

# 비정형텍스트분석

## 5. Document level analysis

2020년 9월 29일

데이터사이언스학과 / 이영훈

TF-IDF

# How to represent documents

---

## Bag-of-words (BOW)

- 비정형 데이터를 정형화하는 방법으로 가장 널리 쓰여온 방법
- 각 문서를 사전의 크기만큼의 벡터로 표현
  - 1) 단어가 해당 문서에 존재하는지의 여부를 binary value로 부여
  - 2) 단어가 해당 문서에 미치는 weight 값을 부여 (e.g. term frequency)

**Document 1: I have a pen. You have a pen, too.**

**Document 2: I had lunch. Did you have lunch?**



ID	I	have	a	pen	you	too	had	lunch	did
1	1	1	1	1	1	1	0	0	0
2	1	1	0	0	1	0	1	1	1


# How to represent documents

---

## Term frequency and inverse document frequency (TF-IDF)

The most popular way to represent documents by weights

- $tf(w)$ : term frequency (number of word occurrences in a document)
- $df(w)$ : number of documents containing the word

$$TF - IDF(w) = \underline{tf(w)} \times \log\left(\frac{N}{\underline{df(w)}}\right)$$
The diagram consists of two arrows. One arrow starts from the blue text 'The word is more important if it appears several times in a target document' and points to the underlined term frequency 'tf(w)' in the formula. The other arrow starts from the orange text 'The word is more important if it appears in less documents' and points to the underlined inverse document frequency 'df(w)' in the formula.

The word is more important if it appears  
several times in a target document

The word is more important if it  
appears in less documents

# How to represent documents

---

Binary term-document incidence matrix

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	1	1	0	0	0	1
Brutus	1	1	0	1	0	0
Caesar	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0

# How to represent documents

---

## Term-document count matrix

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	157	73	0	0	0	0
Brutus	4	157	0	1	0	0
Caesar	232	227	0	2	1	1
Calpurnia	0	10	0	0	0	0
Cleopatra	57	0	0	0	0	0
mercy	2	0	3	5	5	1
worser	2	0	1	1	1	0

# How to represent documents

---

Term-document matrix with TF-IDF values

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	5.25	3.18	0	0	0	0.35
Brutus	1.21	6.1	0	1	0	0
Caesar	8.59	2.54	0	1.51	0.25	0
Calpurnia	0	1.54	0	0	0	0
Cleopatra	2.85	0	0	0	0	0
mercy	1.51	0	1.9	0.12	5.25	0.88
worser	1.37	0	0.11	4.15	0.25	1.95

## Exercise 2 : TF - IDF

DF(w)

- Total number of documents : 1000
- "oil" is found in 200 documents, "Mexico" is found in 50 documents, "refinery" is found in 1000 documents

TF(d, t)

- $d1$ : "oil": 40    "Mexico": 80    "refinery": 100
- $d2$ : "oil": 160    "Mexico": 100    "refinery": 100

IDF: "oil":  "Mexico":  "refinery":

TF.IDF

- $d1$ : "oil":  "Mexico":  "refinery":
- $d2$ : "oil":  "Mexico":  "refinery":



## Exercise 3 : TF - IDF

아래 bag-of-words를 기반으로 TF-IDF 기법을 통해 문서를 표상하시오.

-	과일이	길고	노란	먹고	바나나	사과	싫은	저는	좋아요
문서1	0	0	0	1	0	1	1	0	0
문서2	0	0	0	1	1	0	1	0	0
문서3	0	1	1	0	2	0	0	0	0
문서4	1	0	0	0	0	0	0	1	1

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	$\ln(4/(1+1)) = 0.693147$
좋아요	$\ln(4/(1+1)) = 0.693147$

-	과일이	길고	노란	먹고	바나나	사과	싫은	저는	좋아요
문서1	0	0	0	0.287682	0	0.693147	0.287682	0	0
문서2	0	0	0	0.287682	0.287682	0	0.287682	0	0
문서3	0	0.693147	0.693147	0	0.575364	0	0	0	0
문서4	0.693147	0	0	0	0	0	0	0.693147	0.693147

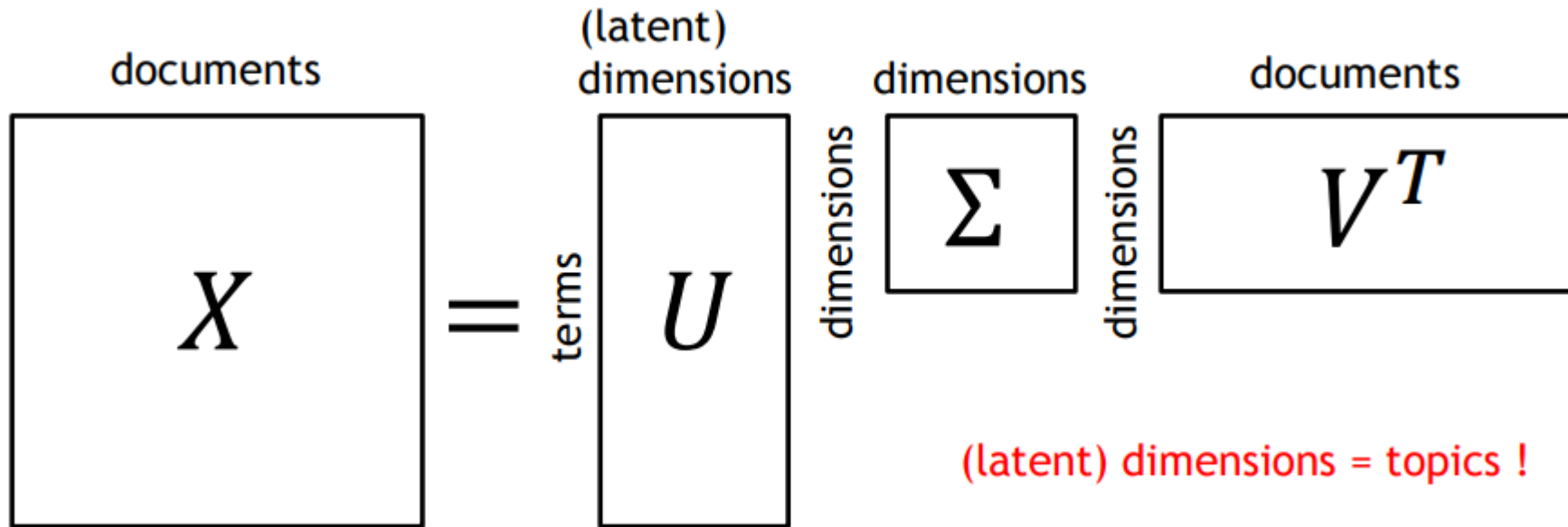
LSA

# Latent Semantic Analysis (LSA)

---

## Latent Semantic Analysis (LSA)

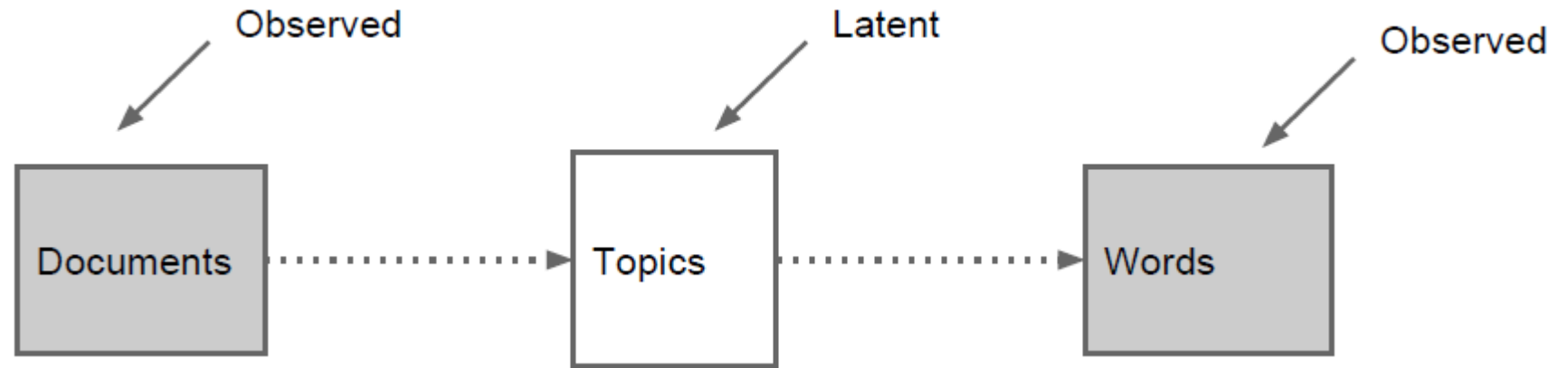
- Applying SVD (Singular Value Decomposition) to document-term matrix
- SVD: Given matrix  $X$ , there exists a factorization of the form  $X=U\Sigma V^T$



# Latent Semantic Analysis (LSA)

---

## Example



### Example:

George Siemens's new blog post (i.e., a document) will likely be about:

MOOCs. (say 80%)

Connectivism (say 30%)

And given that it is 80% about MOOCs,  
it will likely use words:

Massive (90%),  
Coursera (50%),  
Connectivism (85%),  
Analytics (60%)  
etc...

And given that it is 30% about Connectivism  
it will likely use the words:

Network (80%)  
Knowledge (40%)  
Share (30%)  
etc...

# Latent Semantic Analysis (LSA)

3	4	1	0
4	3	0	1
3	4	4	3
0	1	4	3
2	0	3	3
0	1	3	4

=

	To1	To2	To3	To4
Te1	-0.33	-0.53	0.37	-0.14
Te2	-0.32	-0.54	-0.49	0.35
Te3	-0.62	-0.10	0.26	-0.14
Te4	-0.38	0.42	0.30	-0.24
Te5	-0.36	0.25	-0.68	-0.47
Te6	-0.37	0.42	0.02	0.75

X

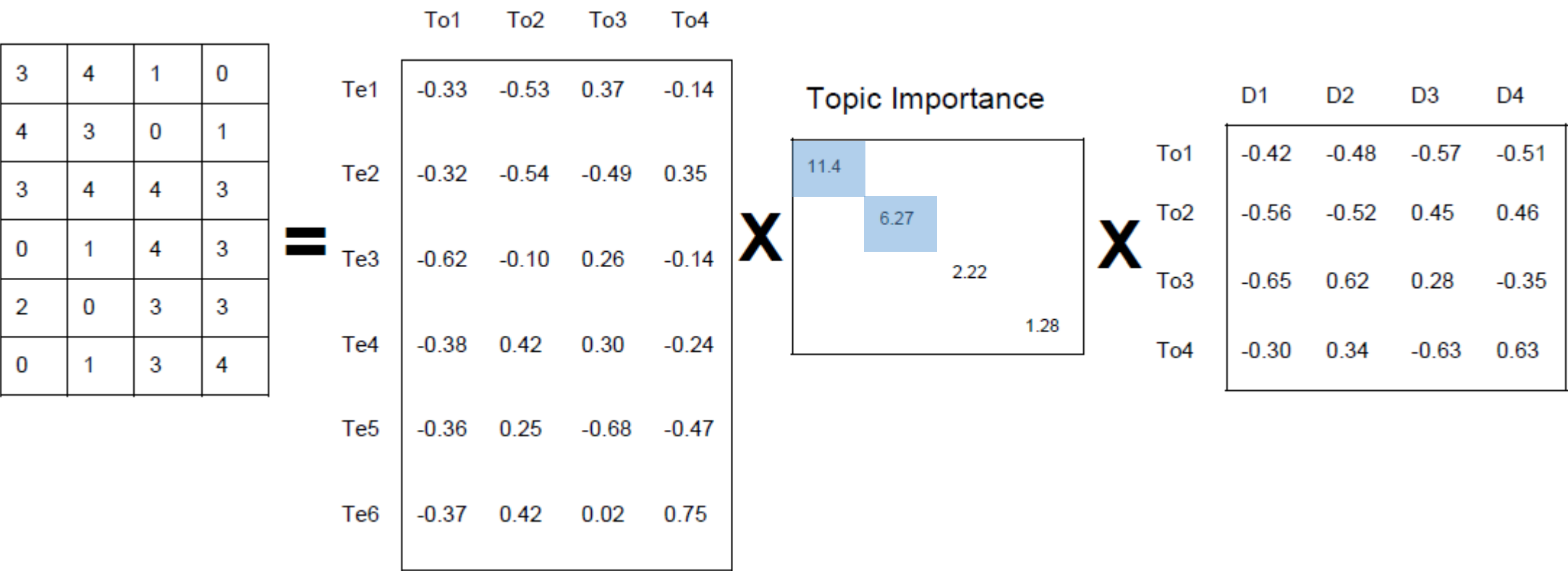
Topic Importance

11.4
6.27
2.22
1.28

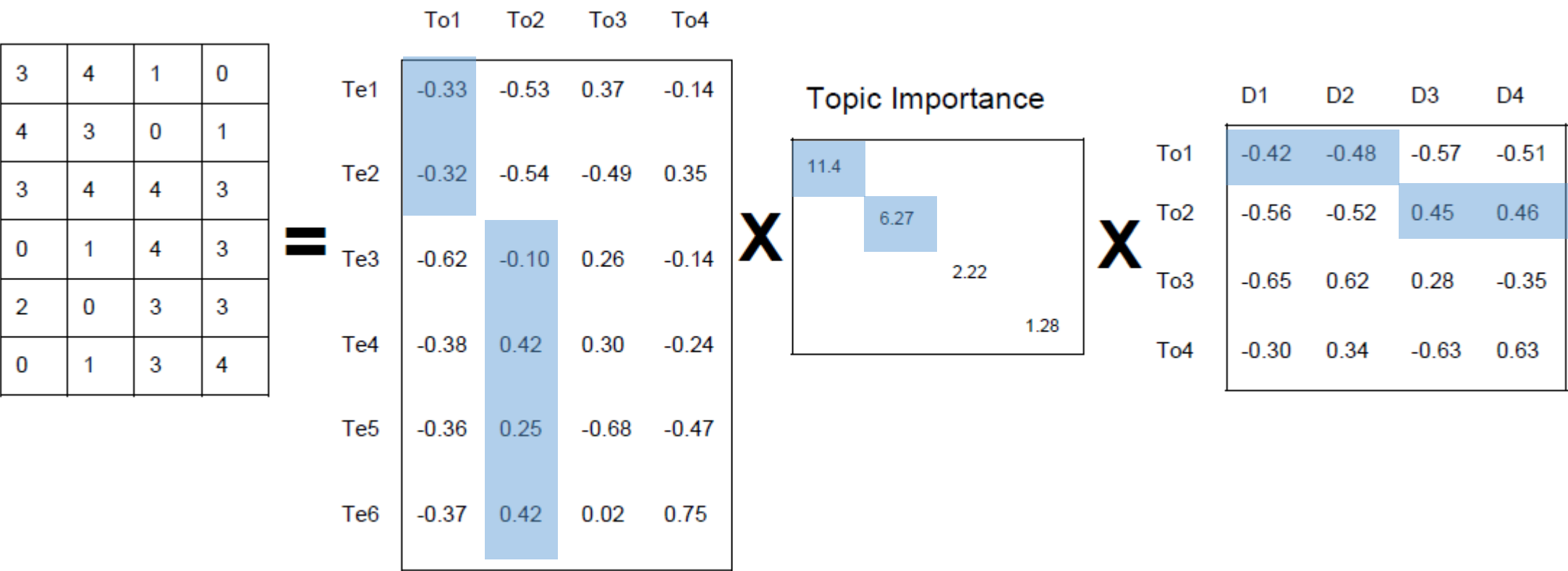
X

	D1	D2	D3	D4
To1	-0.42	-0.48	-0.57	-0.51
To2	-0.56	-0.52	0.45	0.46
To3	-0.65	0.62	0.28	-0.35
To4	-0.30	0.34	-0.63	0.63

# Latent Semantic Analysis (LSA)



# Latent Semantic Analysis (LSA)



# SVD (Singular Value Decomposition, 특이값 분해)

---

## SVD 변환

- 특이값 분해(SVD)는 고유값 분해(eigendecomposition)처럼 행렬을 대각화하는 한 방법
- 특이값 분해가 유용한 이유는 행렬이 정방행렬이든 아니든 관계없이 모든  $m \times n$  행렬에 대해 적용 가능하기 때문
- 실수공간에서 임의의  $m \times n$  행렬에 대한 특이값분해(SVD)는 다음과 같이 정의

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

$A$ :  $m \times n$  rectangular matrix

$U$ :  $m \times m$  orthogonal matrix

$\Sigma$ :  $m \times n$  diagonal matrix

$V$ :  $n \times n$  orthogonal matrix



# SVD (Singular Value Decomposition, 특이값 분해)

## SVD 변환

- U가 orthogonal matrix라고 한다면,  $UU^T = U^T U = I$
- $\Sigma$ 가 diagonal matrix라고 한다면  $\Sigma$ 의 대각 성분을 제외한 나머지 원소의 값은 모두 0

- $\Sigma$ 가 2×2 행렬일 때, 
$$\begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix}$$

만약  $\Sigma$ 가 m×n 행렬일 때, m>n이라면,

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

- 만약  $\Sigma$ 가 m×n 행렬일 때, m<n이라면, 
$$\begin{pmatrix} \sigma_1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 & 0 & \cdots & 0 \\ & & \ddots & & & & \\ 0 & 0 & \cdots & \sigma_m & 0 & \cdots & 0 \end{pmatrix}$$

# SVD (Singular Value Decomposition, 특이값 분해)

---

## SVD 변환의 의미

- 직교하는 벡터 집합에 대하여, 선형 변환 후에 그 크기는 변하지만 여전히 직교할 수 있게 되는 그 직교 집합은 무엇인가?  
그리고 선형 변환 후의 결과는 무엇인가?
- 임의의 벡터  $x$ 와 어떤 행렬  $A$ 를 이용해 선형변환 시킨 결과  $Ax$
- [https://raw.githubusercontent.com/angeloyeo/angeloyeo.github.io/master/pics/2019-08-01\\_SVD/pic1.gif](https://raw.githubusercontent.com/angeloyeo/angeloyeo.github.io/master/pics/2019-08-01_SVD/pic1.gif)
- 임의의 벡터  $x$ 와  $x$ 에 직교하는 벡터  $y$ , 그리고  $x, y$ 를 선형변환한 결과인  $Ax$ 와  $Ay$
- [https://raw.githubusercontent.com/angeloyeo/angeloyeo.github.io/master/pics/2019-08-01\\_SVD/pic2.gif](https://raw.githubusercontent.com/angeloyeo/angeloyeo.github.io/master/pics/2019-08-01_SVD/pic2.gif)

# SVD (Singular Value Decomposition, 특이값 분해)

---

## SVD 변환의 의미

- $Ax$  와  $Ay$ 가 직교하게 되는 경우는 단 한번만 있는 것이 아님
- $Ax$  와  $Ay$ 는  $A$  라는 행렬(즉, 선형변환)을 통해 변환되었을 때, 길이가 조금씩 변했다는 것을 알 수 있음
- 이 값들, 즉 scaling factor 들을 일반적으로는 singular value라고 하고 크기가 큰 값부터  $\sigma_1, \sigma_2, \dots$  등으로 부른다.
- 선형 변환의 관점에서 네 개의 행렬( $A, V, \Sigma, U$ )의 관계를 생각하면 다음과 같다.

$$AV = U\Sigma$$

- 즉, “ $V$ 에 있는 열벡터( $x$  혹은  $y$ )를 행렬  $A$ 를 통해 선형변환 할 때, 그 크기는  $\sigma_1, \sigma_2$ 만큼 변하지만, 여전히 직교하는 벡터들  $u_1, u_2$ 를 찾을 수 있겠느냐?” 라고 묻는 것

# SVD (Singular Value Decomposition, 특이값 분해)

---

## SVD 변환의 목적

- 특이값 분해의 공식을 다시 풀어 써보자면 다음과 같다.

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

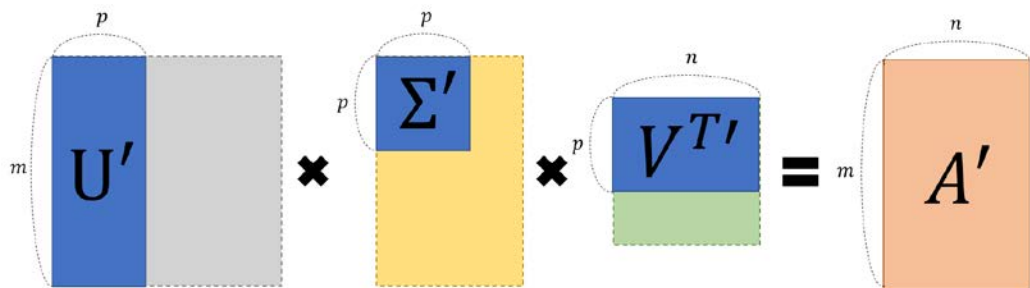
$$= \begin{pmatrix} | & | & & | \\ \vec{u}_1 & \vec{u}_2 & \cdots & \vec{u}_n \\ | & | & & | \end{pmatrix} \begin{pmatrix} \sigma_1 & & & 0 \\ & \sigma_2 & & 0 \\ & & \ddots & 0 \\ & & & \sigma_m & 0 \end{pmatrix} \begin{pmatrix} - & \vec{v}_1^T & - \\ - & \vec{v}_2^T & - \\ & \vdots & \\ - & \vec{v}_n^T & - \end{pmatrix}$$

$$= \sigma_1 \vec{u}_1 \vec{v}_1^T + \sigma_2 \vec{u}_2 \vec{v}_2^T + \cdots + \sigma_m \vec{u}_m \vec{v}_m^T$$

# SVD (Singular Value Decomposition, 특이값 분해)

## SVD 변환의 목적

- $\sigma_1 \vec{u}_1 \vec{v}_1^T$  부분만 놓고 보면 이 행렬의 크기는  $\sigma_1$ 의 값에 의해 정해지게 된다.
- 즉, 우리는 SVD라는 방법을 이용해 A라는 임의의 행렬을 여러 개의 여러 개의 행렬로 분해해서 생각할 수 있는데, 분해된 각 행렬의 원소의 값의 크기는  $\sigma$ 의 값의 크기에 의해 결정된다.
- 다시 말해, SVD를 이용해 임의의 행렬 A를 정보량에 따라 여러 layer로 쪼개서 생각할 수 있게 해준다.
- 즉, 특이 값의 크기에 따라 A의 정보량이 결정되기 때문에 값이 큰 몇 개의 특이 값들을 가지고도 충분히 유용한 정보를 유지할 수 있다.

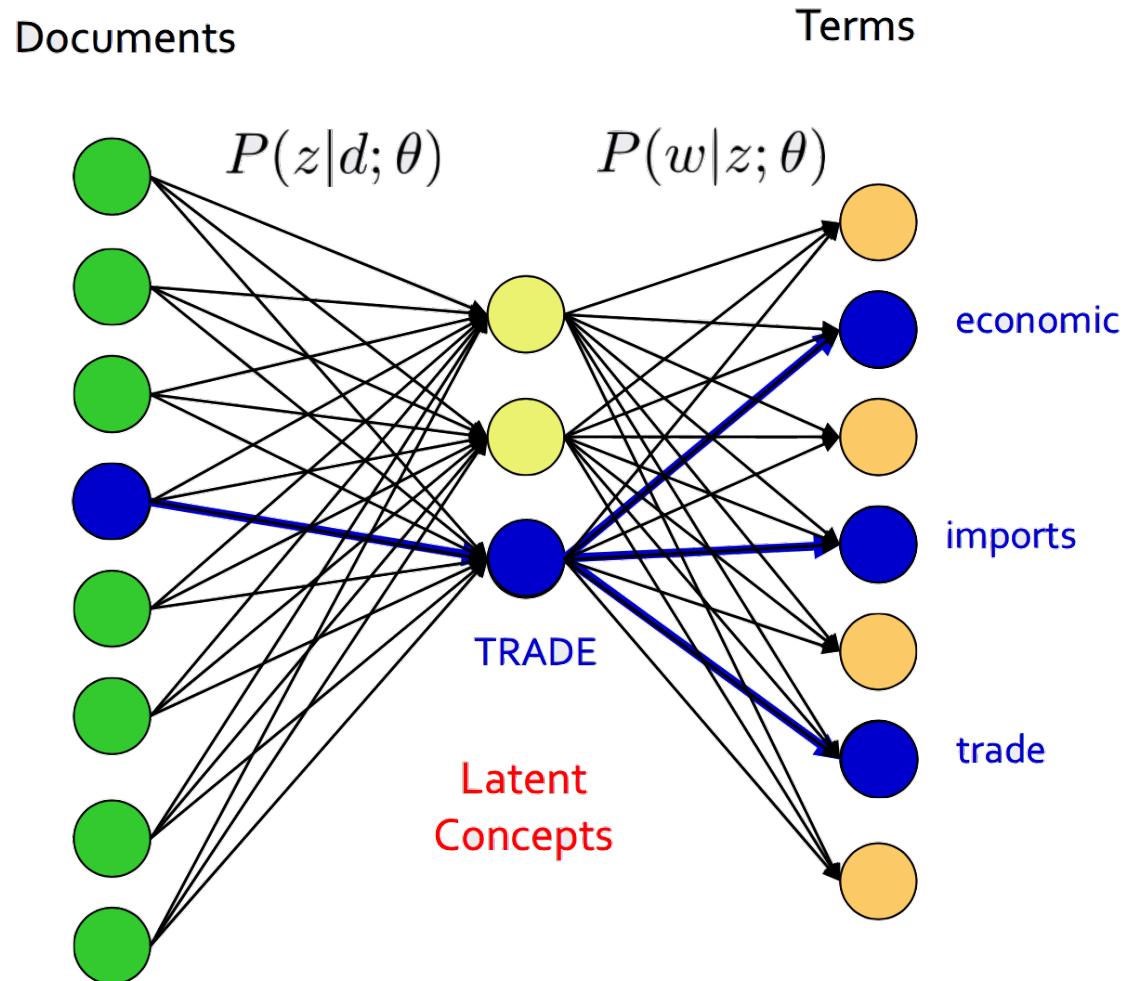


pLSA

# pLSA (probabilistic Latent Semantic Analysis)

## pLSA

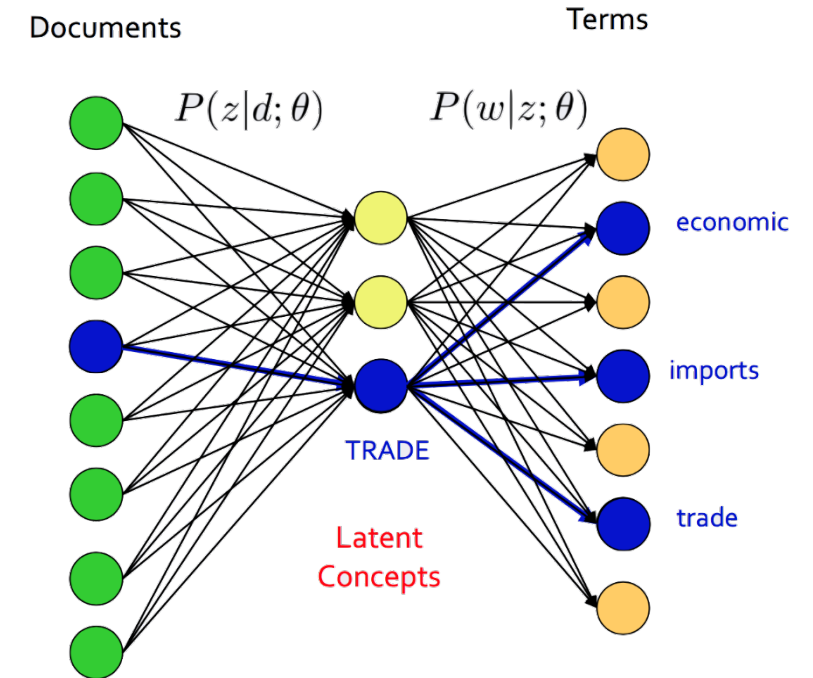
- pLSA는 단어와 문서 사이를 잇는, 우리 눈에 보이지 않는 잠재구조가 있다는 가정 하에 단어와 문서 출현 확률을 모델링한 확률모형



# pLSA (probabilistic Latent Semantic Analysis)

## pLSA

- $P(z|d)$ 는 문서 하나가 주어졌을 때 특정 주제(토픽)가 나타날 확률
- $P(w|z)$ 는 주제가 정해졌을 때 특정 단어가 나타날 확률
- 네번째 문서는 trade라는 주제로만 구성되어 있고, trade라는 주제는 economic, imports, trade 따위의 단어를 선호
- pLSA의 가정에 문제가 없다면 네번째 문서에 해당 단어의 출현 확률이 높을 것임.

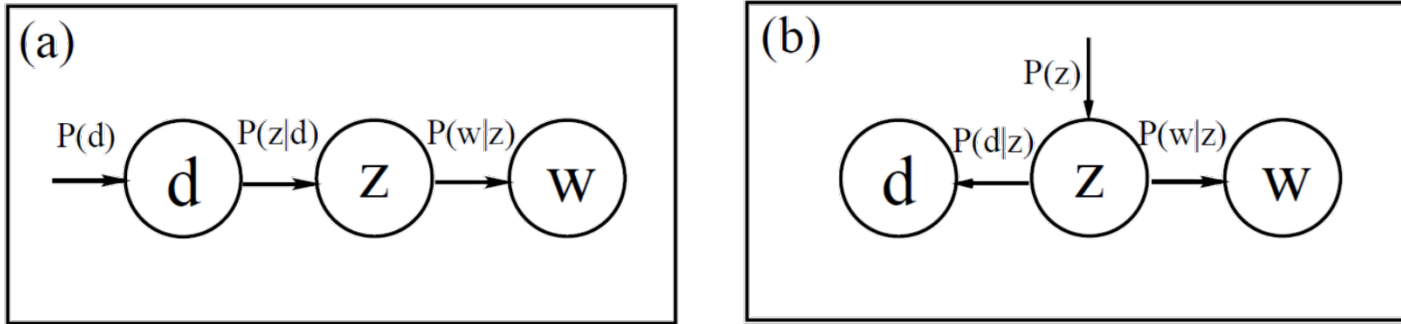




# pLSA (probabilistic Latent Semantic Analysis)

---

## pLSA



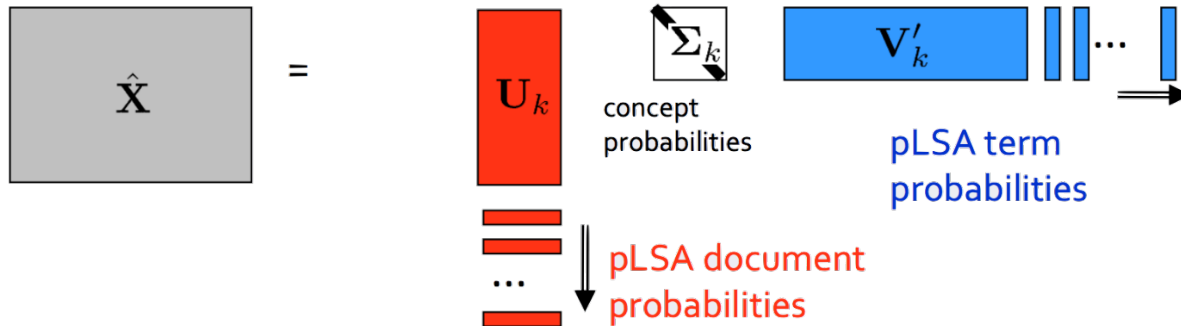
- (a) : 문서를 뽑고, 그 다음 이 문서의 주제를 뽑는다. 마지막으로 해당 주제별로 단어를 뽑음.  
(마치 사람이 글 쓸 때와 유사한 방법)
- (b) : 주제( $z$ )를 뽑은 뒤 이 주제에 해당하는 문서와 단어를 뽑는 방식. pLSA는 이 방식으로 구성되어 있음

# pLSA (probabilistic Latent Semantic Analysis)

## LSA와 pLSA

- LSA와 pLSA는 아예 종류가 다른 모형이지만 개념적으로 연결되는 부분이 있음
- $\Sigma_k$ 의 대각 성분은 토픽 각각이 전체 말뭉치 내에서 얼마나 중요한지 나타내는 가중치가 되는데 이는 pLSA의  $P(z)$ 에 대응

$$\hat{P}_{\text{LSA}}(d, w) = \sum_z P(d|z) P(z) P(w|z)$$



# pLSA (probabilistic Latent Semantic Analysis)

---

## pLSA의 목적식

- pLSA는  $m$ 개 단어,  $n$ 개 문서,  $k$ 개 주제(토픽)에 대해 아래 우도함수를 최대화하는 걸 목표로 함

$$\begin{aligned} L &= \prod_{i=1}^m \prod_{j=1}^n p(w_i, d_j)^{n(w_i, d_j)} \\ &= \prod_{i=1}^m \prod_{j=1}^n \left\{ \sum_{l=1}^k p(d_j | z_l) p(z_l) p(w_i | z_l) \right\}^{n(w_i, d_j)} \end{aligned}$$

- 위 식에서  $n(w_i, d_j)$ 는  $j$ 번째 문서에  $i$ 번째 단어가 등장한 횟수를 나타냄.

# pLSA (probabilistic Latent Semantic Analysis)

---

## pLSA의 학습 : EM 알고리즘

- EM알고리즘은 동시에 최적화할 수 없는 복수의 변수들을 반복적인 방식으로 계산하는 기법
- 우선 모든 값을 랜덤으로 초기화합니다.
- 이후 하나의 파라미터를 고정시킨 다음에 다른 파라미터를 업데이트하고,  
이후 단계에선 업데이트된 파라미터로 고정시킨 파라미터를 다시 업데이트합니다.

# pLSA (probabilistic Latent Semantic Analysis)

## pLSA의 학습 : EM 알고리즘

- E-Step: Posterior probability of latent variables (concepts)

$$p(z|d, w) = \frac{P(z)P(d|z)P(w|z)}{\sum_{z' \in Z} P(z')P(d|z')P(w|z')}$$

Probability that the occurrence of term  $w$  in document  $d$  can be “explained” by concept  $z$

- M-Step: Parameter estimation based on “completed” statistics

$$P(w|z) = \frac{\sum_{d \in D} n(d, w)P(z|d, w)}{\sum_{d \in D, w' \in W} n(d, w')P(z|d, w')}$$

how often is term  $w$  associated with concept  $z$  ?

$$P(d|z) = \frac{\sum_{w \in W} n(d, w)P(z|d, w)}{\sum_{d' \in D, w \in W} n(d', w)P(z|d', w)}$$

how often is document  $d$  associated with concept  $z$  ?

$$P(z) = \frac{\sum_{d \in D, w \in W} n(d, w)P(z|d, w)}{\sum_{d \in D, w \in W} n(d, w)}$$

how prevalent is the concept  $z$  ?

# pLSA (probabilistic Latent Semantic Analysis)

---

분석 예시 (Term-Document Matrix)

word	Doc 1	Doc 2	Doc 3	Doc 4	Doc 5	Doc 6
Baseball	1	2	0	0	0	0
Basketball	3	1	0	0	0	0
Boxing	2	0	0	0	0	0
Money	3	3	2	3	2	4
Interest	0	0	3	2	0	0
Rate	0	0	4	1	0	0
Democrat	0	0	0	0	4	3
Republican	0	0	0	0	2	1
Cocus	0	0	0	0	3	2
President	0	0	1	0	2	3

# pLSA (probabilistic Latent Semantic Analysis)

---

분석 예시

(P(z))	Topic 1	Topic 2	Topic 3
	0.456	0.281	0.263

P(d z)	Docs	Topic 1	Topic 2	Topic 3
	Doc 1	0.000	0.000	0.600
	Doc 2	0.000	0.000	0.400
	Doc 3	0.000	0.625	0.000
	Doc 4	0.000	0.375	0.000
	Doc 5	0.500	0.000	0.000
	Doc 6	0.500	0.000	0.000

# pLSA (probabilistic Latent Semantic Analysis)

---

분석 예시

$P(w|z)$

Terms	Topic 1	Topic 2	Topic 3
Baseball	0.000	0.000	0.200
Basketball	0.000	0.000	0.267
Boxing	0.000	0.000	0.133
Money	0.231	0.313	0.400
Interest	0.000	0.312	0.000
Rate	0.000	0.312	0.000
Democrat	0.269	0.000	0.000
Republican	0.115	0.000	0.000
Cocus	0.192	0.000	0.000
President	0.192	0.063	0.000



Q&A

감사합니다.