Topic Models

Some slides are based on lectures from

Bing Liu, Ata Kaban, and David Blei

Admin Notes

Schedule

Today and next two classes (group meetings first hour, lecture immediately following)

- Today: Topic models (Pt. I)
- 23 Oct.: Topic model (Pt. II) / Semi-supervised learning (Pt. I)
- 30 Oct.: Semi-supervised learning (Pt. II)

Last four lectures (review, projects, and final)

- 6 Nov.: Sultan's review (1:30-3:15)
  Open session (1:30 – 4:15)
  (Project – Shakarian; Final – Sultan)
- 13, 20 Nov.: Project presentations
- 27 Nov.: No class
- 4 Dec.: Final exam (during normal class time)

Today

- NB Classification (quick review to warm-up your Bayesian probability theory skills)
- NB for text classification (Have document topics in historical data, need to classify unseen document as having a specific topic)
- Latent Topic Models (What if we don’t have topics in historical corpus? -- Need to learn from data)

Naïve Bayes Classification

Bayesian classification

- Probabilistic view: Supervised learning can naturally be studied from a probabilistic point of view.
- Let A_1 through A_k be attributes with discrete values. The class is C.
- Given a test example *d* with observed attribute values a_1 through a_k.
- Classification is basically to compute the following posteriori probability. The prediction is the class c_j such that

$$\Pr(C = c_j \mid A_1 = a_1, \ldots, A_{|A|} = a_{|A|})$$

  is maximal

Apply Bayes' Rule

$$\Pr(C = c_j \mid A_1 = a_1, \ldots, A_{|A|} = a_{|A|})$$

$$= \frac{\Pr(A_1 = a_1, \ldots, A_{|A|} = a_{|A|} \mid C = c_j)\Pr(C = c_j)}{\Pr(A_1 = a_1, \ldots, A_{|A|} = a_{|A|})}$$

$$= \frac{\Pr(A_1 = a_1, \ldots, A_{|A|} = a_{|A|} \mid C = c_j)\Pr(C = c_j)}{\sum_{r=1}^{|C|} \Pr(A_1 = a_1, \ldots, A_{|A|} = a_{|A|} \mid C = c_r)\Pr(C = c_r)}$$

- $\Pr(C=c_j)$ is the class *prior* probability: easy to estimate from the training data.

Computing probabilities

- The denominator $P(A_1=a_1,...,A_k=a_k)$ is irrelevant for decision making since it is the same for every class.
- We only need $P(A_1=a_1,...,A_k=a_k \mid C=c_i)$, which can be written as

  $Pr(A_1=a_1|A_2=a_2,...,A_k=a_k, C=c_j)* Pr(A_2=a_2,...,A_k=a_k \mid C=c_j)$
- Recursively, the second factor above can be written in the same way, and so on.
- Now an assumption is needed.

Conditional independence assumption

- All attributes are conditionally independent given the class $C = c_j$.
- Formally, we assume,

  $\Pr(A_1=a_1 \mid A_2=a_2, \ldots, A_{|A|}=a_{|A|}, C=c_j) = \Pr(A_1=a_1 \mid C=c_j)$

  and so on for A_2 through $A_{|A|}$. I.e.,

$$\Pr(A_1 = a_1,\ldots, A_{|A|} = a_{|A|} \mid C = c_i) = \prod_{i=1}^{|A|} \Pr(A_i = a_i \mid C = c_j)$$

Final naïve Bayesian classifier

$$\Pr(C = c_j \mid A_1 = a_1, \ldots, A_{|A|} = a_{|A|})$$

$$= \frac{\Pr(C = c_j) \prod_{i=1}^{|A|} \Pr(A_i = a_i \mid C = c_j)}{\sum_{r=1}^{|C|} \Pr(C = c_r) \prod_{i=1}^{|A|} \Pr(A_i = a_i \mid C = c_r)}$$

- We are done!
- How do we estimate $P(A_i = a_i \mid C = c_j)$? Easy!.

Classify a test instance

- If we only need a decision on the most probable class for the test instance, we only need the numerator as its denominator is the same for every class.
- Thus, given a test example, we compute the following to decide the most probable class for the test instance

$$c = \arg\max_{c_j} \Pr(c_j) \prod_{i=1}^{|A|} \Pr(A_i = a_i \mid C = c_j)$$

An example

- Compute all probabilities required for classification

A	B	C
m	b	t
m	s	t
g	q	t
h	s	t
g	q	t
g	q	f
g	s	f
h	b	f
h	q	f
m	b	f

$Pr(C = t) = 1/2$, $Pr(C = f) = 1/2$

$Pr(A=m \mid C=t) = 2/5$
$Pr(A=m \mid C=f) = 1/5$
$Pr(B=b \mid C=t) = 1/5$
$Pr(B=b \mid C=f) = 2/5$

$Pr(A=g \mid C=t) = 2/5$
$Pr(A=g \mid C=f) = 2/5$
$Pr(B=s \mid C=t) = 2/5$
$Pr(B=s \mid C=f) = 1/5$

$Pr(A=h \mid C=t) = 1/5$
$Pr(A=h \mid C=n) = 2/5$
$Pr(B=q \mid C=t) = 2/5$
$Pr(B=q \mid C=f) = 2/5$

Now we have a test example:

$A = m \quad B = q \quad C = ?$

An Example (cont …)

- For C = t, we have

$$\Pr(C=t)\prod_{j=1}^{2}\Pr(A_j=a_j\,|\,C=t)=\frac{1}{2}\times\frac{2}{5}\times\frac{2}{5}=\frac{2}{25}$$

- For class C = f, we have

$$\Pr(C=f)\prod_{j=1}^{2}\Pr(A_j=a_j\,|\,C=f)=\frac{1}{2}\times\frac{1}{5}\times\frac{2}{5}=\frac{1}{25}$$

- C = t is more probable. t is the final class.

Additional issues

- **Numeric attributes**: Naïve Bayesian learning assumes that all attributes are categorical. Numeric attributes need to be discretized.
- **Zero counts**: An particular attribute value never occurs together with a class in the training set. We need smoothing.

$$\Pr(A_i = a_i \mid C = c_j) = \frac{n_{ij} + \lambda}{n_j + \lambda n_i}$$

- **Missing values**: Ignored

On naïve Bayesian classifier

- Advantages:
  - Easy to implement
  - Very efficient
  - Good results obtained in many applications
- Disadvantages
  - Assumption: class conditional independence, therefore loss of accuracy when the assumption is seriously violated (those highly correlated data sets)

NB for Text Classification

Text classification/categorization

- Due to the rapid growth of online documents in organizations and on the Web, automated document classification has become an important problem.
- Techniques discussed previously can be applied to text classification, but they are not as effective as the next three methods.
- We first study a naïve Bayesian method specifically formulated for texts, which makes use of some text specific features.
- However, the ideas are similar to the preceding method.

Probabilistic framework

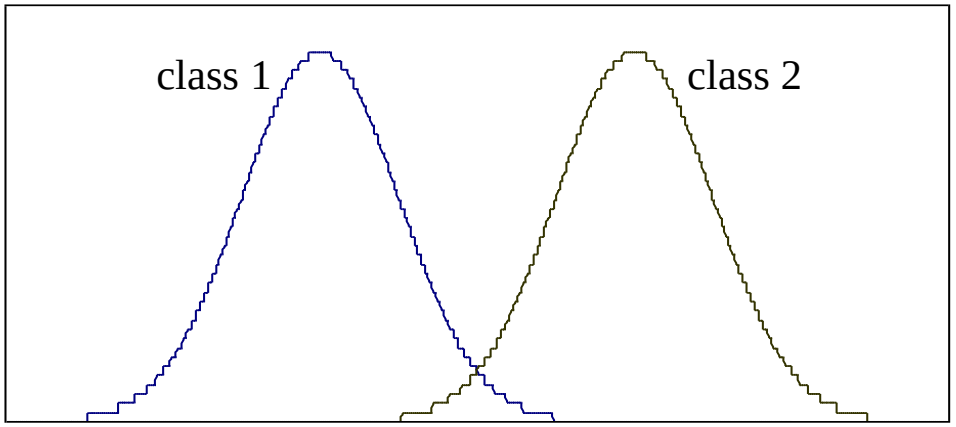
- Generative model: Each document is generated by a parametric distribution governed by a set of hidden parameters.
- The generative model makes two assumptions
  - The data (or the text documents) are generated by a mixture model,
  - There is one-to-one correspondence between mixture components and document classes.

Mixture model

- A **mixture model** models the data with a number of statistical distributions.
  - Intuitively, each distribution corresponds to a data cluster and the parameters of the distribution provide a description of the corresponding cluster.
- Each distribution in a mixture model is also called a **mixture component**.
- The distribution/component can be of any kind

An example

- The figure shows a plot of the **probability density function** of a 1-dimensional data set (with two classes) generated by
  - a mixture of two Gaussian distributions,
  - one per class, whose parameters (denoted by θ_i) are the mean (μ_i) and the standard deviation (σ_i), i.e., $\theta_i = (\mu_i, \sigma_i)$.



Mixture model (cont …)

- Let the number of mixture components (or distributions) in a mixture model be *K*.
- Let the *j*th distribution have the parameters θ_j.
- Let Θ be the set of parameters of all components, Θ = $\{\varphi_1, \varphi_2, \ldots, \varphi_K, \theta_1, \theta_2, \ldots, \theta_K\}$, where φ_j is the *mixture weight* (or *mixture probability*) of the mixture component *j* and θ_j is the parameters of component *j*.
- How does the model generate documents?

Document generation

- Due to one-to-one correspondence, each class corresponds to a mixture component. The mixture weights are *class prior probabilities*, i.e., $\varphi_j = \Pr(c_j|\Theta)$.
- The mixture model generates each document d_i by:
  - first selecting a mixture component (or class) according to class prior probabilities (i.e., mixture weights), $\varphi_j = \Pr(c_j|\Theta)$.
  - then having this selected mixture component (c_j) generate a document d_i according to its parameters, with distribution $\Pr(d_i|c_j; \Theta)$ or more precisely $\Pr(d_i|c_j; \theta_j)$.

$$\Pr(d_i \mid \Theta) = \sum_{j=1}^{|C|} \Pr(c_j \mid \Theta) \Pr(d_i \mid c_j; \Theta) \qquad (23)$$

Model text documents

- The naïve Bayesian classification treats each document as a “bag of words”. The generative model makes the following further assumptions:
  - Words of a document are generated independently of context given the class label. The familiar naïve Bayes assumption used before.
  - The probability of a word is independent of its position in the document. The document length is chosen independent of its class.

Multinomial distribution

- With the assumptions, each document can be regarded as generated by a multinomial distribution.
- In other words, each document is drawn from a multinomial distribution of words with as many independent trials as the length of the document.
- The words are from a given vocabulary $V = \{w_1, w_2, \ldots, w_{|V|}\}$.

Use probability function of multinomial distribution

$$\Pr(d_i \mid c_j;\Theta) = \Pr(|d_i|)\,|d_i|!\prod_{t=1}^{|V|}\frac{\Pr(w_t \mid c_j;\Theta)^{N_{ti}}}{N_{ti}!} \quad (24)$$

where N_{ti} is the number of times that word w_t occurs in document d_i and

$$\sum_{t=1}^{|V|} N_{it} = |d_i| \qquad \sum_{t=1}^{|V|} \Pr(w_t \mid c_j;\Theta) = 1. \quad (25)$$

Parameter estimation

- The parameters are estimated based on empirical counts.

$$\Pr(w_t \mid c_j; \hat{\Theta}) = \frac{\sum_{i=1}^{|D|} N_{ti} \Pr(c_j \mid d_i)}{\sum_{s=1}^{|V|} \sum_{i=1}^{|D|} N_{si} \Pr(c_j \mid d_i)}. \quad (26)$$

- In order to handle 0 counts for infrequent occurring words that do not appear in the training set, but may appear in the test set, we need to smooth the probability. *Lidstone* smoothing, $0 \leq \lambda \leq 1$

$$\Pr(w_t \mid c_j; \hat{\Theta}) = \frac{\lambda + \sum_{i=1}^{|D|} N_{ti} \Pr(c_j \mid d_i)}{\lambda |V| + \sum_{s=1}^{|V|} \sum_{i=1}^{|D|} N_{si} \Pr(c_j \mid d_i)}. \quad (27)$$

Parameter estimation (cont …)

- Class prior probabilities, which are mixture weights φ_j, can be easily estimated using training data

$$\Pr(c_j \mid \hat{\Theta}) = \frac{\sum_{i=1}^{|D|} \Pr(c_j \mid d_i)}{|D|} \tag{28}$$

Classification

- Given a test document d_i, from Eq. (23) (27) and (28)

$$\Pr(c_j \mid d_i; \hat{\Theta}) = \frac{\Pr(c_j \mid \hat{\Theta}) \Pr(d_i \mid c_j; \hat{\Theta})}{\Pr(d_i \mid \hat{\Theta})}$$

$$= \frac{\Pr(c_j \mid \hat{\Theta}) \prod_{k=1}^{|d_i|} \Pr(w_{d_i,k} \mid c_j; \hat{\Theta})}{\sum_{r=1}^{|C|} \Pr(c_r \mid \hat{\Theta}) \prod_{k=1}^{|d_i|} \Pr(w_{d_i,k} \mid c_r; \hat{\Theta})}$$

where $w_{d_i,k}$ is the word in position k of document d_i. If the final classifier is to classify each document into a single class, then the class with the highest posterior probability is selected:

$$\arg\max_{c_j \in C} \Pr(c_j \mid d_i; \hat{\Theta}) \qquad (30)$$

Discussions

- Most assumptions made by naïve Bayesian learning are violated to some degree in practice.
- Despite such violations, researchers have shown that naïve Bayesian learning produces very accurate models.
  - The main problem is the mixture model assumption. When this assumption is seriously violated, the classification performance can be poor.
- Naïve Bayesian learning is extremely efficient.

Latent Topic Models

Overview

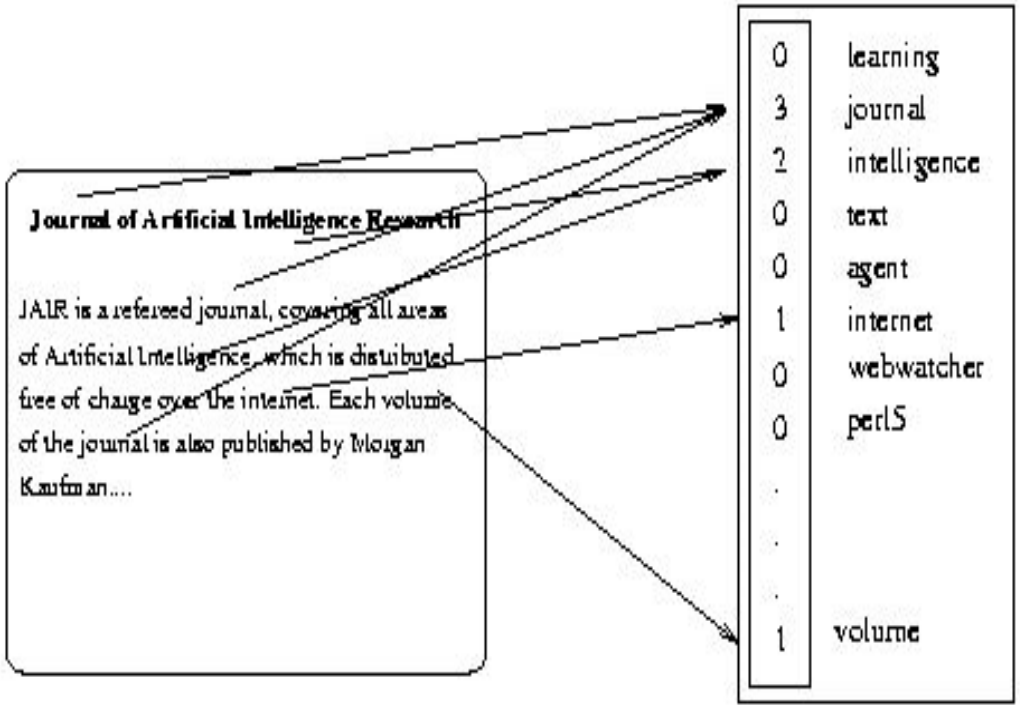
- We learn how can we
  - represent text in a simple numerical form in the computer
  - find out topics from a collection of text documents

Salton's Vector Space Model

Gerald Salton

'60 – '70

- Represent each document by a high-dimensional vector in the space of words



- Represent the doc as a vector where each entry corresponds to a different word and the number at that entry corresponds to how many times that word was present in the document (or some function of it)
  - Number of words is huge
  - Select and use a smaller set of words that are of interest
  - E.g. uninteresting words: ‘and’, ‘the’ ‘at’, ‘is’, etc. These are called stop-words
  - Stemming: remove endings. E.g. ‘learn’, ‘learning’, ‘learnable’, ‘learned’ could be substituted by the single stem ‘learn’
  - Other simplifications can also be invented and used
  - The set of different remaining words is called dictionary or vocabulary. Fix an ordering of the terms in the dictionary so that you can operate them by their index.

Example

This is a small document collection that consists of 9 text documents. Terms that are in our dictionary are in bold.

c1: *Human* machine *interface* for Lab ABC *computer* applications
c2: A *survey* of *user* opinion of *computer system response time*
c3: The *EPS user interface* management *system*
c4: *System* and *human system* engineering testing of *EPS*
c5: Relation of *user*-perceived *response time* to error measurement

m1: The generation of random, binary, unordered *trees*
m2: The intersection *graph* of paths in *trees*
m3: *Graph minors* IV: Widths of *trees* and well-quasi-ordering
m4: *Graph minors*: A *survey*

Collect all doc vectors into a term by document matrix

human	1	0	0	1	0	0	0	0	0
interface	1	0	1	0	0	0	0	0	0
computer	1	1	0	0	0	0	0	0	0
user	0	1	1	0	1	0	0	0	0
system	0	1	1	2	0	0	0	0	0
response	0	1	0	0	1	0	0	0	0
time	0	1	0	0	1	0	0	0	0
EPS	0	0	1	1	0	0	0	0	0
survey	0	1	0	0	0	0	0	0	1
trees	0	0	0	0	0	1	1	1	0
graph	0	0	0	0	0	0	1	1	1
minors	0	0	0	0	0	0	0	1	1

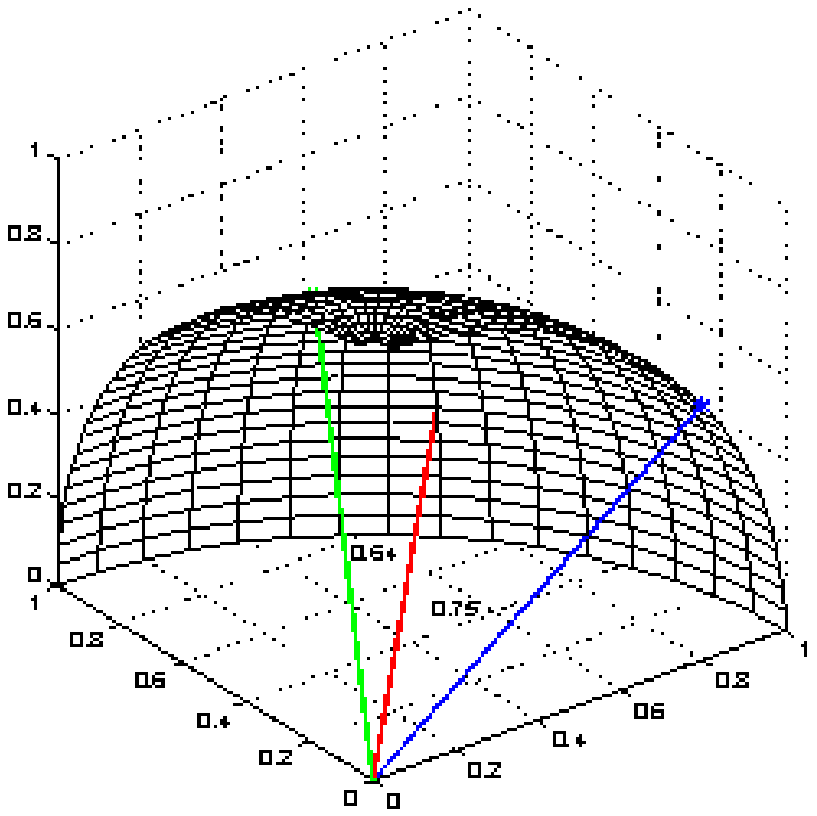
Queries

- Have a collection of documents
- Want to find the most relevant documents to a query
- A query is just like a very short document
- Compute the similarity between the query and all documents in the collection
- Return the best matching documents

- When are two document similar?
- When are two document vectors similar?

Document similarity

cosine of angle between query and document(s)



$$\cos(\mathbf{x},\mathbf{y}) = \frac{\mathbf{x}^T\mathbf{y}}{\|\mathbf{x}\| \cdot \|\mathbf{y}\|}$$

Simple, intuitive

Fast to compute, because x and y are typically sparse (i.e. have many 0-s)

How to measure success?

- Assume there is a set of ‘correct answers’ to the query. The docs in this set are called relevant to the query
- The set of documents returned by the system are called retrieved documents
- *Precision*: what percentage of the retrieved documents are relevant
- *Recall*: what percentage of all relevant documents are retrieved

Problems

- Synonyms: separate words that have the same meaning.
  - E.g. ‘car’ & ‘automobile’
  - They tend to reduce recall
- Polysems: words with multiple meanings
  - E.g. ‘saturn’
  - They tend to reduce precision

→ The problem is more general: there is a disconnect between *topics* and *words*

LSA using SVD

Latent Semantic Analysis (LSA)

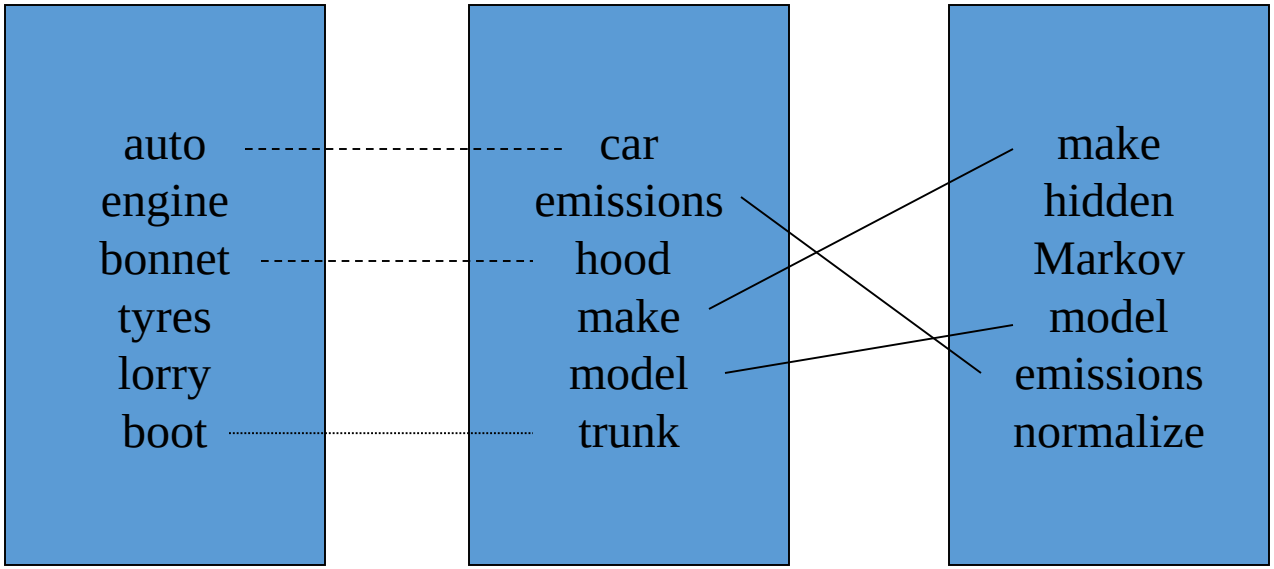
- LSA aims to discover something about the meaning behind the words; about the topics in the documents.
- What is the difference between topics and words?
  - Words are observable
  - Topics are not. They are latent.
- How to find out topics from the words in an automatic way?
  - We can imagine them as a compression of words
  - A combination of words
  - Try to formalise this

The Problem

- Two problems that arise using the vector space model:
  - synonymy: many ways to refer to the same object, e.g. car and automobile
    - leads to poor recall
  - polysemy: most words have more than one distinct meaning, e.g. model, python, chip
    - leads to poor precision

The Problem

- Example: Vector Space Model
  - (from Lillian Lee)



Synonymy

Will have small cosine

but are related

Polysemy

Will have large cosine

but not truly related

The Setting

- Corpus, a set of N documents
  - D={d_1, … ,d_N}
- Vocabulary, a set of M words
  - W={w_1, … ,w_M}
- A matrix of size N * M to represent the occurrence of words in documents
  - Called the term-document matrix

Latent Semantic Indexing

- Latent – “present but not evident, hidden”
- Semantic – “meaning”

LSI finds the “hidden meaning” of terms based on their occurrences in documents

Latent Semantic Space

- LSI maps terms and documents to a “latent semantic space”
- Comparing terms in this space should make synonymous terms look more similar

LSI Method

- Singular Value Decomposition (SVD)

  - $A(n*m) = U(n*n)\ E(n*m)\ V(m*m)$
    - U's colums are Eigenvectors of AA^T
    - V's columns are Eigenvectors of A^TA
    - E is a diagonal matrix containing the square roots of the eigenvalues of U or V in descending order
  - Keep only k eigen values from E
    - $A(n*m) \approx U(n*k)\ E(k*k)\ V(k*m)$
  - Convert terms and documents to points in k-dimensional space

A Small Example

Technical Memo Titles

c1: *Human* machine *interface* for ABC *computer* applications

c2: A *survey* of *user* opinion of *computer system response time*

c3: The *EPS user interface* management *system*

c4: *System* and *human system* engineering testing of *EPS*

c5: Relation of *user* perceived *response time* to error measurement

m1: The generation of random, binary, ordered *trees*

m2: The intersection *graph* of paths in *trees*

m3: *Graph minors* IV: Widths of *trees* and well-quasi-ordering

m4: *Graph minors*: A *survey*

A Small Example – 2

	c1	c2	c3	c4	c5	m1	m2	m3	m4
human	1	0	0	1	0	0	0	0	0
interface	1	0	1	0	0	0	0	0	0
computer	1	1	0	0	0	0	0	0	0
user	0	1	1	0	1	0	0	0	0
system	0	1	1	2	0	0	0	0	0
response	0	1	0	0	1	0	0	0	0
time	0	1	0	0	1	0	0	0	0
EPS	0	0	1	1	0	0	0	0	0
survey	0	1	0	0	0	0	0	0	1
trees	0	0	0	0	0	1	1	1	0
graph	0	0	0	0	0	0	1	1	1
minors	0	0	0	0	0	0	0	1	1

r (human.user) = -.38 r (human.minors) = -.29

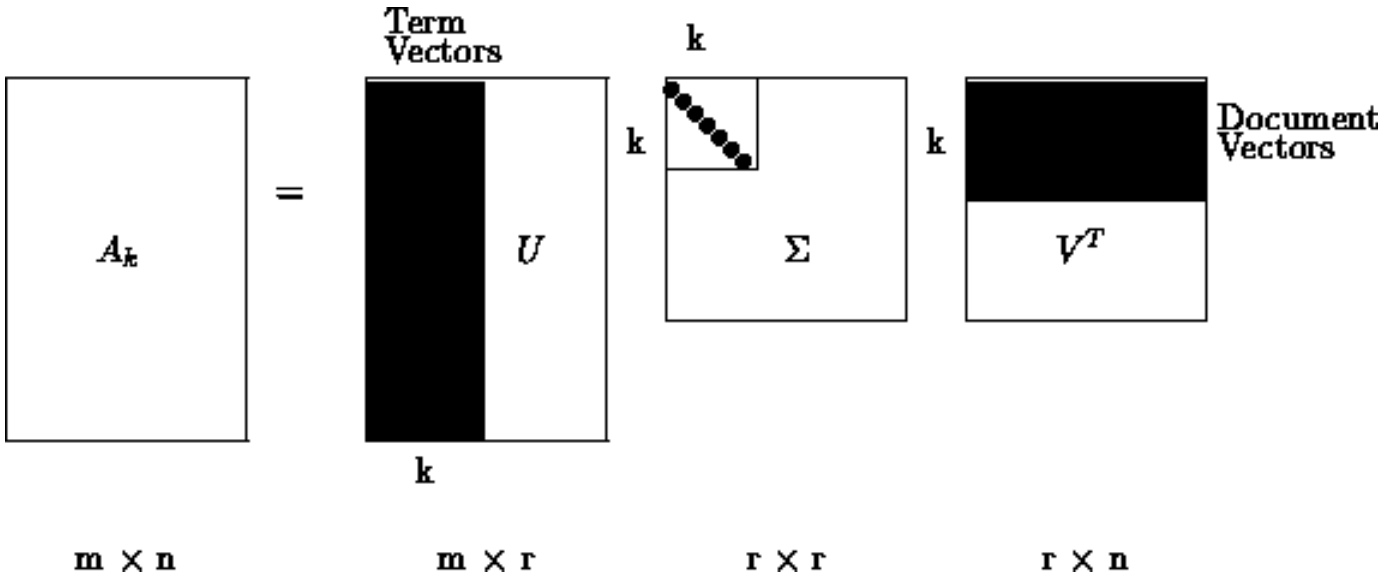
A Small Example – 3

- Singular Value Decomposition

$$\{A\}=\{U\}\{S\}\{V\}^T$$

- Dimension Reduction

$$\{\tilde{A}\}\sim=\{\tilde{U}\}\{\tilde{S}\}\{\tilde{V}\}^T$$



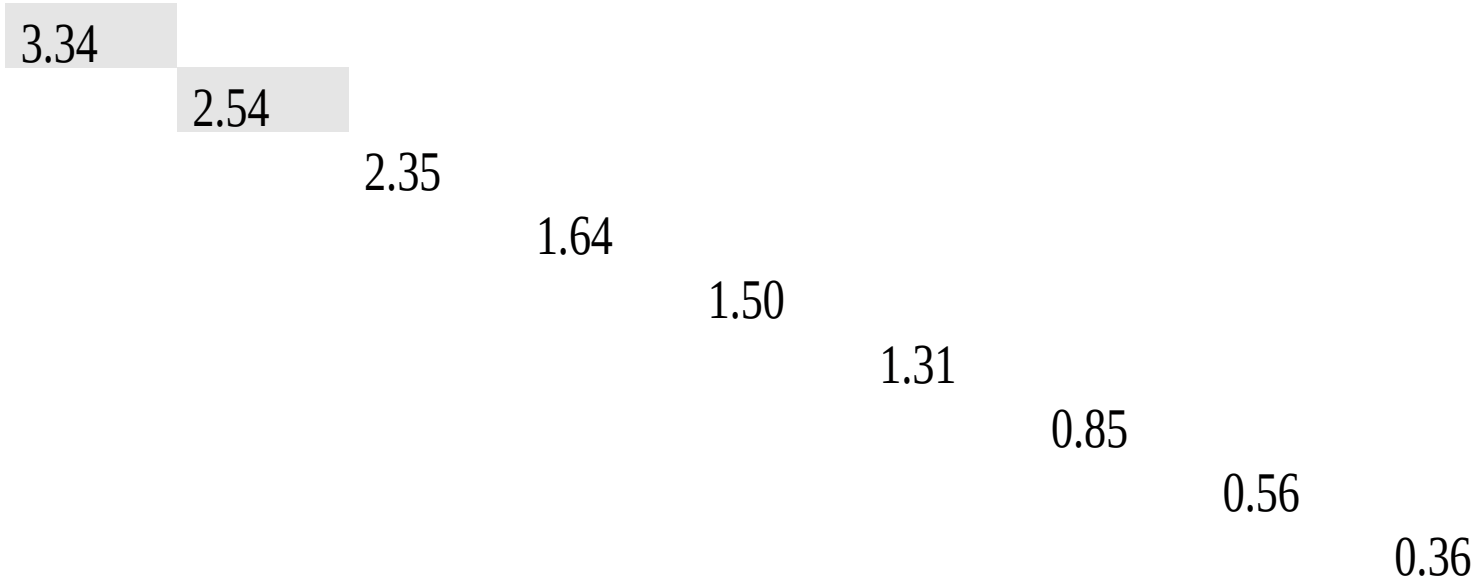
A Small Example – 4

- {U} =

0.22	-0.11	0.29	-0.41	-0.11	-0.34	0.52	-0.06	-0.41
0.20	-0.07	0.14	-0.55	0.28	0.50	-0.07	-0.01	-0.11
0.24	0.04	-0.16	-0.59	-0.11	-0.25	-0.30	0.06	0.49
0.40	0.06	-0.34	0.10	0.33	0.38	0.00	0.00	0.01
0.64	-0.17	0.36	0.33	-0.16	-0.21	-0.17	0.03	0.27
0.27	0.11	-0.43	0.07	0.08	-0.17	0.28	-0.02	-0.05
0.27	0.11	-0.43	0.07	0.08	-0.17	0.28	-0.02	-0.05
0.30	-0.14	0.33	0.19	0.11	0.27	0.03	-0.02	-0.17
0.21	0.27	-0.18	-0.03	-0.54	0.08	-0.47	-0.04	-0.58
0.01	0.49	0.23	0.03	0.59	-0.39	-0.29	0.25	-0.23
0.04	0.62	0.22	0.00	-0.07	0.11	0.16	-0.68	0.23
0.03	0.45	0.14	-0.01	-0.30	0.28	0.34	0.68	0.18

A Small Example – 5

- {S} =



A Small Example – 6

- {V} =

0.20	0.61	0.46	0.54	0.28	0.00	0.01	0.02	0.08
-0.06	0.17	-0.13	-0.23	0.11	0.19	0.44	0.62	0.53
0.11	-0.50	0.21	0.57	-0.51	0.10	0.19	0.25	0.08
-0.95	-0.03	0.04	0.27	0.15	0.02	0.02	0.01	-0.03
0.05	-0.21	0.38	-0.21	0.33	0.39	0.35	0.15	-0.60
-0.08	-0.26	0.72	-0.37	0.03	-0.30	-0.21	0.00	0.36
0.18	-0.43	-0.24	0.26	0.67	-0.34	-0.15	0.25	0.04
-0.01	0.05	0.01	-0.02	-0.06	0.45	-0.76	0.45	-0.07
-0.06	0.24	0.02	-0.08	-0.26	-0.62	0.02	0.52	-0.45

Pros and Cons

- LSI puts documents together even if they don't have common words if
  - The docs share frequently co-occurring terms

- Disadvantages:
  - Statistical foundation is missing

Summary

- Some Issues
  - SVD Algorithm complexity O(n^2k^3)
    - n = number of terms
    - k = number of dimensions in semantic space (typically small ~50 to 350)
    - for stable document collection, only have to run once
    - dynamic document collections: might need to rerun SVD

Summary

- Some issues
  - Finding optimal dimension for semantic space
    - precision-recall improve as dimension is increased until hits optimal, then slowly decreases until it hits standard vector model
    - run SVD once with big dimension, say k = 1000
      - then can test dimensions <= k
    - in many tasks 150-350 works well, still room for research

Summary

- Some issues
  - SVD assumes normally distributed data
    - term occurrence is not normally distributed
    - matrix entries are weights, not counts, which may be normally distributed even when counts are not

Summary

- Has proved to be a valuable tool in many areas of NLP as well as IR
  - summarization
  - cross-language IR
  - topics segmentation
  - text classification
  - question answering
  - more

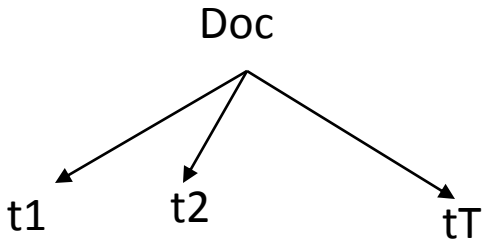
pLSI

Probabilistic Latent Semantic Analysis

- Let us start from what we know
- Remember the random sequence model

$$P(doc) = P(term_1 \mid doc)P(term_2 \mid doc)...P(term_L \mid doc)$$

$$= \prod_{l=1}^{L} P(term_l \mid doc) = \prod_{t=1}^{T} P(term_t \mid doc)^{X(term_t, doc)}$$

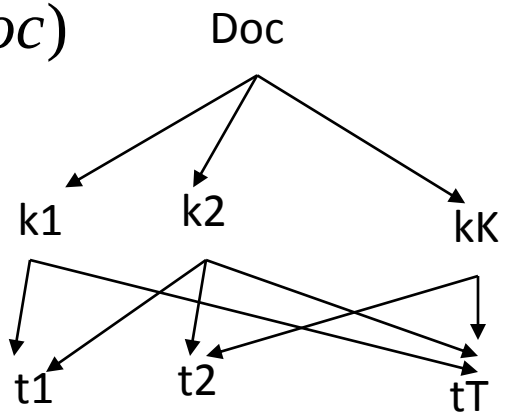


Probabilistic Latent Semantic Analysis

Think: Topic ~ Factor

- Now let us have K topics as well:

$$P(term_t \mid doc) = \sum_{k=1}^{K} P(term_t \mid topic_k) P(topic_k \mid doc)$$



The same, written using shorthands :

$$P(t \mid doc) = \sum_{k=1}^{K} P(t \mid k) P(k \mid doc)$$

So by replacing this, for any doc in the collection ,

$$P(doc) = \prod_{t=1}^{T} \{ \sum_{k=1}^{K} P(t \mid k) P(k \mid doc) \}^{X(t,doc)}$$

Which are the parameters of this model?

Probabilistic Latent Semantic Analysis

- The parameters of this model are:
  P(t|k)
  P(k|doc)
- It is possible to derive the equations for computing these parameters by Maximum Likelihood.
- If we do so, what do we get?
  P(t|k) for all t and k, is a term by topic matrix
  (gives which terms make up a topic)
  P(k|doc) for all k and doc, is a topic by document matrix
  (gives which topics are in a document)

"Arts"	"Budgets"	"Children"	"Education"
NEW	MILLION	CHILDREN	SCHOOL
FILM	TAX	WOMEN	STUDENTS
SHOW	PROGRAM	PEOPLE	SCHOOLS
MUSIC	BUDGET	CHILD	EDUCATION
MOVIE	BILLION	YEARS	TEACHERS
PLAY	FEDERAL	FAMILIES	HIGH
MUSICAL	YEAR	WORK	PUBLIC
BEST	SPENDING	PARENTS	TEACHER
ACTOR	NEW	SAYS	BENNETT
FIRST	STATE	FAMILY	MANIGAT
YORK	PLAN	WELFARE	NAMPHY
OPERA	MONEY	MEN	STATE
THEATER	PROGRAMS	PERCENT	PRESIDENT
ACTRESS	GOVERNMENT	CARE	ELEMENTARY
LOVE	CONGRESS	LIFE	HAITI

The William Randolph Hearst Foundation will give $1.25 million to Lincoln Center, Metropolitan Opera Co., New York Philharmonic and Juilliard School. "Our board felt that we had a real opportunity to make a mark on the future of the performing arts with these grants an act every bit as important as our traditional areas of support in health, medical research, education and the social services," Hearst Foundation President Randolph A. Hearst said Monday in announcing the grants. Lincoln Center's share will be $200,000 for its new building, which will house young artists and provide new public facilities. The Metropolitan Opera Co. and New York Philharmonic will receive $400,000 each. The Juilliard School, where music and the performing arts are taught, will get $250,000. The Hearst Foundation, a leading supporter of the Lincoln Center Consolidated Corporate Fund, will make its usual annual $100,000 donation, too.

Deriving the parameter estimation algorithm

- The log likelihood of this model is the log probability of the entire collection:

$$\sum_{d=1}^{N} \log P(d) = \sum_{d=1}^{N} \sum_{t=1}^{T} X(t,d) \log \sum_{k=1}^{K} P(t \mid k) P(k \mid d)$$

which is to be maximised w.r.t. parameters $P(t \mid k)$ and then also $P(k \mid d)$,

subject to the constraint s that $\sum_{t=1}^{T} P(t \mid k) = 1$ and $\sum_{k=1}^{K} P(k \mid d) = 1.$

For those who would enjoy to work it out:

- Lagrangian terms are added to ensure the constraints
- Derivatives are taken wrt the parameters (one of them at a time) and equate these to zero
- Solve the resulting equations. You will get fixed point equations which can be solved iteratively. This is the PLSA algorithm.

Note these steps are the same in deriving the Maximum Likelihood estimate for random sequence models, just the working is a little more tedious.

We skip doing this in the class, we just give the resulting algorithm (see next slide)

The PLSA algorithm

- Inputs: term by document matrix X(t,d), t=1:T, d=1:N and the number K of topics sought
- Initialise arrays P1 and P2 randomly with numbers between [0,1] and normalise them to sum to 1 along rows
- Iterate until convergence

  For d=1 to N, For t =1 to T, For k=1:K

$$P1(t,k) \leftarrow P1(t,k)\sum_{d=1}^{N}\frac{X(t,d)}{\sum_{k=1}^{K}P1(t,k)P2(k,d)}P2(k,d); \quad P1(t,k) \leftarrow \frac{P1(t,k)}{\sum_{t=1}^{T}P1(t,k)}$$

$$P2(k,d) \leftarrow P2(k,d)\sum_{t=1}^{T}\frac{x(t,d)}{\sum_{k=1}^{K}P1(t,k)P2(k,d)}P1(t,k); \quad P2(k,d) \leftarrow \frac{P2(k,d)}{\sum_{k=1}^{K}P2(k,d)}$$

- Output: arrays P1 and P2, which hold the estimated parameters P(t|k) and P(k|d) respectively

Example of topics found from a Science Magazine papers collection

universe	0.0439
galaxies	0.0375
clusters	0.0279
matter	0.0233
galaxy	0.0232
cluster	0.0214
cosmic	0.0137
dark	0.0131
light	0.0109
density	0.01

drug	0.0672
patients	0.0493
drugs	0.0444
clinical	0.0346
treatment	0.028
trials	0.0277
therapy	0.0213
trial	0.0164
disease	0.0157
medical	0.00997

cells	0.0675
stem	0.0478
human	0.0421
cell	0.0309
gene	0.025
tissue	0.0185
cloning	0.0169
transfer	0.0155
blood	0.0113
embryos	0.0111

sequence	0.0818
sequences	0.0493
genome	0.033
dna	0.0257
sequencing	0.0172
map	0.0123
genes	0.0122
chromosome	0.0119
regions	0.0119
human	0.0111

years	0.156
million	0.0556
ago	0.045
time	0.0317
age	0.0243
year	0.024
record	0.0238
early	0.0233
billion	0.0177
history	0.0148

bacteria	0.0983
bacterial	0.0561
resistance	0.0431
coli	0.0381
strains	0.025
microbiol	0.0214
microbial	0.0196
strain	0.0165
salmonella	0.0163
resistant	0.0145

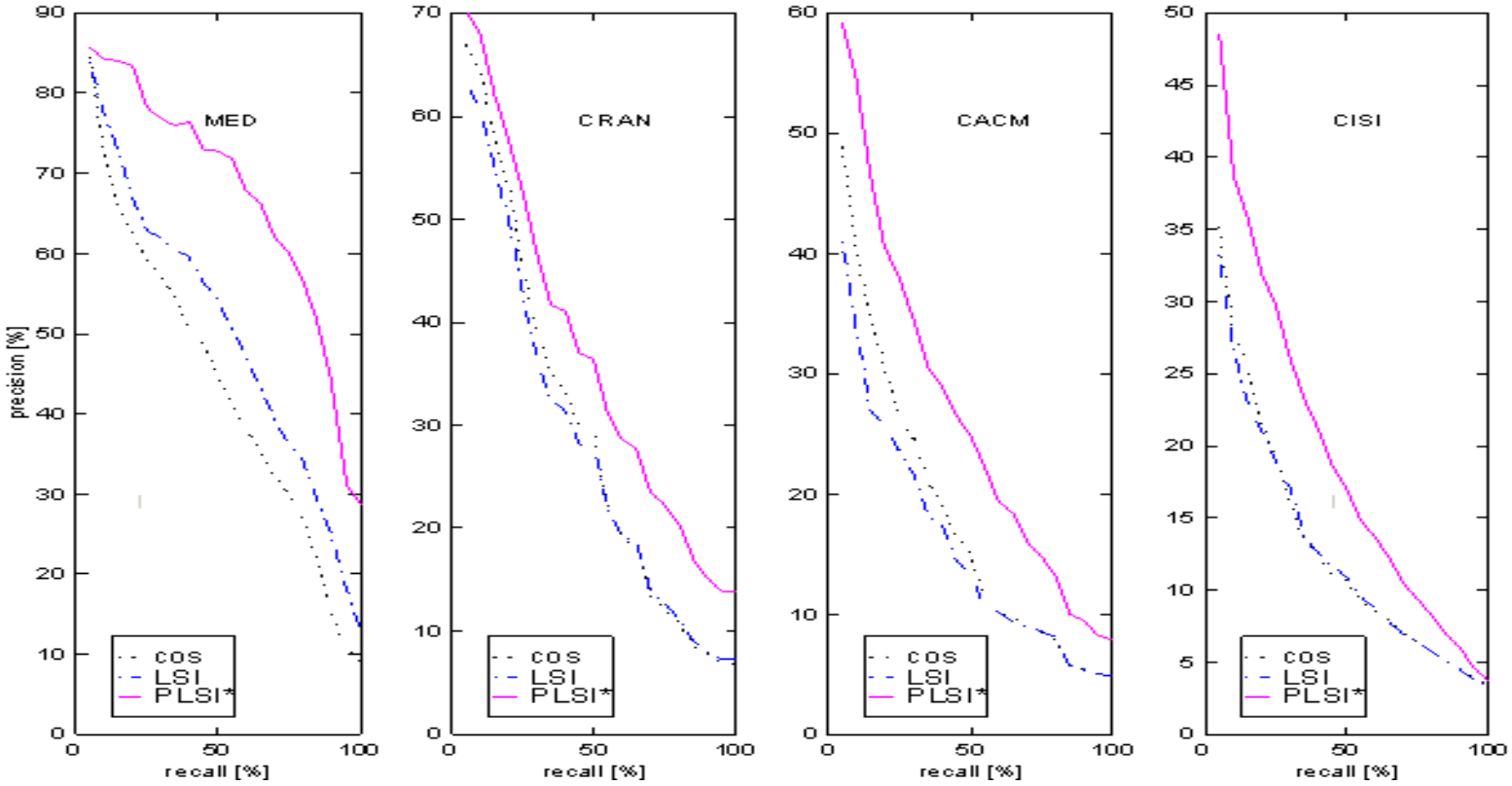
male	0.0558
females	0.0541
female	0.0529
males	0.0477
sex	0.0339
reproductive	0.0172
offspring	0.0168
sexual	0.0166
reproduction	0.0143
eggs	0.0138

theory	0.0811
physics	0.0782
physicists	0.0146
einstein	0.0142
university	0.013
gravity	0.013
black	0.0127
theories	0.01
aps	0.00987
matter	0.00954

immune	0.0909
response	0.0375
system	0.0358
responses	0.0322
antigen	0.0263
antigens	0.0184
immunity	0.0176
immunology	0.0145
antibody	0.014
autoimmune	0.0128

stars	0.0524
star	0.0458
astrophys	0.0237
mass	0.021
disk	0.0173
black	0.0161
gas	0.0149
stellar	0.0127
astron	0.0125
hole	0.00824

The performance of a retrieval system based on this model (PLSI) was found superior to that of both the vector space based similarity (cos) and a non-probabilistic latent semantic indexing (LSI) method. (We skip details here.)



From Th. Hofmann, 2000

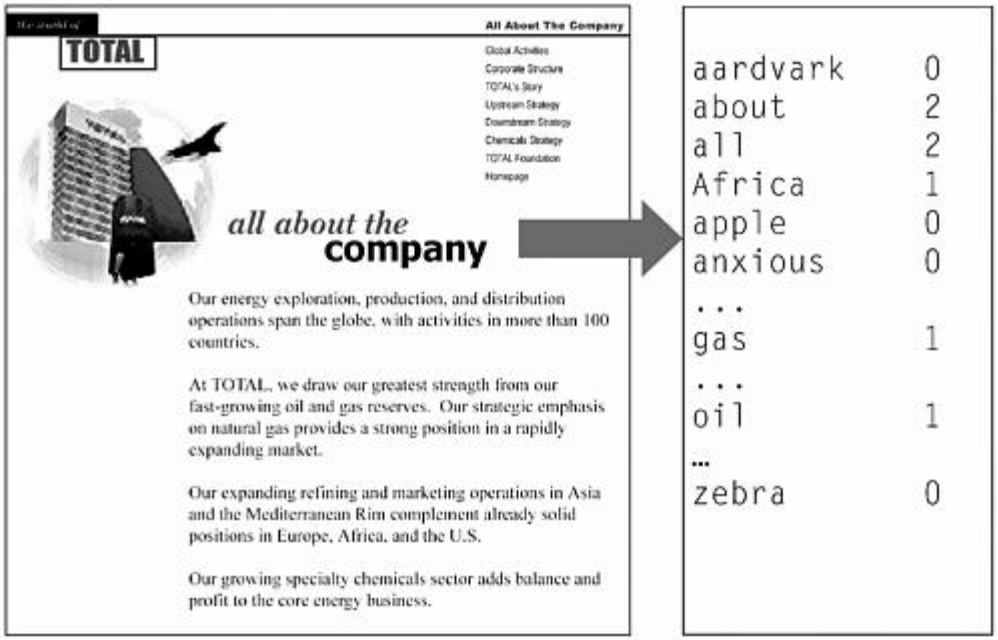
Summing up

- Documents can be represented as numeric vectors in the space of words.
- The order of words is lost but the co-occurrences of words may still provide useful insights about the topical content of a collection of documents.
- PLSA is an unsupervised method based on this idea.
- We can use it to find out what topics are there in a collection of documents
- It is also a good basis for information retrieval systems

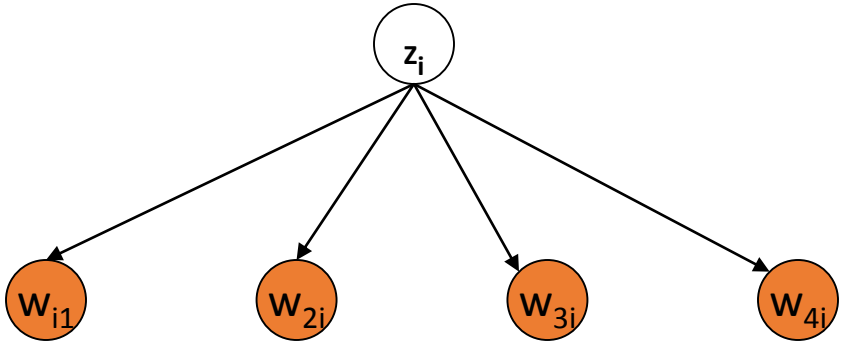
LDA

“Bag of Words” Models

- Let’s assume that all the words within a document are exchangeable.



Mixture of Unigrams



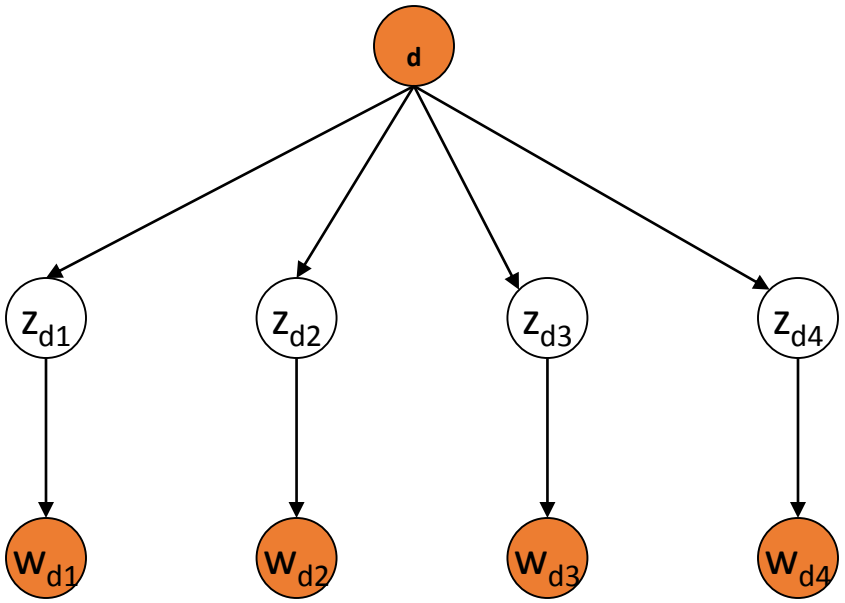
Mixture of Unigrams Model (this is just Naïve Bayes)

For each of M documents,

- Choose a topic z.
- Choose N words by drawing each one independently from a multinomial conditioned on z.

In the Mixture of Unigrams model, we can only have one topic per document!

The pLSI Model



Probabilistic Latent Semantic Indexing (pLSI) Model

For each word of document d in the training set,

- Choose a topic z according to a multinomial conditioned on the index d.
- Generate the word by drawing from a multinomial conditioned on z.

In pLSI, documents can have multiple topics.

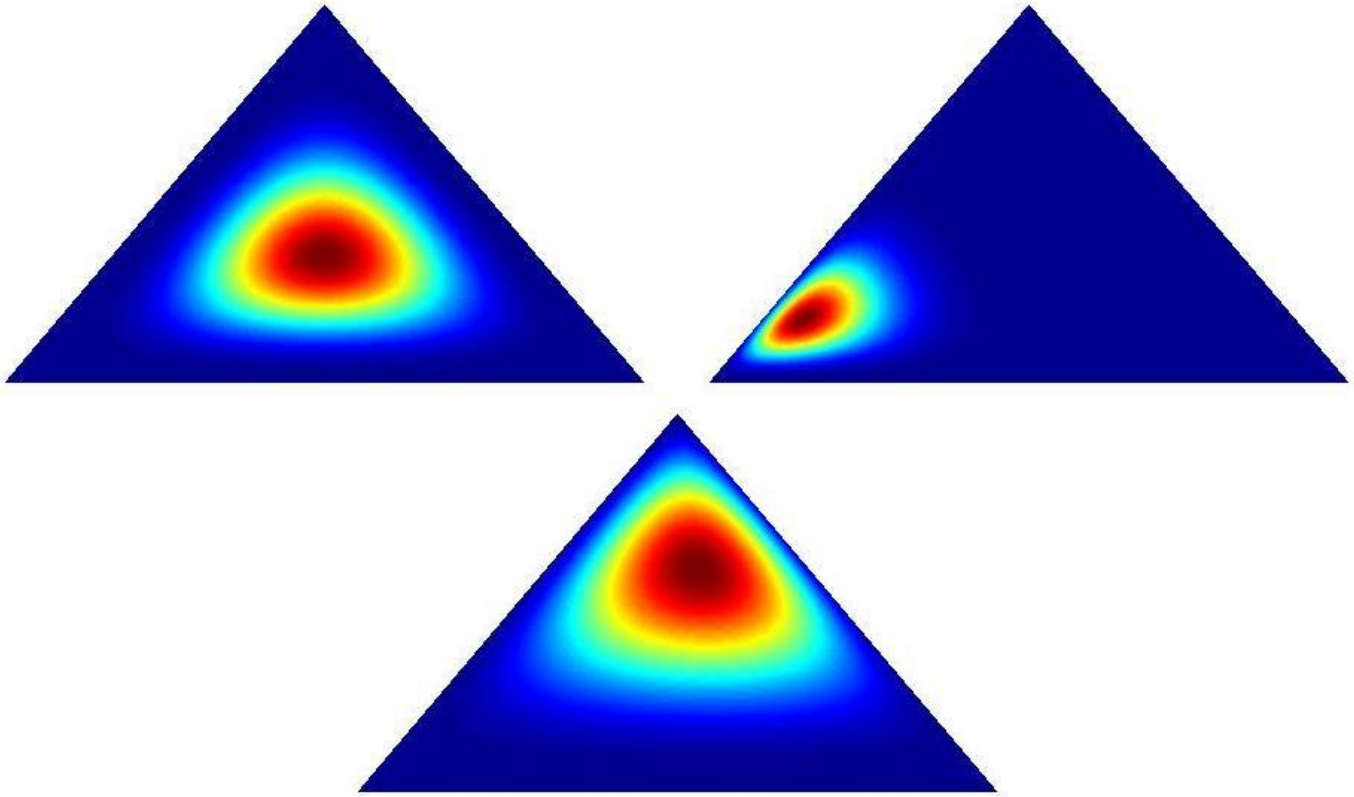
Motivations for LDA

- In pLSI, the observed variable d is an index into some training set. There is no natural way for the model to handle previously unseen documents.
- The number of parameters for pLSI grows linearly with M (the number of documents in the training set).
- We would like to be Bayesian about our topic mixture proportions.

Dirichlet Distributions

- In the LDA model, we would like to say that the *topic mixture proportions* for each document are drawn from some distribution.
- So, we want to put a distribution on multinomials. That is, k-tuples of non-negative numbers that sum to one.
- The space is of all of these multinomials has a nice geometric interpretation as a (k-1)-*simplex*, which is just a generalization of a triangle to (k-1) dimensions.
- Criteria for selecting our prior:
  - It needs to be defined for a (k-1)-simplex.
  - Algebraically speaking, we would like it to play nice with the multinomial distribution.

Dirichlet Examples



Dirichlet Distributions

$$p(\theta|\alpha) = \frac{\Gamma(\sum_{i=1}^{k} \alpha_i)}{\prod_{i=1}^{k} \Gamma(\alpha_i)} \prod_{i=1}^{k} \theta_i^{\alpha_i - 1}$$

- Useful Facts:
  - This distribution is defined over a (k-1)-simplex. That is, it takes k non-negative arguments which sum to one. Consequently it is a natural distribution to use over multinomial distributions.
  - In fact, the Dirichlet distribution is the conjugate prior to the multinomial distribution. (This means that if our likelihood is multinomial with a Dirichlet prior, then the posterior is also Dirichlet!)
  - The Dirichlet parameter α_i can be thought of as a prior count of the i^{th} class.

Gamma Function

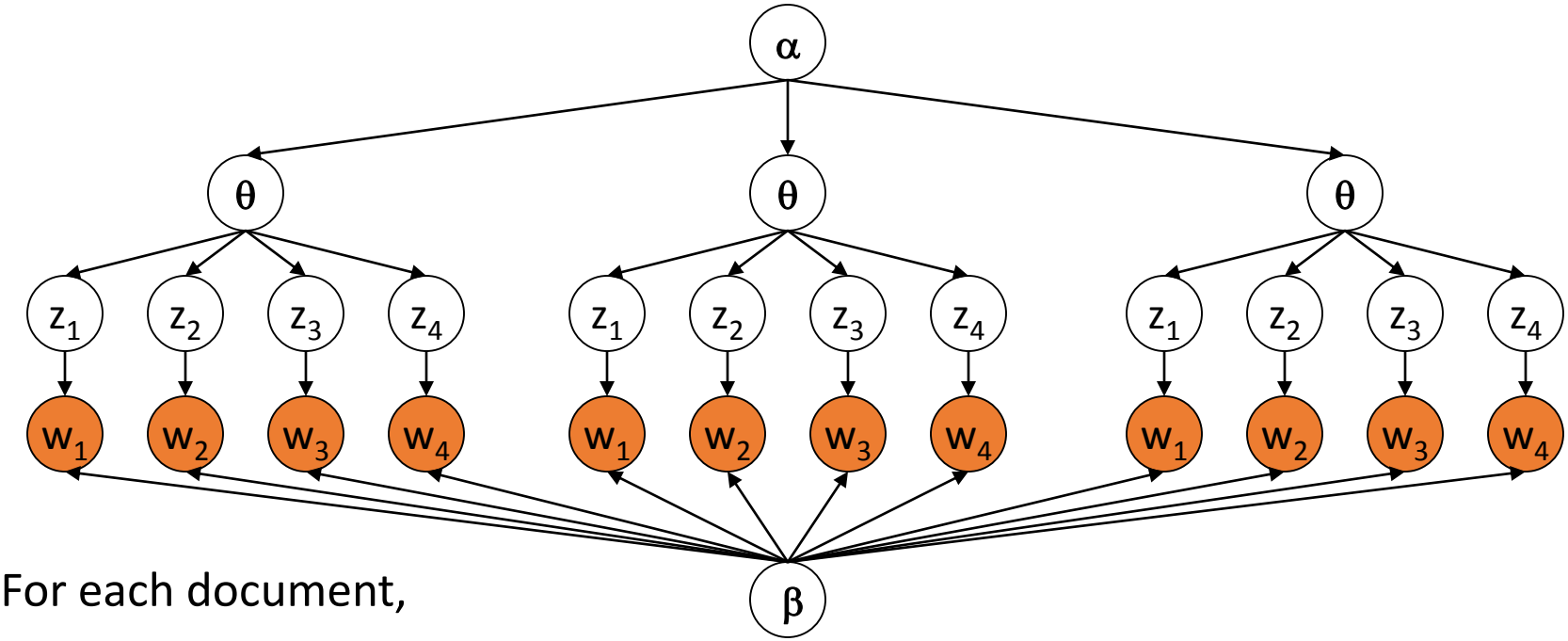
- Intuition: a version of factorial for real numbers

$$\Gamma(x) = \int_0^\infty e^{-t} t^{x-1} dt$$

- Has the following property:

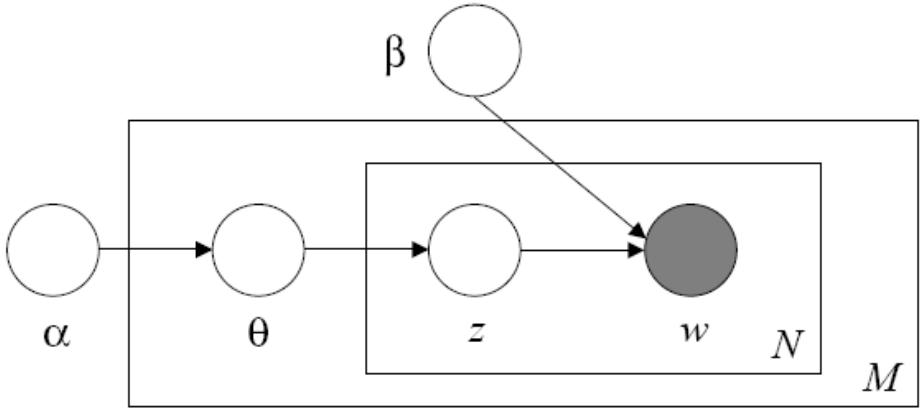
$$\Gamma(1 + x) = x\,\Gamma(x)$$

The LDA Model



- For each document,
- Choose $\theta \sim \text{Dirichlet}(\alpha)$
- For each of the N words wn:
  - Choose a topic $z_n \text{»} \text{ Multinomial}(\theta)$
  - Choose a word w_n from $p(w_n | z_n, \beta)$, a multinomial probability conditioned on the topic z_n.

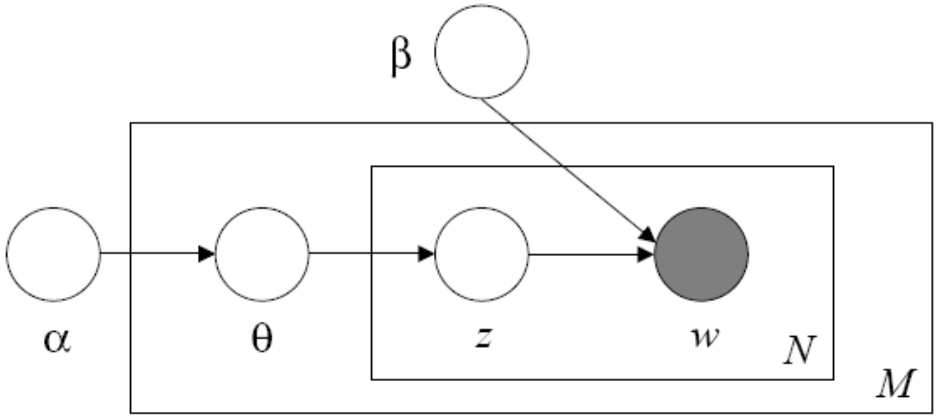
The LDA Model



For each document,

- Choose θ» Dirichlet(α)
- For each of the N words w_n:
  - Choose a topic z_n» Multinomial(θ)
  - Choose a word w_n from $p(w_n|z_n,\beta)$, a multinomial probability conditioned on the topic z_n.

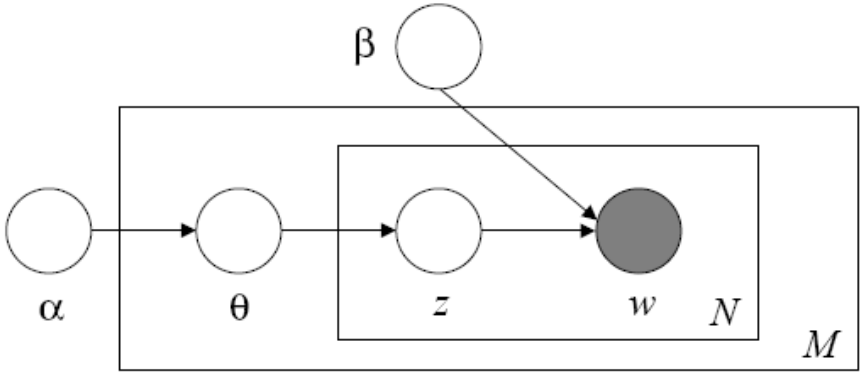
Inference

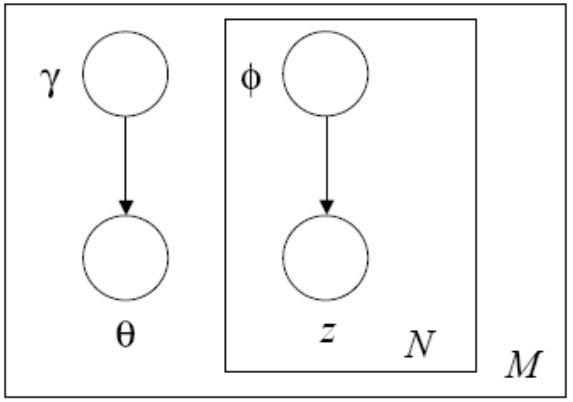


•The inference problem in LDA is to compute the posterior of the hidden variables given a document and corpus parameters α and β. That is, compute $p(\theta,z|w,\alpha,\beta)$.

•Unfortunately, exact inference is intractable, so we turn to alternatives…

Variational Inference





•In variational inference, we consider a simplified graphical model with variational parameters γ, ϕ and minimize the KL Divergence between the variational and posterior distributions.

$$(\gamma^*, \phi^*) = \arg\min_{(\gamma,\phi)} KL(q(\theta, z|\gamma, \phi)||p(\theta, z|w, \alpha, \beta))$$

Parameter Estimation

- Given a corpus of documents, we would like to find the parameters α and β which maximize the likelihood of the observed data.
- Strategy (*Variational EM)*:
  - Lower bound log p(w|α,β) by a function L(γ,ϕ;α,β)
  - Repeat until convergence:
    - Maximize L(γ,ϕ;α,β) with respect to the variational parameters γ,ϕ.
    - Maximize the bound with respect to parameters α and β.

Questions