

---

# Topic Extraction

---

# Problem

---

Identifying topics that best describes a set of documents

# Sentence

---

I love eating pizza while looking at charts.

# Sentence

---

I love eating pizza while looking at charts.

77% - Topic 1

27% - Topic 2

# Topic

---

Topic 1: 22% Pizza, 15% mozzarella, 11% Italian, ...

Topic 2: 32% Data Analysis, 21% Statistics, 19% charts, ...



# LDA



# LDA

---

LDA represents documents as a mixture of topics.

A topic is a mixture of words. If a word has high probability of being in a topic  $t$ , all the documents having the word will be more strongly associated with  $t$  as well.

Similarly, if  $w$  is not very probable to be in  $t$ , the documents which contain the  $w$  will be having very low probability of being in  $t$ , because rest of the words in  $d$  will belong to some other topic and hence  $d$  will have a higher probability for those topic.

# Algorithm

---

Select number of topic

LDA will go through each of the words in each of the documents, and it will randomly assign the word to one of the  $K$  topics.

We now have a topic representations (how the words are distributed in each topic) and documents represented in terms of topics but it's random.



# Sentence

---

To improve this representation LDA will analyze:

- For each single document : the percentage of words within the document that were assigned to a particular topic.
- For each word in the document: analyze over all the documents, what is the percentage of times that particular word has been assigned to a particular topic.

# LDA

---

For each document  $d$ , go through each word  $w$  and compute:

1.  $p(\text{topic } t \mid \text{document } d)$ : the proportion of words in document  $d$  that are assigned to topic  $t$ .
2.  $p(\text{word } w \mid \text{topic } t)$ : the proportion of assignments to topic  $t$  over all documents that come from this word  $w$ . Tries to capture how many documents are in topic  $t$  because of word  $w$ .

- Update the probability for the word  $w$  belonging to topic  $t$ , as

$$p(\text{word } w \text{ with topic } t) = p(\text{topic } t \mid \text{document } d) * p(\text{word } w \mid \text{topic } t)$$

# LDA

LDA will therefore be calculating:

1)  $p(\text{topic } t \mid \text{document } d)$  = percentage of words in a document  $d$  that are currently assigned to topic  $t$ .

2)  $p(\text{word } w \mid \text{topic } t)$  = percentage of times the word  $w$  was assigned to topic  $t$  over all documents.

LDA will decide to move a word  $w$  from topic  $A$  to topic  $B$  when:

$$p(\text{topic } A \mid \text{document } d) * p(\text{word } w \mid \text{topic } A) < p(\text{topic } B \mid \text{document } d) * p(\text{word } w \mid \text{topic } B)$$

After a while, LDA "converges" to a more optimal state, where topic representations and documents represented in terms of these topics are ok.

# Sentence

---

The problem with LDA is that the posterior probability we need to calculate in order to reassign words to topics is very difficult to compute. Therefore researchers use approximation techniques to find what this posterior probability is.

Generally algorithms for approximating posterior probability:

- sampling approaches (MCMC)

- optimization approaches (Variational Inference seeks to optimize a simplified parametric distribution to be close in Kullback-Leibler divergence)

Now, despite the benefits Variational Inference brings. Large scale data analysis can still be difficult. What many groups have done is to use batch variational inference, where there is a constant iteration between analyzing each observation

# Sentence

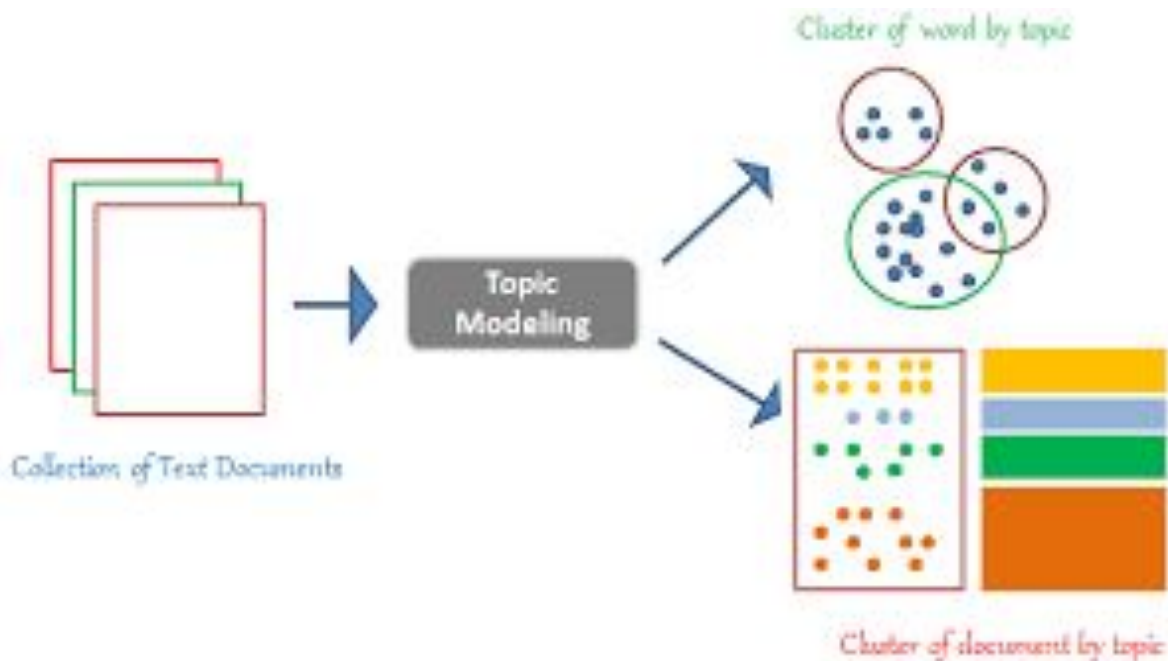
---

Now, despite the benefits Variational Inference brings. Large scale data analysis can still be difficult.

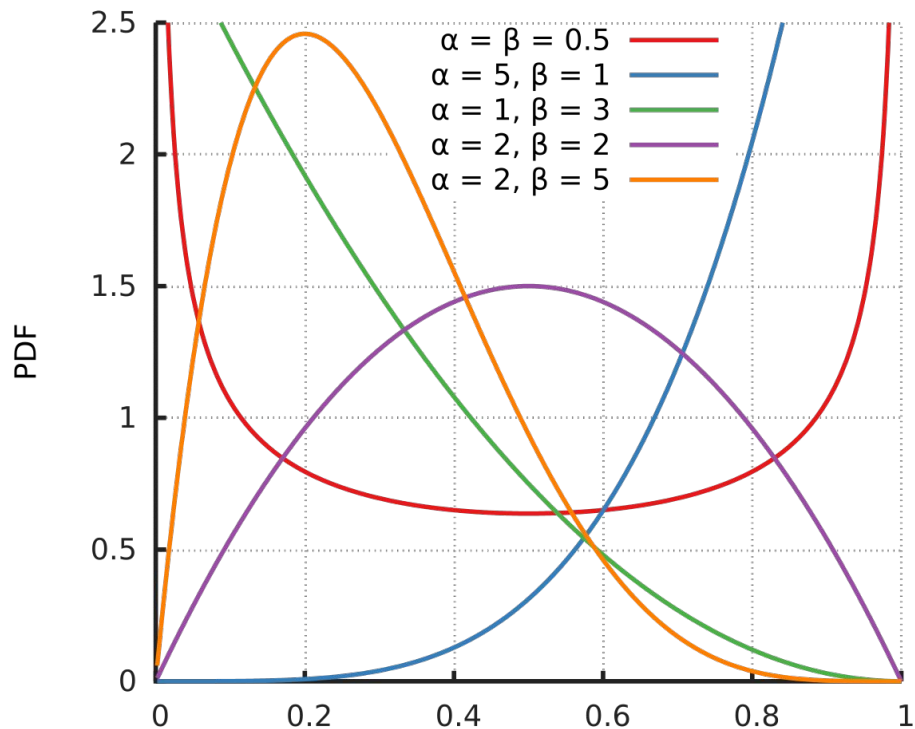
Online LDA is based on online stochastic optimization, which has shown to produce good parameter estimates dramatically faster than batch algorithms on large datasets.

Online stochastic optimization in LDA is about finding a balance between exploiting the knowledge gained on a particular topic assignment, and exploring new topic assignments.

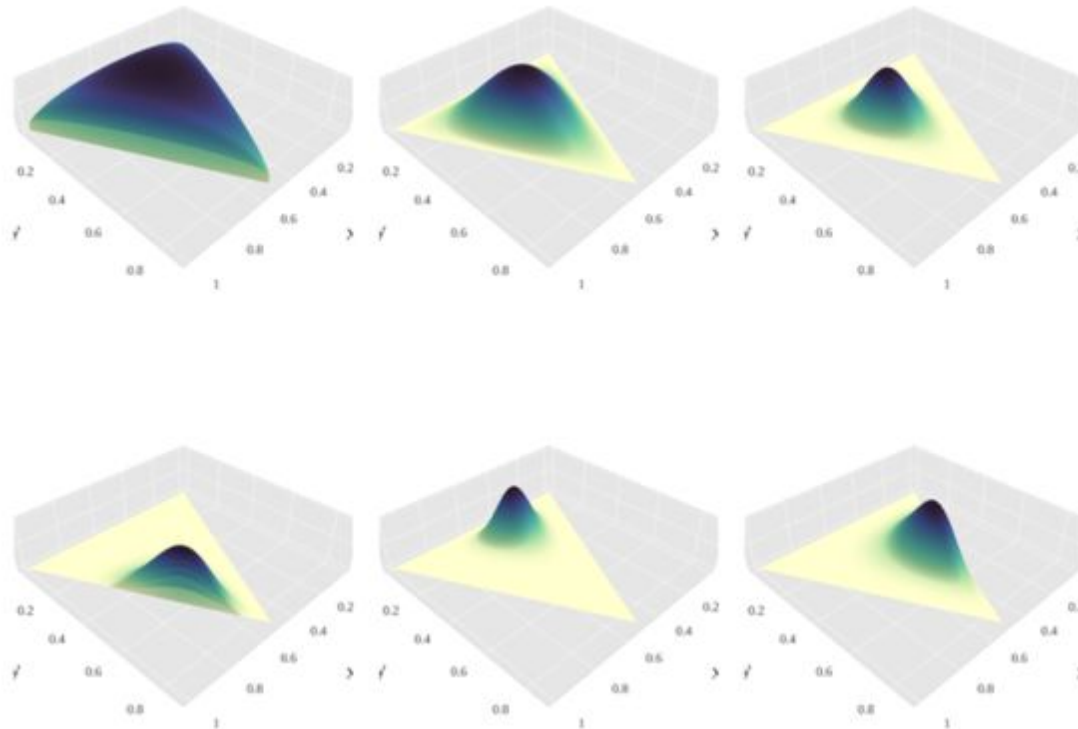
# Latent Dirichlet Allocation



# Beta distribution

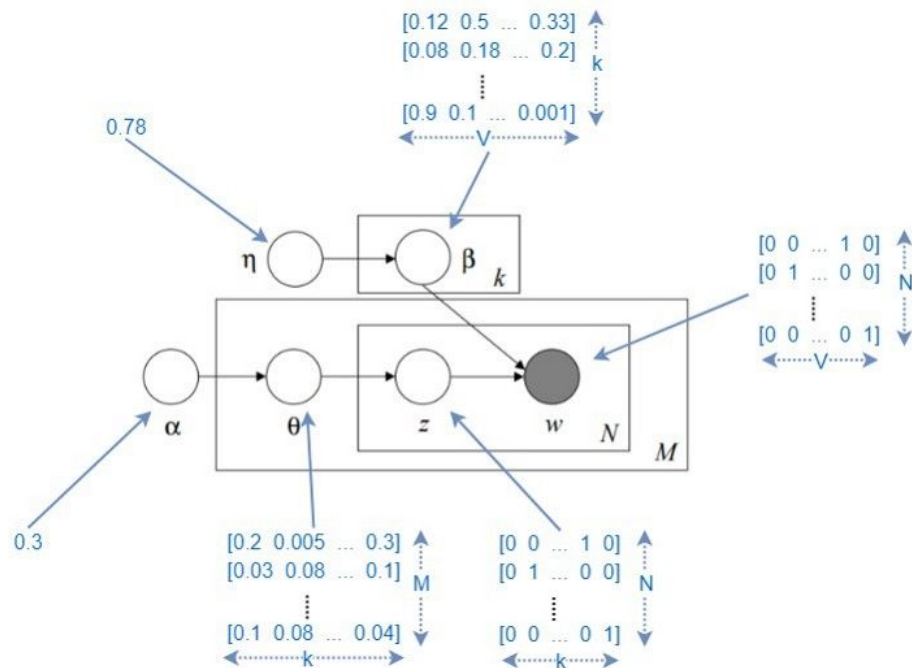


# Dirichlet distribution





# Dirichlet distribution



$$P(\theta_{1:M}, \mathbf{z}_{1:M}, \beta_{1:k} | \mathcal{D}; \alpha_{1:M}, \eta_{1:k})$$



# NMF



# Matrix factorization \*decomposition)

	Item			
	W	X	Y	Z
A		4.5	2.0	
B	4.0		3.5	
C		5.0		2.0
D		3.5	4.0	1.0

Rating Matrix

=

A	1.2	0.8
B	1.4	0.9
C	1.5	1.0
D	1.2	0.8

User Matrix

X

	W	X	Y	Z
	1.5	1.2	1.0	0.8
	1.7	0.6	1.1	0.4

Item Matrix

# Matrix factorization \*decomposition)

	Item			
	W	X	Y	Z
A		4.5	2.0	
B	4.0		3.5	
C		5.0		2.0
D		3.5	4.0	1.0

Rating Matrix

=

A	1.2	0.8
B	1.4	0.9
C	1.5	1.0
D	1.2	0.8

User Matrix

X

	W	X	Y	Z
	1.5	1.2	1.0	0.8
	1.7	0.6	1.1	0.4

Item Matrix

# Non-Negative Matrix factorization

---

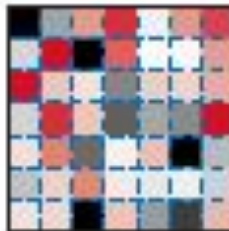
When component should not be negative

# Matrix factorization

PCA



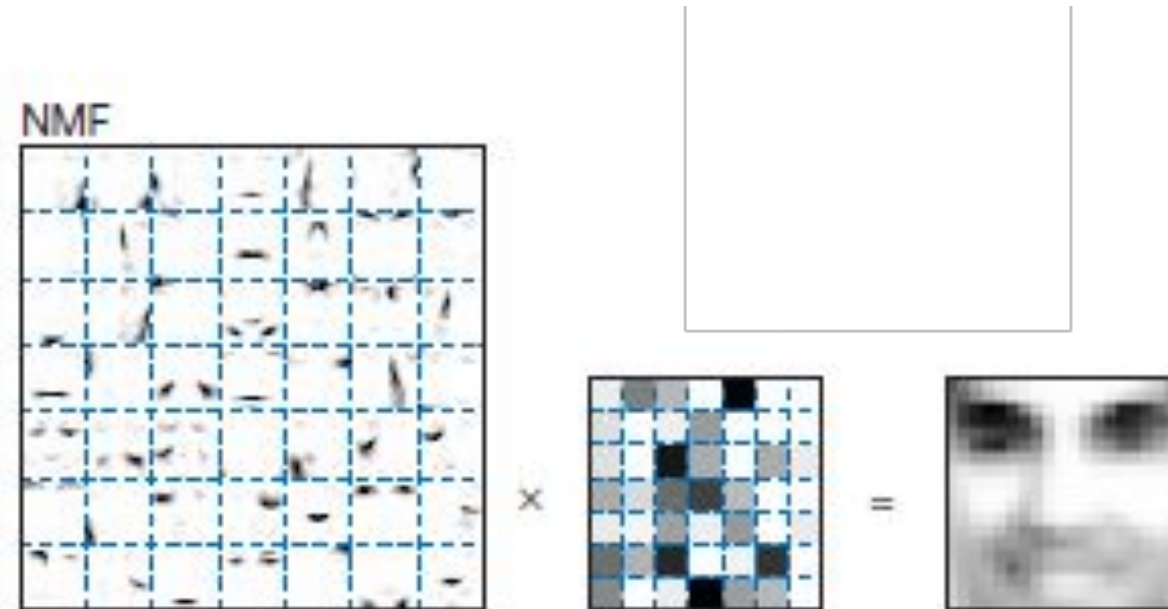
$\times$



$=$



# Non-Negative Matrix factorization



# Topic Modelling

---

decompose a term-document matrix,

each column represented a document

each element in the document represented the weight of a certain word



# Topic Modelling

---

The word "eat" would be likely to appear in food-related articles, and therefore co-occur with words like "tasty" and "food". Therefore, these words would probably be grouped together into a "food" component vector, and each article would have a certain weight of the "food" topic

# Topic Modelling

---

Decompose the term-document matrix in topics

decompose each document into a weighted sum of topics.



# LSA



# Latent Semantic Analysis

---

The core idea is to take a matrix of what we have — documents and terms — and decompose it into a separate document-topic matrix and a topic-term matrix.

The first step is generating our document-term matrix. Given  $m$  documents and  $n$  words in our vocabulary, we can construct an  $m \times n$  matrix  $A$  in which each row represents a document and each column represents a word.

# Latent Semantic Analysis

---

In practice, raw counts do not work well because they do not account for the *significance* of each word in the document. For example, the word “nuclear” probably informs us more about the topic(s) of a given document than the word “test.”

# TF-IDF

The diagram illustrates the TF-IDF formula with the following components and annotations:

- tf-idf score** (red text) points to the variable  $w_{i,j}$ .
- # occurrences of term in document** (green text) points to the term frequency  $tf_{i,j}$ .
- # documents containing word** (purple text) points to the denominator  $df_j$ .
- # total documents** (blue text) points to the numerator  $N$ .

$$w_{i,j} = tf_{i,j} \times \log \frac{N}{df_j}$$

# Truncated SVD

---

Now we need to find out our latent *topics*. Here's the thing: in all likelihood,  $A$  is very sparse, very noisy, and very redundant across its many dimensions. As a result, to find the few latent topics that capture the relationships among the words and documents, we want to perform dimensionality reduction on  $A$ .

# Truncated SVD

---

**Truncated SVD.** SVD:

factorizes any matrix  $M$  into the product of 3 separate matrices:  $M=U*S*V$ , where  $S$  is a diagonal matrix of the singular values of  $M$ .

SVD reduces dimensionality by selecting only the  $t$  largest singular values, and only keeping the first  $t$  columns of  $U$  and  $V$ . In this case,  $t$  is a hyperparameter we can select and adjust number of topics



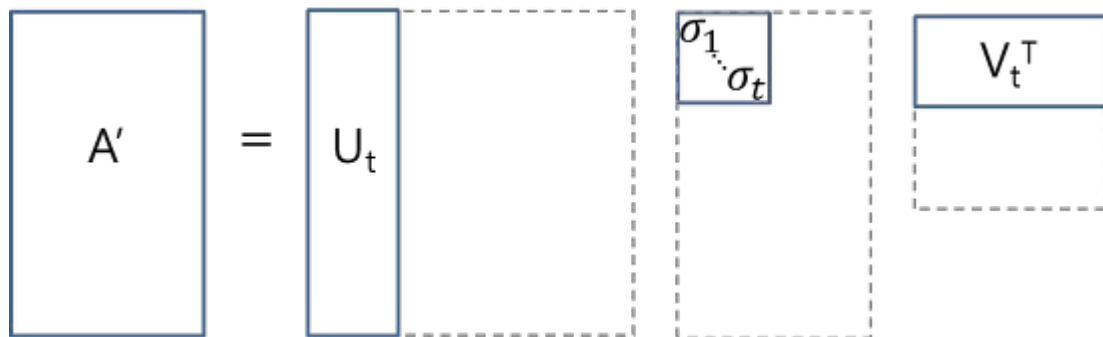
## Truncated SVD

---

$$A \approx U_t S_t V_t^T$$

# Truncated SVD

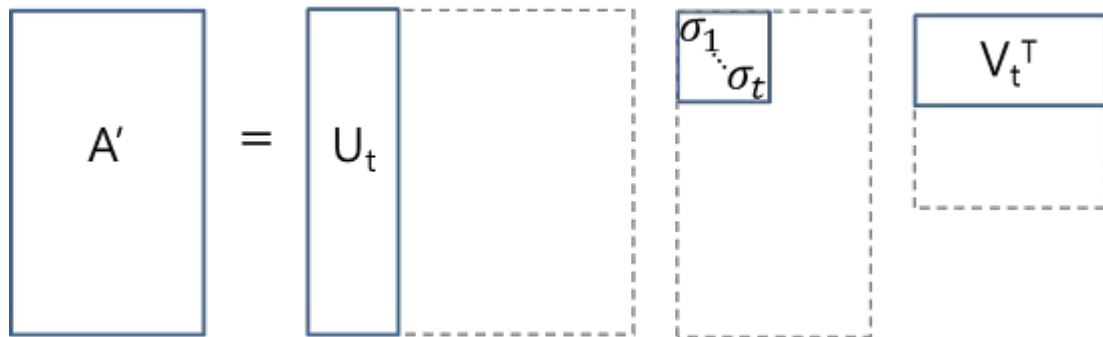
$$A \approx U_t S_t V_t^T$$



# Truncated SVD

$U$ : document-topic matrix, and

$V$ : term-topic matrix.



The diagram illustrates the Truncated SVD equation  $A' = U_t \Sigma V_t^T$ . It consists of four main components arranged horizontally: a solid blue rectangle labeled  $A'$ , an equals sign, a solid blue rectangle labeled  $U_t$ , a dashed blue rectangle containing a solid blue rectangle labeled  $\sigma_1 \dots \sigma_t$ , and a dashed blue rectangle containing a solid blue rectangle labeled  $V_t^T$ . The dashed boxes indicate that the matrices  $\Sigma$  and  $V_t^T$  are truncated to their top  $t$  components.

In both  $U$  and  $V$ , the columns correspond to one of our  $t$  topics. In  $U$ , rows represent document vectors expressed in terms of topics; in  $V$ , rows represent term vectors expressed in terms of topics.