공되는 콘텐츠 또한 많아지면서 이용자에게 적합한 콘텐츠, 특히 유저의 니즈에 맞는 콘텐츠를 추천해 줄 필요성이 증가하였다.

내용 기반 필터링 (Content-based Filtering)



이런 추천 시스템은 콘텐츠 기반 필터링과 협업 필터링으로 나뉜다. 콘텐츠 기반 필터링은 콘텐츠를 설명하는 메타데이터를 통해 각 콘텐츠끼리의 유사도를 구하고, 이를 바탕으로 사용자가 이전에 소비한 콘텐츠 내지는 관심있는 콘텐츠와 가장 유사한 콘텐츠를 사용자에게 추천하는 방법이다. 그러나 이 방법은 각 콘텐츠간 유사도를 구하는 과정에서 분석자와 개발자의 주관이 개입될 여지가 있다.

협업 필터링 (Collaborative Filtering)



그렇다면 협업필터링은 어떻게? 협업 필터링은 구입내역, 선호도, 만족도를 기반으로 사용자 혹은 제품 간의 협업(상호작용)을 통하여 비슷한 성향을 가진 사용자가 선호하는 제품을 추천하는 방법이다.

협업필터링을 위해서는 사용자와 제품 간의 상호작용데이터를 표현해줘야 한다. 이런 상호작용하는 데이터는 숫자로 표현되기 때문에, 이를 행렬의 형태로 표현할 수 있다. 보통 사용자가 매긴 제품에 대한 선호도(평점)을 바탕으로 시스템을 설계하기 때문에 이를 평점행렬이라고 한다. 4명의 유저와 5개의 아이템에 대한 평점행렬은 다음과 같을 것이다.

	Item 1	Item 2	Item 3	Item 4	Item 5
User 1	5	4	4	3	1

	Item 1	Item 2	Item 3	Item 4	Item 5
User 2	1	1	1	3	4
User 3	4	4	2		3
User 4	4	2	3	2	2

이런식으로 행렬의 형태로 유저와 아이템 간의 관계를 표현한다면, 각 유저 혹은 각 아이템에 대해서 쉽게 벡터로 표현할 수 있게 된다. 벡터 연산을 통해 사용자 혹은 아이템끼리 유사도를 계산한다면 아직 사용자가 매기지 않은 아이템에 대한 평점도 유사도를 가중치로 두어 다른 유저/아이템의 평점과 가중합을 계산하는것으로 예측평점을 계산할 수 있을 것이다. 이런 방법을 유저 기반/아이템 기반 협업필터링 (User-Based/Item-Based Collaborative Filtering)이라고 한다.

그러나 이렇게 예측평점을 매기는 방법은 충분히 합리적이고 효과적으로 사용자의 선호에 맞추어 추천을 해 줄 수 있겠지만, 실제 데이터는 수천만 명의 사용자와 수만 개의 콘텐츠에 대한 선호도를 담고 있기 때문에, 이를 모두 고려하여 유사도를 계산하는 것은 시간적으로나 공간적으로나 비효율적일 것이다.

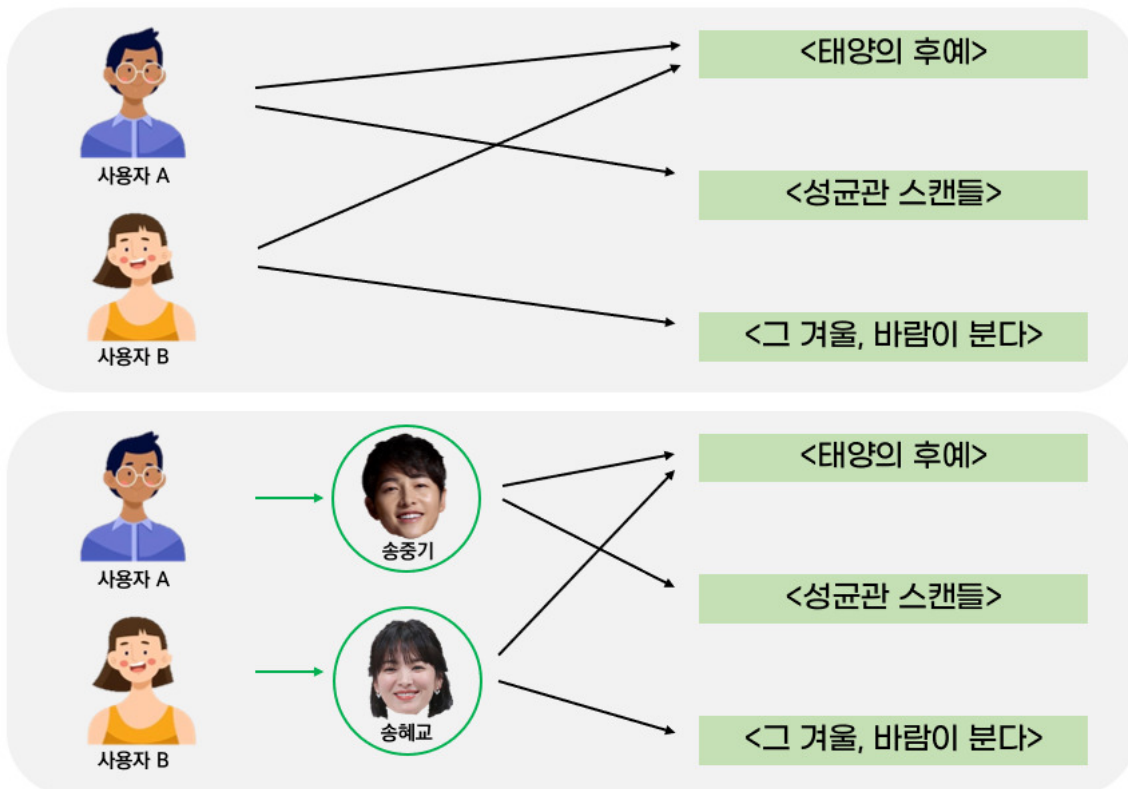
따라서, 평점 행렬을 그대로 사용하여 분석하는 것에는 한계가 있다.

2.6.2.1. 잠재 요인 협업 필터링(Latent Factor-Based Collaborative Filtering)

따라서, 원래 평점행렬을 그대로 사용하는 것이 아닌 이를 분해하여 사용자가 평점을 부여할 때 잠재적으로 고려된 요소를 파악하는 잠재 요인 협업 필터링이 고안되었다. 이는 LSA에서 토픽모델링을 진행할 때 문서-단어의 관계를 설명할 수 있는 잠재적인 주제가 있다고 가정하는 것과 같다. 잠재요인 협업필터링은 유저-아이템 간의 관계에서 생성된 평점(선호도)가 잠재적인 요소에 의해 나타난다고 가정한다.

모델 기반 협업 필터링

(Model-based Collaborative Filtering) ※ 화살표는 선호를 반영



기존 유저 기반/아이템 기반 협업필터링은 이런 잠재적인 요인에 대한 가정을 하지 않기 때문에, 사용자와 아이템간의 유사도를 전부 계산 했어야 했고, 이로 인해 연산량이 기하급수적으로 늘어나는 현상이 나타났다.

그러나 유저-아이템 사이에 작용하는 잠재적인 요인이 존재한다고 가정하게 된다면, SVD를 통해 쉽게 유저-잠재요인/잠재요인-아이템간의 관계를 쉽게 파악할 수 있게되고, 그 관계에서 각 잠재요인이 평점(선호도)에 얼마나 영향을 끼치는지 쉽게 파악할 수 있게 된다.

또한 특이값으로 나타나는 잠재요인들의 영향력 중에서 영향력이 적은 잠재요소들은 제외해준다면, Truncated SVD를 통해 최대한 원래 유저와 아이템의 평점행렬을 잘 설명해줄 수 있는 행렬을 만든다면 연산량도 훨씬 줄일 수 있을 것이다.

그러나 이는 실질적으로 불가능하기 때문이다. 이는 평점행렬의 특성 때문이다. 대부분의 평점행렬은 위에서 본 것들과 같이 이상적으로 나타나지 않고, 다음과 같은 형태로 나타게 된다.

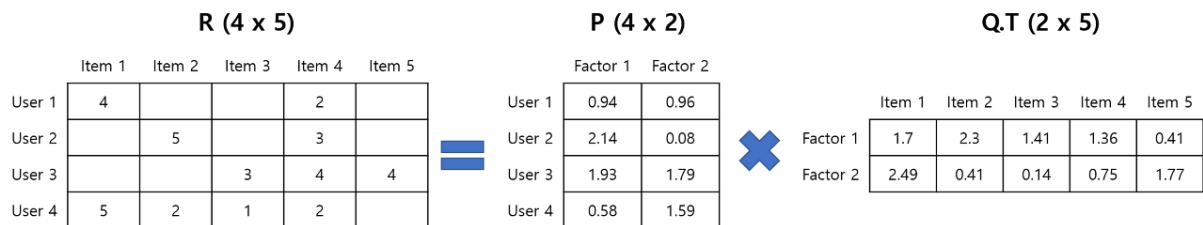
	Item 1	Item 2	Item 3	Item 4	Item 5
User 1	4			2	
User 2		5		3	1
User 3			3	4	4
User 4	5	2	1	2	

모든 유저들은 모든 아이템(컨텐츠)을 소비하지 않고, 소비하더라도 항상 평점을 매기는 것이 아니다. 이런 경향은 유저의 수가 많을수록, 아이템의 수가 많을수록 심화된다. 따라서, 대부분의 평점행렬은 대부분의 원소가 결측치(NA)인 희소 행렬(Sparse Matrix)의 형태를 띄게 된다.

이렇게 행렬 내의 원소들이 결측치로 나타나게 된다면 예측의 정확도를 떠나서 기본적인 행렬연산과 벡터연산이 불가능해진다. 따라서 이 상태로는 유사도를 바탕으로 한 유저 기반/아이템 기반 협업필터링은 물론이고, 특이값분해 또한 진행할 수 없다.

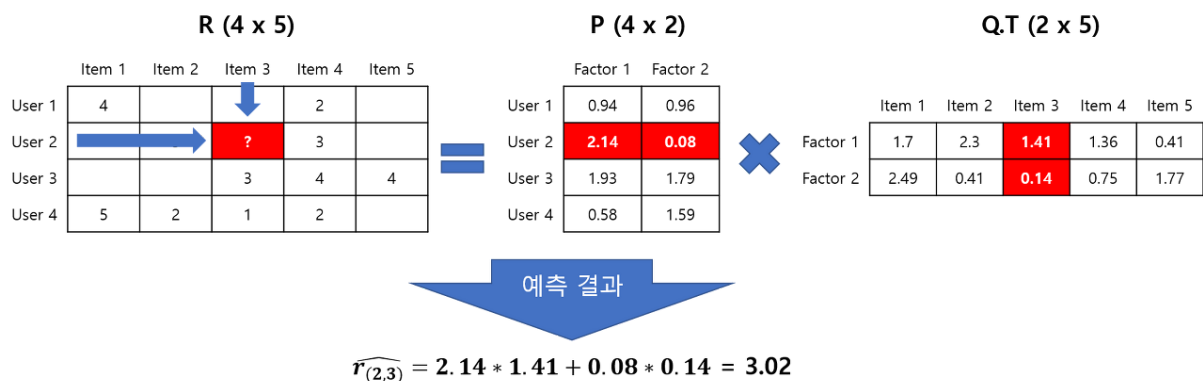
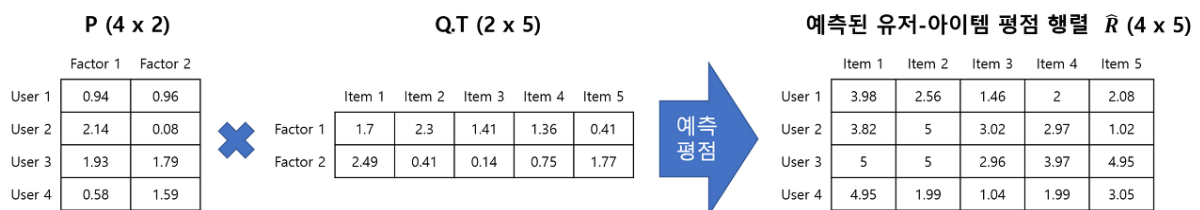
또한, 결측치를 어떻게든 잘 처리한다고 해도 유저 기반/아이템 기반 협업필터링의 정확도는 떨어질 수 밖에 없고, 유저 기반/아이템 기반 협업필터링의 특성상 원래 행렬을 그대로 사용하기 때문에 어떤 오차를 줄이는 방향으로 결측치를 채울 수 없게 된다.

따라서, 행렬분해를 기반으로 진행되는 잠재 요인 협업 필터링은 이런 결측치에 대한 예측과 이 예측값이 원래 행렬의 값과 비슷하도록 최적화하는 과정을 필요로 한다. 그렇다면 행렬분해에 기반한 잠재요인 협업필터링이 어떻게 진행되는지 알아보자.



먼저 앞서 본 희소 행렬의 예제를 생각해보자. 이 희소 행렬이 원래 평점 행렬 R 이라고 하자. 그리고 전체 특이값 중 2개만 사용하여 Truncated SVD를 사용했을 때 얻을 수 있는 행렬 P, Q^T 가 있다고 가정한다. 이때, P, Q^T 행렬의 초기값은 평점행렬의 값의 범위에 따라 랜덤하게 나온 임의의 값을 사용한다.

$$R \cong \hat{R} = P * Q.T$$



앞으로 행렬 P, Q^T 를 바탕으로 평점 행렬 R 의 예측 행렬인 \hat{R} 을 얻어내게 되고, 예측행렬 \hat{R} 의 값으로 원래 행렬 R 의 결측치를 처리하게 된다.

예를 들어 User 2가 아직 매기지 않은 Item 3에 대한 평점 $\hat{r}_{(2,3)}$ 은 행렬 P, Q^T 를 곱해서 얻어낸 값 $\hat{r}_{(2,3)} = 2.14 \times 1.41 + 0.08 \times 0.14 = 3.02$ 로 예측하게 된다.

이렇게 결측된 평점을 다음과 같이 처리할 수 있다면, 남은 것은 행렬 P, Q^T 를 통해 얻어내는 예측행렬 \tilde{R} 이 최대한 원래 행렬 R 과 비슷하게 최적화해주는 작업을 진행해주면 된다. 이를 위해선 원래 행렬에서 결측치가 아닌 값(실제 사용자가 아이템에 대해 매긴 값)과 예측행렬의 해당부분에 대한 값의 오차가 최소화되도록 최적화를 진행해주면 될 것이다.

원래 행렬 R 과 예측행렬 \tilde{R} 의 오차가 최소가 되는 행렬 P, Q^T 를 찾기 위해서 다음과 같이 최적화를 진행한다. 이는 우리가 이전 스터디 미적분 파트에서 다루었던 모든 내용의 응용이다.

$$\min \sum (r_{(u,i)} - p_u q_i)^2 + \lambda (\|q_i\|^2 + \|p_u\|^2)$$

- $r_{(u,i)}$: 실제 R 행렬의 u행, i열에 위치한 값
- p_u : P 행렬의 사용자 u행 벡터
- q_i : Q 행렬의 아이템 i행 전치벡터
- λ : L2 정규화 계수

먼저 최적화식의 목적함수부터 살펴보자. 우리의 목표는 원래 행렬 R 과 예측행렬 \tilde{R} 의 오차가 최소가 되도록 하는 것이었기 때문에, 우리가 최소화하고 싶은 목적함수는 다음과 같았을 것이다.

$$L(r) = (r_{(u,i)} - \hat{r}_{(u,i)})^2, \text{ where } \hat{r}_{(u,i)} = p_u q_i$$

그러나, 실제로 우리의 최적화 식에는 p_u, q_i 의 norm의 제곱, 즉 L2 Norm을 통해 Regularization을 주었다. 지난 스터디 시간에 우리는 Regularization을 통해 모델의 Variance를 줄일 수 있다는 것을 확인하였다. 그렇다면 Regularization을 사용하지 않은 모델이 높은 Variance를 가지고 있다는 것인데, 왜 모델이 높은 Variance를 가지게 되었고, L2 Regularization을 사용해줄까?

이는 바로 원래 평점 행렬이 희소 행렬이기 때문이다. 우리는 결국 원래 행렬 R 과 예측행렬 \tilde{R} 의 오차가 최소가 되도록 최적화를 진행해주는데, 이때 모든 오차는 결국 원래 행렬에서 결측치가 아닌 값 $r_{(u,i)}$ 과 예측행렬의 해당 부분의 값 $\hat{r}_{(u,i)}$ 의 차이에서 나오기 때문이다. 그러나 원래 평점 행렬 R 이 희소 행렬이기 때문에, 결측치가 아닌 $r_{(u,i)}$ 의 수는 적을 수 밖에 없다. 이는 모델이 학습할 데이터가 적다는 것을 의미하고, 적은 학습 데이터는 모델의 Variance를 증가시켜 결과적으로 모델이 Overfitting되는 결과를 야기하게 된다. 따라서 우리는 목적함수에 L2 Norm을 적용하여 행렬을 최적화하게 된다.

그렇다면 다음으로, 목적함수를 바탕으로 어떻게 행렬 P, Q^T 의 각 벡터 p_u, q_i 를 업데이트하는지 확인해보자. 행렬 P, Q^T 의 각 벡터 p_u, q_i 는 다음과 같이 업데이트 된다.

$$p_{u_{new}} = p_u + \eta(e_{(u,i)} * q_i - \lambda * p_u)$$

$$q_{i_{new}} = q_i + \eta(e_{(u,i)} * p_u - \lambda * q_i)$$

- $r_{(u,i)}$: 실제 R 행렬의 u행, i열에 위치한 값
- p_u : P 행렬의 사용자 u행 벡터
- q_i : Q 행렬의 아이템 i행 전치벡터
- λ : L2 정규화 계수
- $\hat{r}_{(u,i)}$: 예측 R 행렬의 u행, i열에 위치한 값
- $e_{(u,i)}$: u행, i열에 위치한 실제 행렬 값과 예측 행렬 값 차이
- η : 학습률
- λ : L2 정규화 계수

각 벡터가 업데이트 되는 것을 확인해보면, Stochastic Gradient Descent를 통해 p_u, q_i 를 업데이트 하는 것을 확인할 수있다. 이때 다른 Gradient Descent(Batch, Mini-Batch)를 사용하지 않는 이유 또한 평점 행렬의 특성 때문이다. 평점행렬은 앞서 말했던 것처럼 매우 많은 행과 열을 가지고 있는 행렬이기 때문에 모델이 High variance를 가진다고 앞서 말하기는 했지만, 그럼에도 불구하고 여전히 전체 데이터수는 굉장히 많기 때문에, 이를 모두 계산한 다음에 p_u, q_i 를 업데이트하는 것은 바람직하지 않을것이다. 또한, 결론적으로는 최적화를 통해서 원래 행렬과 예측행렬의 차이를 일정 값 이하로 줄이는 것이기 때문에, Stochastic Gradient Descent를 사용하면 굳이 복잡한 전체 행렬에 대해서 연산할 필요 없이 빠르게 p_u, q_i 를 업데이트하면서 전체 행렬에 대한 최적화를 진행할 수 있게 된다.

이렇게 전체적인 행렬분해 기반의 협업필터링이 어떻게 작동하는지를 확인해보았다. 이 내용을 다시 한번 정리하면 다음과 같이 정리할 수 있다.



행렬분해 기반 협업필터링의 작동 방식

1. P와 Q 행렬을 임의의 값을 가진 행렬로 초기화 한다.
2. P와 Q 전치행렬을 곱해 예측 R 행렬을 계산하고, 실제 R 행렬과의 차이를 계산한다.
3. 차이를 최소화할 수 있도록 P와 Q 행렬의 값을 적절한 값으로 각각 업데이트한다.
4. 특정임계치 아래로 수렴할 때까지 2, 3번 작업을 반복하면서 P와 Q 행렬을 업데이트해 근사화한다.

이런 행렬분해 기반의 협업필터링은 다른 추천 시스템 알고리즘보다 더 좋은 효과를 보인다고 한다.

Reference:

SVD와 LSA, Collaborative Filtering-설명은 무시하면 됩니다(특히 LSA부분)

Google Colaboratory

<https://colab.research.google.com/drive/1mrqd6M-y7Hp-vczRCfKA-rkjwFBVNPjT?authuser=1>

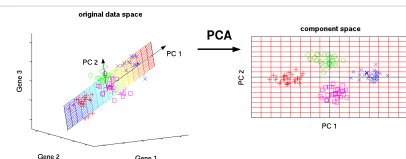


특이값 분해(SVD):

SVD와 PCA, 그리고 잠재의미분석(LSA)

© 2020 ratsgo. This work is licensed under CC BY-NC 4.0.

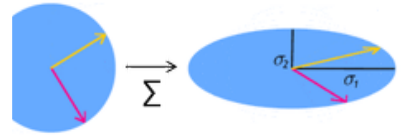
<https://ratsgo.github.io/from%20frequency%20to%20semantics/2017/04/06/pcasvds/>



[선형대수학 #4] 특이값 분해(Singular Value Decomposition, SVD)의 활용

활용도 측면에서 선형대수학의 꽃이라 할 수 있는 특이값 분해(Singular Value Decomposition, SVD)에 대한 내용입니다. 보통은 복소수 공간을 포함하여 정의하는 것이 일반적이지만 이 글에서는 실수(real) 공간에 한정하여 내용을 적겠습니다. 1. 특이값분해(Singular Value Decomposition, SVD) 특이값 분해(SVD)는 고유값 분해

📌 <https://darkpgmr.tistory.com/106>



SVD(특이값 분해)

특이값 분해는 $m \times n$ 크기의 임의의 사각 행렬 A 를 특이 벡터(Singular vector)의 행렬과 특이값(Singular value)의 대각 행렬로 분해하는 것이다. 특이값 분해를 하면 모든 성분이 가치가 높은 순으로 정렬되어 분해된다. 가치가 낮은 부분은 truncate로 제거, 그리고 다시 복원한다.

📌 <https://velog.io/@guide333/SVM%ED%8A%B9%EC%9D%B4%EA%B0%92-%EB%B6%84%ED%95%B4>

velog

토픽 모델링:

http://ai4school.org/?page_id=2117

나이브 베이즈 분류기 (Naive Bayes Classifier)

본 포스트는 문일철 교수님의 인공지능 및 기계학습 개론 I 강의를 바탕으로 작성하였습니다. 나이브 베이즈 분류기(Naive Bayes Classifier)는 간단하고 여러 분류 문제에 적용하기 쉬우면서도 뛰어난 성능을 보이는 분류기 중 하나입니다. 인스턴스의 레이블을 예측할 때 베이즈 결정 이론(Bayes decision theory)을 사용하여 판단하기 때문에 이러한 이름이 붙었습니다. 이번 게시물에서는 베이즈 결정 이론이란 무엇인지?

📌 https://yngie-c.github.io/machine%20learning/2020/04/08/naive_bayes/

[BoostCamp AI] Bag-of Words 정리

현재 NLP는 Transformer가 주도하고 그 이전에는 RNN과 같이 Recurrent한 모델이 주도했지만 그 이전에는 단어 및 문서를 숫자 형태로 나타내는 Bag-of Words 기법을 정리한다. Step 1. Constructing the vocabulary co

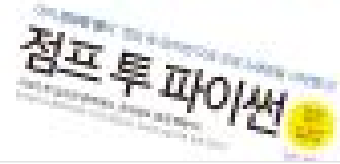
📌 <https://velog.io/@emeraldgoose/bagofwords>

velog

점프 투 파이썬

점프 투 파이썬 오프라인 책(개정판) 출간 !! (2019.06) ** * [책 구입 안내](https://wikidocs.net/4321) 이 책은 파이썬 ...

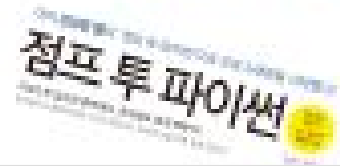
📌 <https://wikidocs.net/22650>



점프 투 파이썬

점프 투 파이썬 오프라인 책(개정판) 출간 !! (2019.06) ** * [책 구입 안내](https://wikidocs.net/4321) 이 책은 파이썬 ...

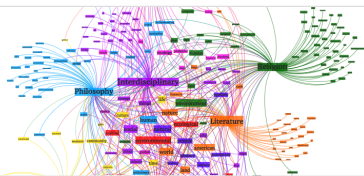
📌 <https://wikidocs.net/24559>



Topic Modeling with LSA, PSLS, LDA & Lda2Vec

This article is an updated comprehensive overview of Topic Modeling and its associated techniques. In natural language understanding (NLU) tasks, there is a hierarchy of lenses through which we can extract meaning - from words to sentences to paragraphs to documents.

<https://medium.com/nanonets/topic-modeling-with-lsa-psla-lda-and-lda2vec-555ff65b0b05>




https://drive.google.com/file/d/1ae4_j55QgBo8qFbBOLfKQ-pM7kgfszcz/view?usp=sharing

[토픽 모델링] LSA와 LDA의 관계 (+ LDA라는 이름의 유래)

Latent Dirichlet Allocation이라는 이름은 언뜻 들으면 무슨 뜻인지 이해하기가 어렵습니다. 토픽 모델링 기법 중 하나로 문헌-용어 행렬에서 문헌별 주제분포와 주제별 단어분포를 찾아주는 기술인데, 이름은 Latent Dirichlet Allocation 이죠. Dirichlet는 이 기법이 디리클레 분포를 기반으로하기 때문에 붙은것이라고 유추가능하지만 Latent

📌 <https://bab2min.tistory.com/585>



 <http://www.cs.columbia.edu/~blei/papers/BleiNgJordan2003.pdf>

잠재요인 협업필터링:

데이터마이닝 3주차 클린업 교안

4. 행렬 분해를 이용한 잠재요인 협업 필터링

용어정의 행렬 분해의 이해 확률적 경사하강법(SGD)을 이용한 행렬 분해 확률적 경사하강법(SGD) 활용예제 행렬 분해를 이용한 개인화 영화 추천시스템 개발 행렬 분해는 다차원 매트릭스를 저차원 매트릭스로 분해하는 기법으로 대표적으로 SVD(Singular Value Decomposition), NMF(Non-Negative Matrix Factorization) 등이 있습니다.

🌐 <https://big-dream-world.tistory.com/69>

4		Item 5				P (4 x 2)	
						Factor 1	Factor 2
4	Item 5	=		User 1	0.94	0.96	X Factor 1 Factor 2
				User 2	2.14	0.08	
				User 3	1.93	1.79	
				User 4	0.58	1.59	