Outline

13.1 IR Effectiveness Measures

13.2 Probabilistic IR

13.3 Statistical Language Model

13.4 Latent-Topic Models

- 13.4.1 LSI based on SVD
- 13.4.2 pLSI and LDA
- 13.4.3 Skip-Gram Model

13.5 Learning to Rank

Not only does God play dice, but He sometimes confuses us by throwing them where they can't be seen.

-- Stephen Hawking

13.4 Latent Topic Models

- Ranking models like tf*idf, Prob. IR and Statistical LMs do not capture lexical relations between terms in natural language: **synonymy** (e.g. *car* and *automobile*), **homonymy** (e.g. *java*), hyponymy (e.g. *SUV* and *car*), meronymy (e.g. *wheel* and *car*), etc.

- **Word co-occurrence** and **indirect co-occurrence** can help:
  car and *automobile* both occur with *fuel*, *emission*, *garage*, …
  java occurs with *class* and *method* but also with *grind* and *coffee*

- **Latent topic models** assume that documents are composed from a number *k* of **latent (hidden) topics**
  where k ≪ |V| with vocabulary V
  → project docs consisting of terms into
  lower-dimensional space of docs consisting of latent topics

13.4.1 Flashback: SVD

Theorem:

Each real-valued $m \times n$ matrix A with rank r can be decomposed into the form $\mathbf{A = U \times \Delta \times V^T}$ with

an **$m \times r$ matrix U** with orthonormal column vectors,

an **$r \times r$ diagonal matrix Δ**, and

an **$n \times r$ matrix V** with orthonormal column vectors.

This decomposition is called **singular value decomposition (SVD)** and is unique when the elements of Δ or sorted.

Theorem:

In the singular value decomposition $A = U \times \Delta \times V^T$ of matrix A the matrices U, Δ, and V can be derived as follows:

- Δ consists of the singular values of A, i.e. the positive roots of the Eigenvalues of $A^T \times A$,
- the columns of U are the Eigenvectors of $A \times A^T$,
- the columns of V are the Eigenvectors of $A^T \times A$.

SVD as Low-Rank Approximation (Regression)

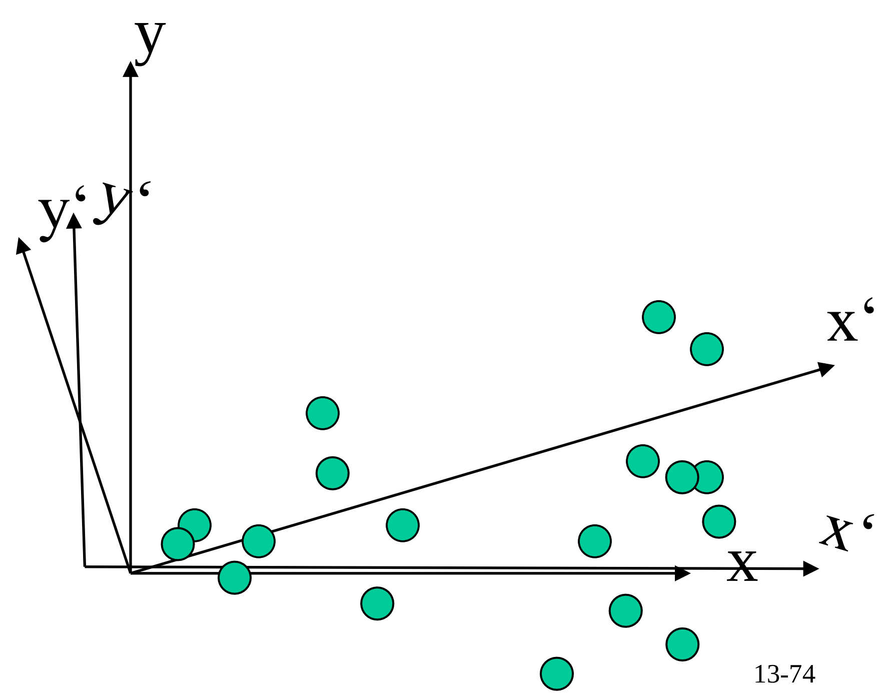
Theorem:

Let A be an m×n matrix with rank r, and let $\mathbf{A_k = U_k \times \Delta_k \times V_k^T}$,
where the k×k diagonal matrix Δ_k contains the **k largest singular values** of A and the m×k matrix U_k and the n×k matrix V_k contain the corresponding Eigenvectors from the SVD of A.

Among all m×n matrices C with rank at most k
A_k is the matrix that minimizes the Frobenius norm

$$\|A-C\|_F^2 = \sum_{i=1}^{m}\sum_{j=1}^{n}(A_{ij}-C_{ij})^2$$

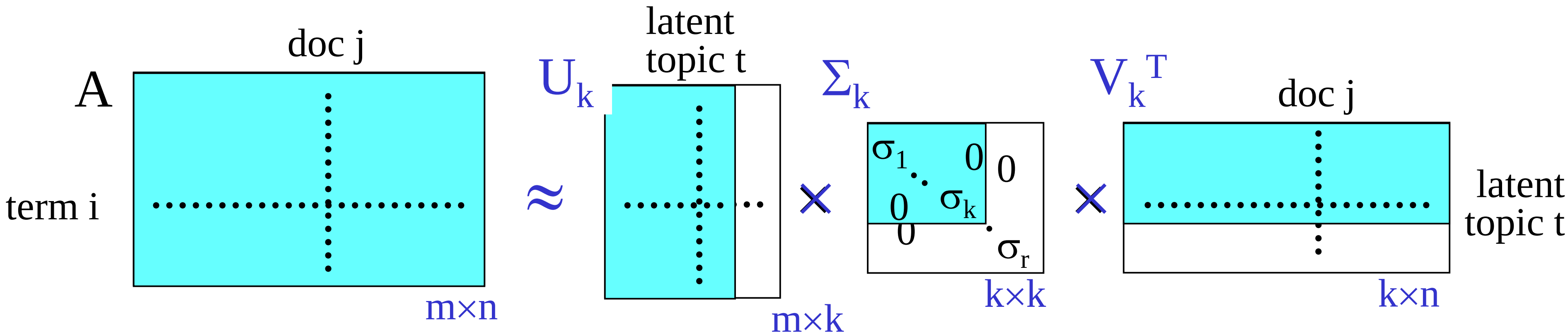
Example:
m=2, n=8, k=1
projection onto x‘ axis
minimizes „error“ or
maximizes „variance“
in k-dimensional space



Latent Semantic Indexing (LSI): Applying SVD to Vector Space Model

A is the m×n term-document similarity matrix. Then:

- U and U_k are the m×r and m×k term-topic similarity matrices,
- V and V_k are the n×r and n×k document-topic similarity matrices,
- $A \times A^T$ and $A_k \times A_k^T$ are the m×m term-term similarity matrices,
- $A^T \times A$ and $A_k^T \times A_k$ are the n×n document-document similarity matrices



mapping of m×1 vectors into latent-topic space:
$$d_j \mapsto U_k^T \times d_j =: d_j'$$
$$q \mapsto U_k^T \times q =: q'$$

scalar-product similarity in latent-topic space: $d_j'^T \times q' = ((\Delta_k V_k^T)_{*j})^T \times q'$

Indexing and Query Processing

- The matrix $\Delta_k V_k^T$ corresponds to a „topic index" and is stored in a suitable data structure.
  Instead of $\Delta_k V_k^T$ the simpler index V_k^T could be used.
- Additionally the term-topic mapping U_k must be stored.
- A query q (an m×1 column vector) in the term vector space is transformed into query $q' = U_k^T \times q$ (a k×1 column vector) and evaluated in the topic vector space (i.e. V_k) (e.g. by scalar-product similarity $V_k^T \times q'$ or cosine similarity)
- A new document d (an m×1 column vector) is transformed into $d' = U_k^T \times d$ (a k ×1 column vector) and appended to the „index" V_k^T as an additional column („folding-in")

Example 1 for Latent Semantic Indexing

m=5 (interface, library, Java, Kona, blend), n=7

$$A = \begin{pmatrix} 1 & 2 & 1 & 5 & 0 & 0 & 0 \\ 1 & 2 & 1 & 5 & 0 & 0 & 0 \\ 1 & 2 & 1 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 3 & 1 \\ 0 & 0 & 0 & 0 & 2 & 3 & 1 \end{pmatrix} = \underbrace{\begin{pmatrix} 0.58 & 0.00 \\ 0.58 & 0.00 \\ 0.58 & 0.00 \\ 0.00 & 0.71 \\ 0.00 & 0.71 \end{pmatrix}}_{U} \times \underbrace{\begin{pmatrix} 9.64 & 0.00 \\ 0.00 & 5.29 \end{pmatrix}}_{\Delta} \times \underbrace{\begin{pmatrix} 0.18 & 0.36 & 0.18 & 0.90 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.53 & 0.80 & 0.27 \end{pmatrix}}_{V^T}$$

query $q = (0\ 0\ 1\ 0\ 0)^T$ is transformed into
$q' = U^T \times q = (0.58\ \ 0.00)^T$ and evaluated on V^T

the new document $d8 = (1\ 1\ 0\ 0\ 0)^T$ is transformed into
$d8' = U^T \times d8 = (1.16\ \ 0.00)^T$ and appended to V^T

Example 2 for Latent Semantic Indexing

m=6 terms
- t1: bak(e,ing)
- t2: recipe(s)
- t3: bread
- t4: cake
- t5: pastr(y,ies)
- t6: pie

n=5 documents
- d1: How to bake bread without recipes
- d2: The classic art of Viennese Pastry
- d3: Numerical recipes: the art of scientific computing
- d4: Breads, pastries, pies and cakes: quantity baking recipes
- d5: Pastry: a book of best French recipes

$$A = \begin{pmatrix} 0.5774 & 0.0000 & 0.0000 & 0.4082 & 0.0000 \\ 0.5774 & 0.0000 & 1.0000 & 0.4082 & 0.7071 \\ 0.5774 & 0.0000 & 0.0000 & 0.4082 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.4082 & 0.0000 \\ 0.0000 & 1.0000 & 0.0000 & 0.4082 & 0.7071 \\ 0.0000 & 0.0000 & 0.0000 & 0.4082 & 0.0000 \end{pmatrix}$$

Example 2 for Latent Semantic Indexing (2)

$$A = \begin{pmatrix} 0.2670 & -0.2567 & 0.5308 & -0.2847 \\ 0.7479 & -0.3981 & -0.5249 & 0.0816 \\ 0.2670 & -0.2567 & 0.5308 & -0.2847 \\ 0.1182 & -0.0127 & 0.2774 & 0.6394 \\ 0.5198 & 0.8423 & 0.0838 & -0.1158 \\ 0.1182 & -0.0127 & 0.2774 & 0.6394 \end{pmatrix} \qquad \mathrm{U}$$

$$\times \begin{pmatrix} 1.6950 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 1.1158 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.8403 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.4195 \end{pmatrix} \qquad \Delta$$

$$\times \begin{pmatrix} 0.4366 & 0.3067 & 0.4412 & 0.4909 & 0.5288 \\ -0.4717 & 0.7549 & -0.3568 & -0.0346 & 0.2815 \\ 0.3688 & 0.0998 & -0.6247 & 0.5711 & -0.3712 \\ -0.6715 & -0.2760 & 0.1945 & 0.6571 & -0.0577 \end{pmatrix} \qquad \mathrm{V}^{\mathrm{T}}$$

Example 2 for Latent Semantic Indexing (3)

$$A_3 = \begin{pmatrix} 0.4971 & -0.0330 & 0.0232 & 0.4867 & -0.0069 \\ 0.6003 & 0.0094 & 0.9933 & 0.3858 & 0.7091 \\ 0.4971 & -0.0330 & 0.0232 & 0.4867 & -0.0069 \\ 0.1801 & 0.0740 & -0.0522 & 0.2320 & 0.0155 \\ -0.0326 & 0.9866 & 0.0094 & 0.4402 & 0.7043 \\ 0.1801 & 0.0740 & -0.0522 & 0.2320 & 0.0155 \end{pmatrix} = U_3 \times \Delta_3 \times V_3^T$$

Example 2 for Latent Semantic Indexing (4)

query q: baking bread

$q = (1\ 0\ 1\ 0\ 0\ 0)^T$

transformation into topic space with k=3

$q' = U_k^T \times q = (0.5340\ \ -0.5134\ \ 1.0616)^T$

scalar product similarity in topic space with k=3:

$sim(q, d1) = {V_k^T}_{*1} \times q' \approx 0.86$ $\qquad$ $sim(q, d2) = {V_k^T}_{*2} \times q \approx -0.12$

$sim(q, d3) = {V_k^T}_{*3} \times q' \approx -0.24$ $\qquad$ etc.

Folding-in of a new document d6:

algorithmic recipes for the computation of pie

$d6 = (0\ \ 0.7071\ \ 0\ \ 0\ \ 0\ \ 0.7071)^T$

transformation into topic space with k=3

$d6' = U_k^T \times d6 \approx (0.5\ \ -0.28\ \ -0.15)$

$d6'$ is appended to V_k^T as a new column

Multilingual Retrieval with LSI

- Construct LSI model (U_k, Δ_k, V_k^T) from **training documents** that are available in multiple languages:
  - consider **all language variants** of the same document as a **single document** and
  - extract all terms or words for all languages.
- Maintain index for further documents by „folding-in“, i.e. mapping into topic space and appending to V_k^T.
- **Queries** can now be asked **in any language**, and the query results include documents from all languages.

<u>Example:</u>

d1: ***How to bake bread without recipes.***
Wie man ohne Rezept Brot backen kann.

d2: ***Pastry: a book of best French recipes.***
Gebäck: eine Sammlung der besten französischen Rezepte.

Terms are e.g. bake, bread, recipe, backen, Brot, Rezept, etc.
Documents and terms are mapped into compact topic space.

Connections between LSI and Clustering

LSI can also be seen as an
unsupervised clustering method (cf. ***spectral clustering***):
simple variant for k clusters
- map each data point into **k-dimensional space**
- assign each point to its highest-value dimension: **strongest spectral component**

Conversely, we could compute k clusters
for the data points (using any clustering algorithm) and
project data points onto k centroid vectors („axes“ of k-dim. space)
to represent data in LSI-style manner („***concept indexing*** (CI)“)

More General Matrix Factorizations

Non-negative Matrix Factorization (NMF)

$A_{m\times n} \approx L_{m\times k} \times R_{k\times n}$ to minimize $\left\| A - L^T R \right\|_F^2 = \sum_{i=1}^{m} \sum_{j=1}^{n} (A_{ij} - L^T R_{ij})^2$

with $L_{ij} \geq 0$ and $R_{ij} \geq 0$

Matrix Factorization with **L2 Regularizer**

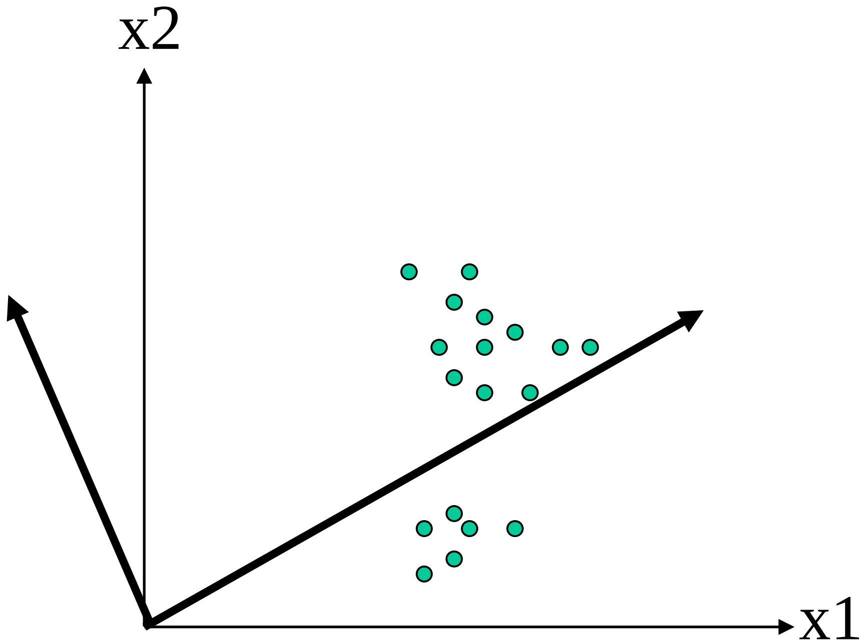
$A_{m\times n} \approx L_{m\times k} \times R_{k\times n}$ to minimize $\underbrace{\left\| A - L^T R \right\|_F^2}_{\textbf{data loss}} + \underbrace{\left\| L \right\|_F^2 + \left\| L \right\|_F^2}_{\textbf{model complexity}}$

Matrix Factorization with **L1 Regularizer** (favors sparseness)

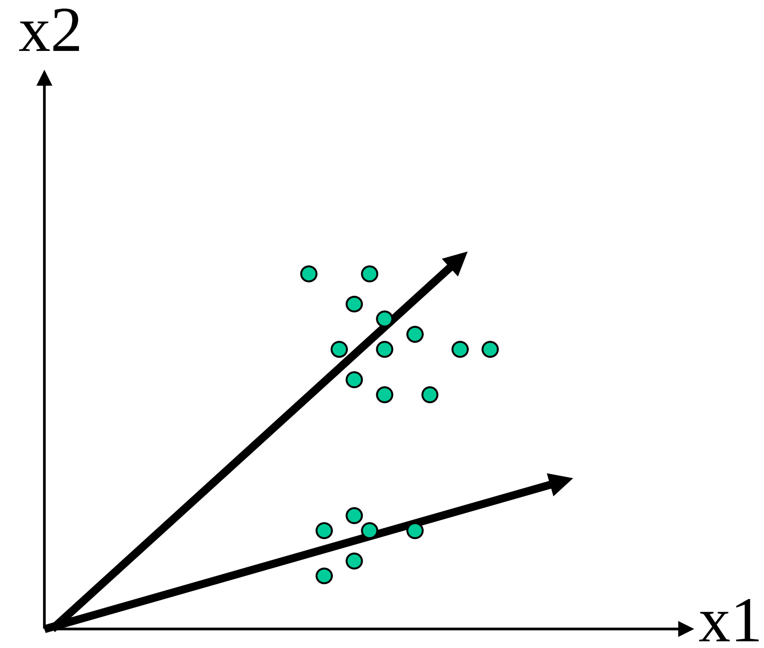
$A_{m\times n} \approx L_{m\times k} \times R_{k\times n}$ to minimize $\underbrace{\left\| A - L^T R \right\|_F^2}_{\textbf{data loss}} + \underbrace{\left\| L \right\|_1 + \left\| R \right\|_1}_{\textbf{model complexity}}$

→ numerical methods for non-convex optimization
e.g. iterative **gradient descent**

Power of Non-negative Matrix Factorization (NMF) vs. SVD



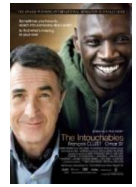
SVD of data matrix A

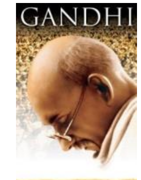


NMF of data matrix A

Application: Recommender Systems

Users × Items → Ratings



Alice 	3	2	?	3	1
Bob 	?	4	?	?	?
Claire 	5	?	4	4	?
Don 	4	?	5	1	4

Low-rank matrix factorization with regularization:

$$M_{u\times t} \approx L_{u\times k} \times R_{k\times t}$$

such that $\Sigma_{ij}\,(\underbrace{M_{ij} - (L\times R)_{ij}}_{\text{data loss}} - \underbrace{b_i}_{\text{user bias}} - \underbrace{b_j}_{\text{item bias}})^2 + \underbrace{\lambda\,(\|L\|_2 + \|R\|_2)}_{\text{regularizer}} = \min!$

alternatively:

$\ldots.. + \lambda\,(\|L\|_1 + \|R\|_1) = \min!$

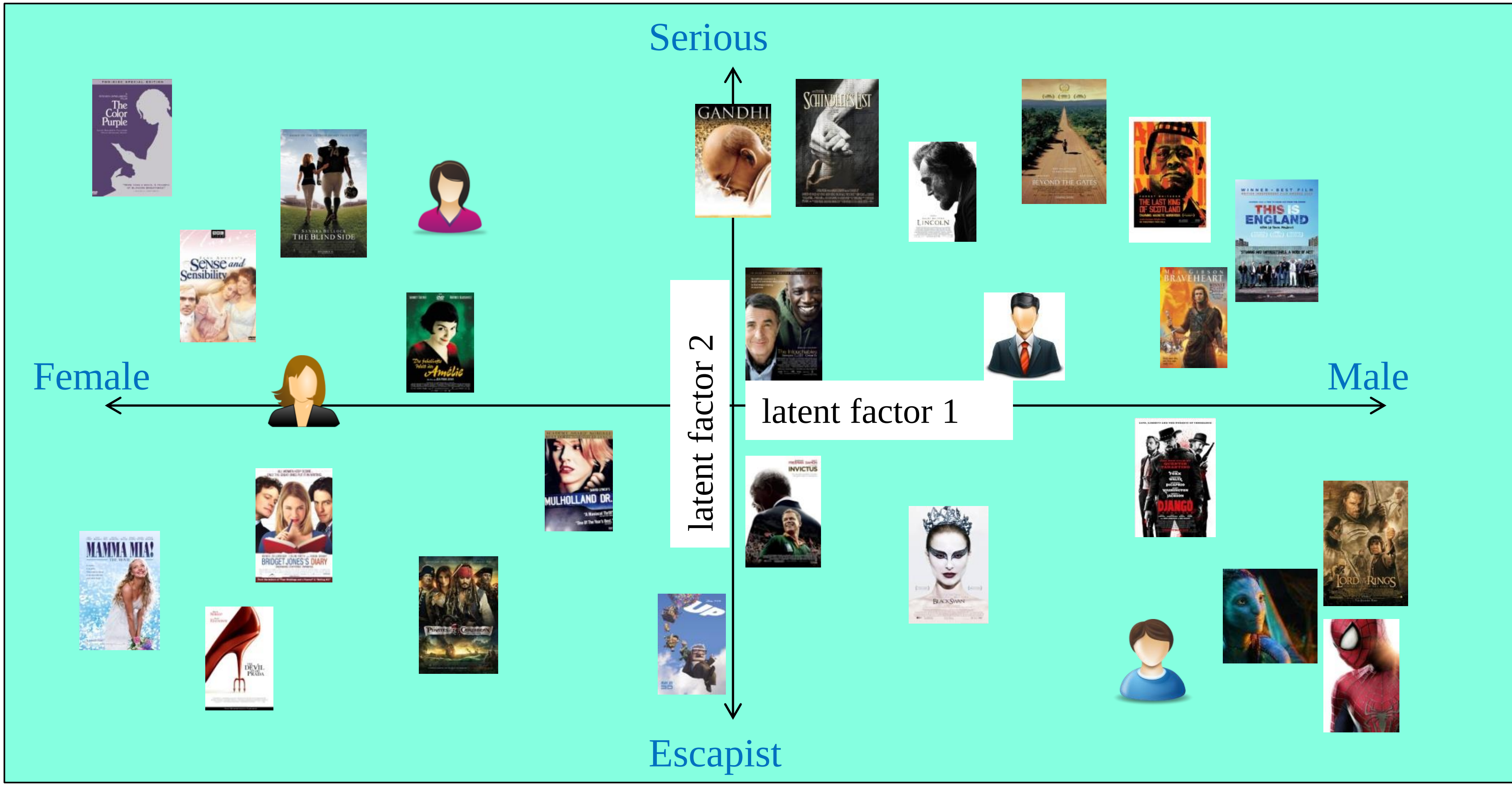
possibly with constraints: $L_{ij} \geq 0$ and $R_{ij} \geq 0$

plus temporal bias …

plus user-user profile sim …

plus item-item contents sim …

Application: Recommender Systems



also applicable to social graphs, and
co-occurrence graphs from user logs, text mining, etc.
for recommending „friends“, communities, bars, songs, etc. (see IRDM Chapter 7)
→ huge size poses scalability challenge

LSI Issues

+ Elegant well-founded model with
  automatic consideration of **term-term (cor)relations**
  (incl. synonymy/homonymy, morphological variations, cross-lingual)

– **Model Selection**: choice of low rank k not easy

– **Computational and storage cost**:
  term-doc matrix is sparse, SVD factors are dense
  SVD does not scale to Web dimensions (10s of Mio‘s to 100s of Bio‘s))

– Unconvincing results for **IR benchmarks** and Web search

13.4.2 Probabilistic Aspect Models
(pLSI, LDA, …)

- each **document d** is viewed as a mix of **(latent) topics (aspects) z**, each with a certain probability (summing up to 1)
- each topic generates **words w with topic-specific probabilities**

- P[wdz]: prob. of word w occurring in doc d about topic z
- we postulate: conditional independence of w and d given z

$$P[wdz] = P[wd|z]\, P[z] = P[w|z]\, P[d|z]\, P[z]$$

$$P[wd] = \Sigma_z\, P[w|z]\, P[d|z]\, P[z]$$

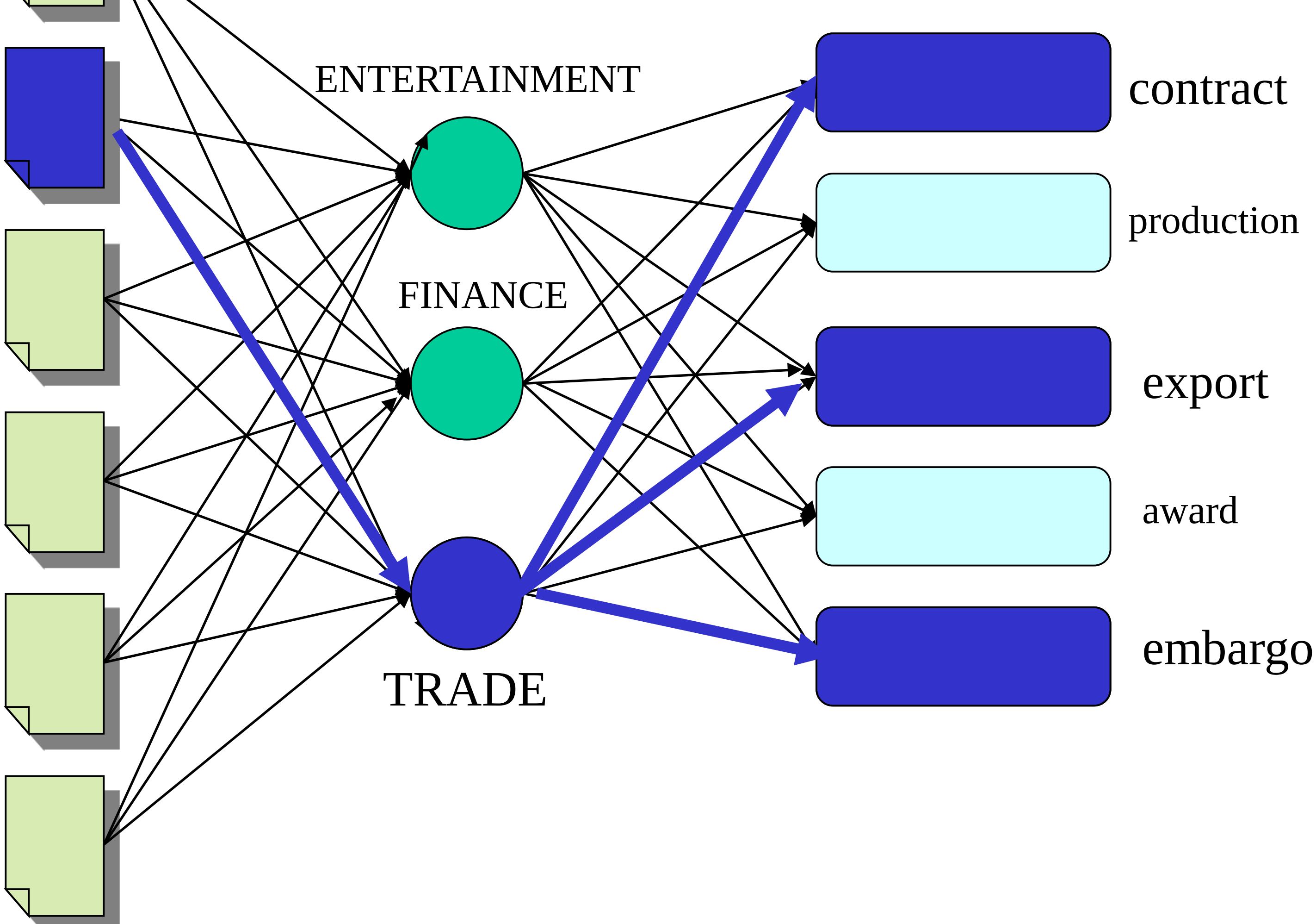
$$P[w|d] = \Sigma_z\, P[z|d]\, P[w|z]$$

generative model

Probabilistic LSI (pLSI)

$$P[w \mid d] = \sum_z P[z \mid d] \cdot P[w \mid z]$$

d and w conditionally independent given z



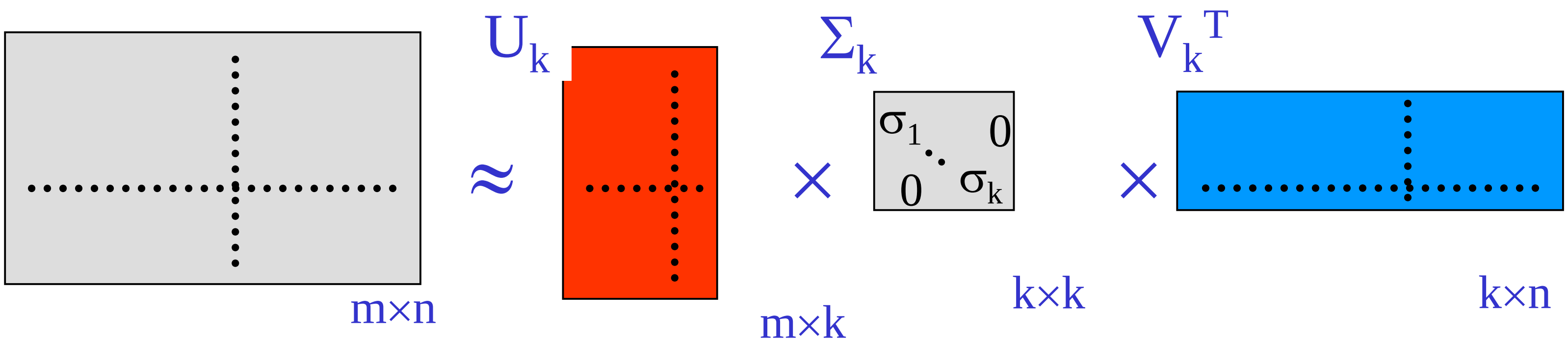
documents d

latent concepts z (aspects)

terms w (words)

Relationship of pLSI to LSI

$$P[w,d] = \sum_z P[w|z] \cdot P[z] \cdot P[d|z]$$



term probs per concept

concept probs

doc probs per concept

Key difference to LSI:
- non-negative matrix decomposition
- with L1 normalization

Key difference to LMs:
- no generative model for docs
- tied to given corpus

Learning and Using the pLSI Model

Parameter estimation:
given (d,w) data and #aspects k,
estimate P[z|d] and P[w|z] by EM
(Expectation Maximization, see Chapter 5: EM Clustering)
or gradient-descent methods for analytically intractable MLE or MAP

Query processing:
$q = \{w_1 \ldots w_n\}$ is „folded in“ (via EM and learned model)
to compute P[z|q]: aspect vector that best explains the query

Ranking of query results:
compare the aspect vectors of query and candidate documents
by Kullback-Leibler divergence or other similarity measure (e.g. cosine)

Experimental Results: Example

- Concepts (10 of 128) extracted from Science Magazine articles (12K)

P(w|z)

universe	0.0439	drug	0.0672	cells	0.0675	sequence	0.0818	years	0.156
galaxies	0.0375	patients	0.0493	stem	0.0478	sequences	0.0493	million	0.0556
clusters	0.0279	drugs	0.0444	human	0.0421	genome	0.033	ago	0.045
matter	0.0233	clinical	0.0346	cell	0.0309	dna	0.0257	time	0.0317
galaxy	0.0232	treatment	0.028	gene	0.025	sequencing	0.0172	age	0.0243
cluster	0.0214	trials	0.0277	tissue	0.0185	map	0.0123	year	0.024
cosmic	0.0137	therapy	0.0213	cloning	0.0169	genes	0.0122	record	0.0238
dark	0.0131	trial	0.0164	transfer	0.0155	chromosome	0.0119	early	0.0233
light	0.0109	disease	0.0157	blood	0.0113	regions	0.0119	billion	0.0177
density	0.01	medical	0.00997	embryos	0.0111	human	0.0111	history	0.0148

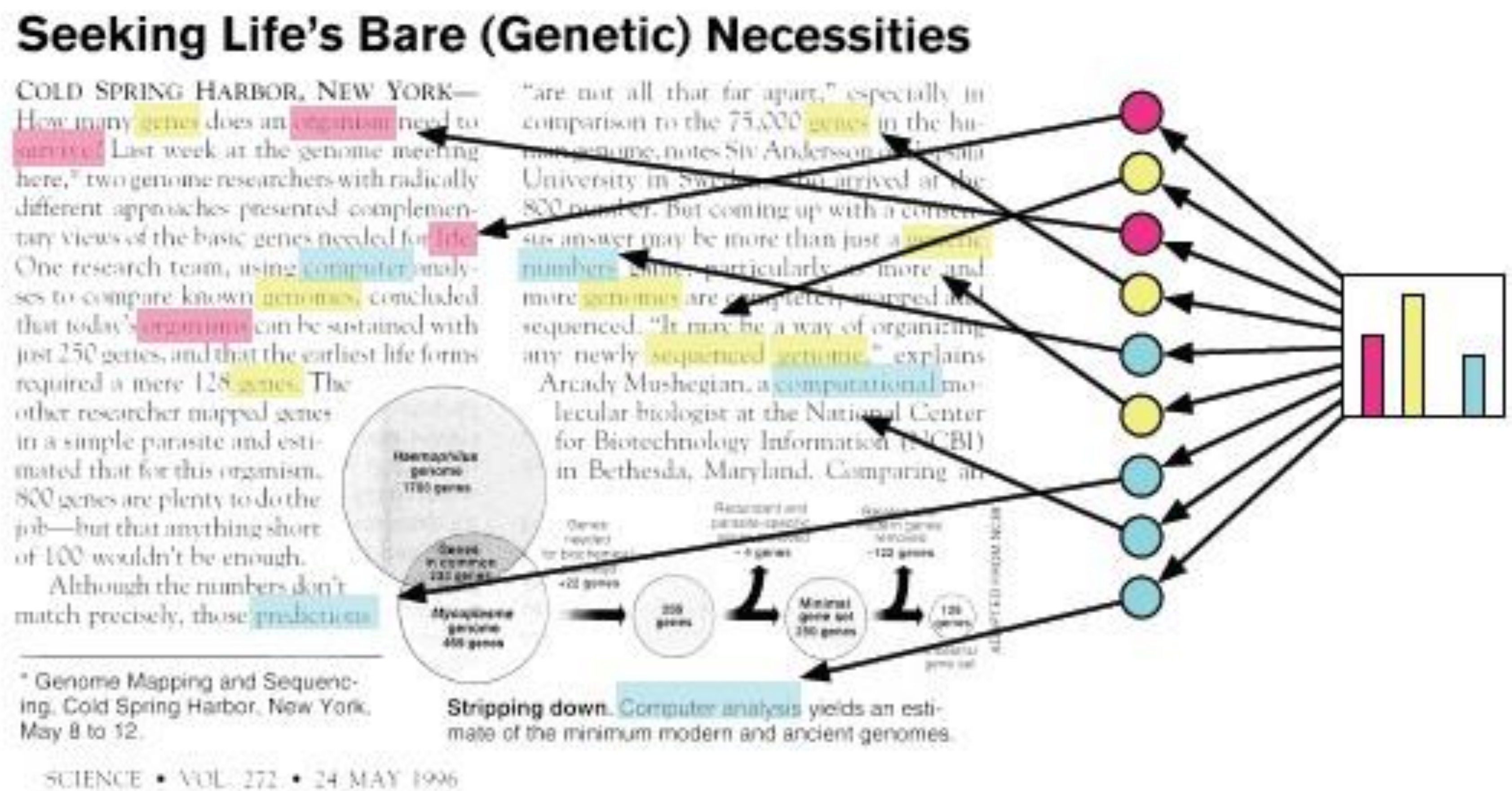
P(w|z)

bacteria	0.0983	male	0.0558	theory	0.0811	immune	0.0909	stars	0.0524
bacterial	0.0561	females	0.0541	physics	0.0782	response	0.0375	star	0.0458
resistance	0.0431	female	0.0529	physicists	0.0146	system	0.0358	astrophys	0.0237
coli	0.0381	males	0.0477	einstein	0.0142	responses	0.0322	mass	0.021
strains	0.025	sex	0.0339	university	0.013	antigen	0.0263	disk	0.0173
microbiol	0.0214	reproductive	0.0172	gravity	0.013	antigens	0.0184	black	0.0161
microbial	0.0196	offspring	0.0168	black	0.0127	immunity	0.0176	gas	0.0149
strain	0.0165	sexual	0.0166	theories	0.01	immunology	0.0145	stellar	0.0127
salmonella	0.0163	reproduction	0.0143	aps	0.00987	antibody	0.014	astron	0.0125
resistant	0.0145	eggs	0.0138	matter	0.00954	autoimmune	0.0128	hole	0.00824

Source: Thomas Hofmann, Tutorial at ADFOCS 2004

13.4.3 Latent Dirichlet Allocation (LDA)

- Multiple-cause mixture model
- Documents contain **multiple latent topics**
- Topics are expressed by **(multinomial) word distribution**
- LDA is a generative model for such docs (Dirichlet topic mixtures)



Seeking Life's Bare (Genetic) Necessities

COLD SPRING HARBOR, NEW YORK— How many genes does an organism need to survive? Last week at the genome meeting here,* two genome researchers with radically different approaches presented complementary views of the basic genes needed for life. One research team, using computer analyses to compare known genomes, concluded that today's organisms can be sustained with just 250 genes, and that the earliest life forms required a mere 128 genes. The other researcher mapped genes in a simple parasite and estimated that for this organism, 800 genes are plenty to do the job—but that anything short of 100 wouldn't be enough.

Although the numbers don't match precisely, those predictions "are not all that far apart," especially in comparison to the 75,000 genes in the human genome, notes Siv Andersson of Uppsala University in Sweden, who arrived at the 800 number. But coming up with a consensus answer may be more than just a genetic numbers game; particularly as more and more genomes are completely mapped and sequenced. "It may be a way of organizing any newly sequenced genome," explains Arcady Mushegian, a computational molecular biologist at the National Center for Biotechnology Information (NCBI) in Bethesda, Maryland. Comparing an

* Genome Mapping and Sequencing, Cold Spring Harbor, New York, May 8 to 12.

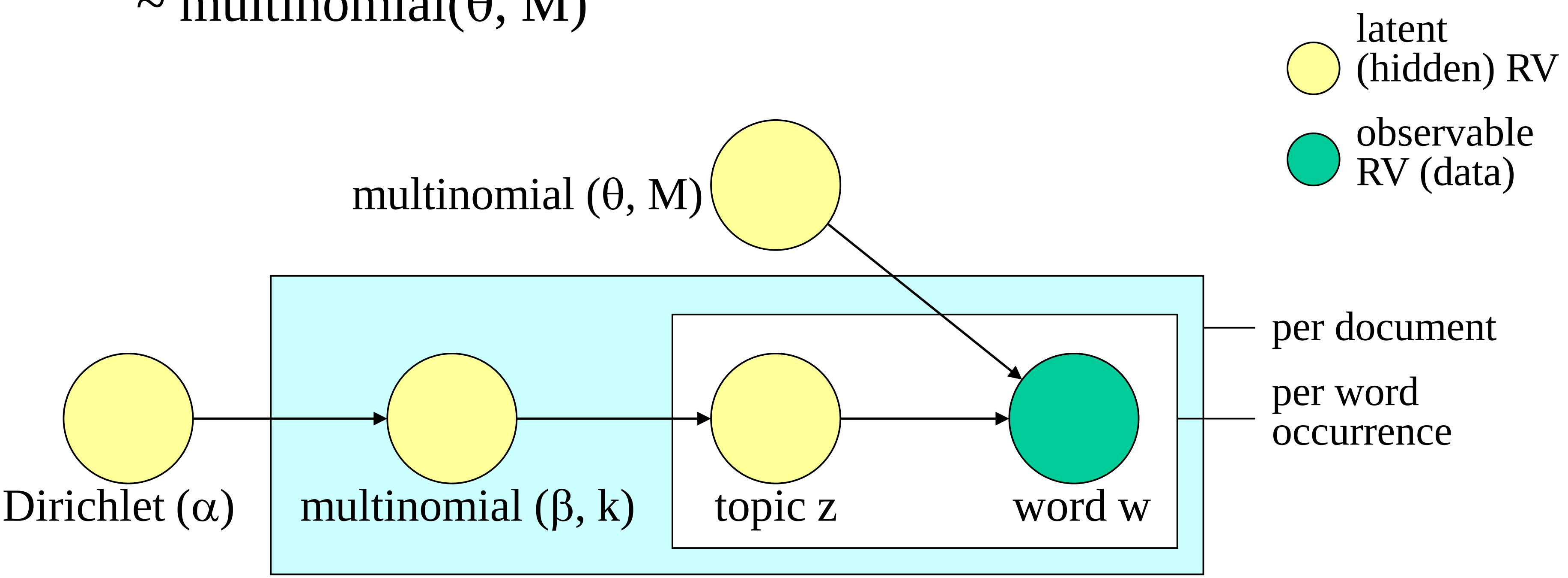
Stripping down. Computer analysis yields an estimate of the minimum modern and ancient genomes.

SCIENCE • VOL. 272 • 24 MAY 1996

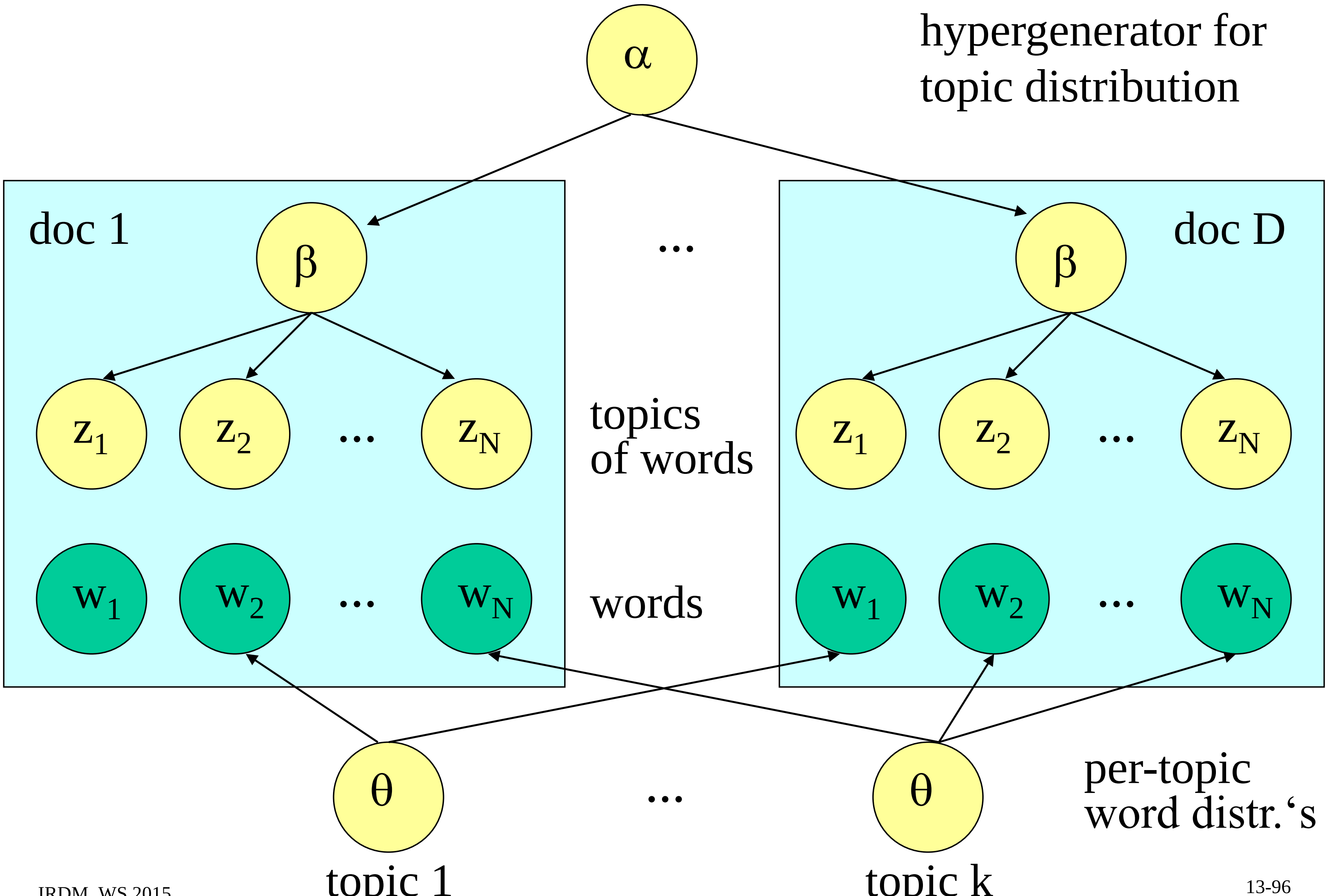
LDA Generative Model

for each doc d:

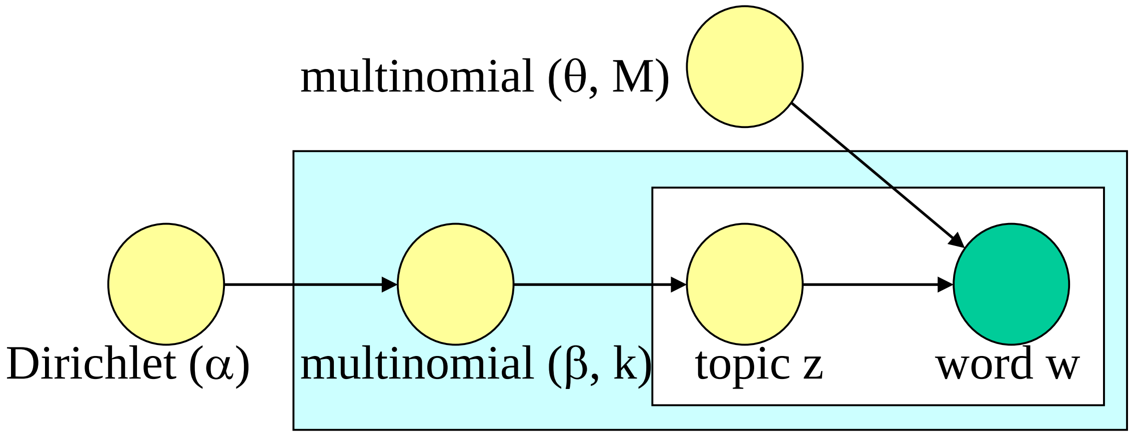
- choose doc length N (# word occurrences) ~ Poisson(λ)
- choose topic-probability params β ~ Dirichlet(α)
- for each of the N word occurrences in d (at position n):
  - choose one of k topics z_n ~ multinomial(β, k)
  - choose one of M words w_n from per-topic distribution ~ multinomial(θ, M)



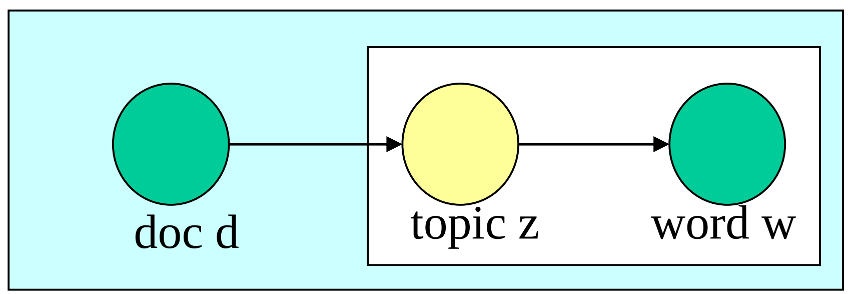
LDA Instance-Level Model



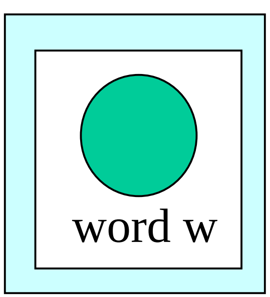
Comparison to Other Latent-Topic Models



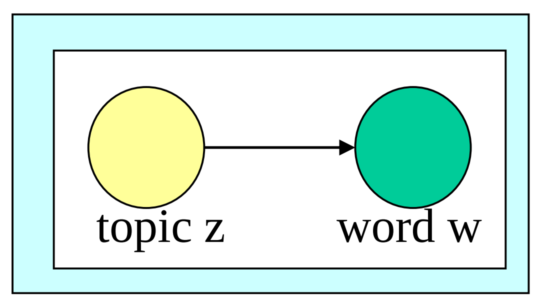
LDA



pLSI
aspect model



simple
unigram
model



discrete
univariate
distribution

single-cause
mixture of
unigrams

LDA Parameter Estimation

for doc x (if β were known):

$$P[x \mid \beta, \theta] = \prod_{n=1}^{N} \sum_{z_n=1}^{k} P[z_n \mid \beta] P[x_n \mid \theta_{z_n}]$$

$$= \prod_{n=1}^{N} \sum_{z_n=1}^{k} \beta_{z_n} \theta_{z_n, x_n}$$

with unknown β: $P[x \mid \alpha, \theta] = \int_\beta P[\beta \mid \alpha] \left(\prod_{n=1}^{N} \sum_{z_n=1}^{k} \beta_{z_n} \theta_{z_n, x_n} \right) d\beta$

$\Leftrightarrow$

$$P[x \mid \alpha, \theta] = \frac{\Gamma(\sum_y \alpha_y)}{\prod_y \Gamma(\alpha_y)} \int_\beta \prod_{y=1}^{k} \beta_y^{\alpha_y - 1} \left(\prod_{n=1}^{N} \sum_{z_n=1}^{k} \beta_{z_n} \theta_{z_n, x_n} \right) d\beta$$

log-likelihood function (for corpus of D docs) is analytically intractable
$\rightarrow$ EM algorithm or other variational methods or MCMC sampling

LDA Experimental Results: Example

"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

Source:
D.M. Blei, A.Y. Ng, M.I. Jordan: Latent Dirichlet Allocation, Journal of Machine Learning Research 2003

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.

13.4.4 Word2Vec: Latent Model for Term-Term Similarity

- view **distributional representation** (latent aspects) as a machine learning problem
- focus on **term vectors** for term-term similarity (terms: words, phrases, perhaps paragraphs)

Learn from **text windows C** of Web-scale corpora

Example: once upon a time in the west

window C of size 4

Aim to predict

P[w|C] = $P[w_t \mid w_{t-j}, \ldots, w_{t+j}$ with $1 \le j \le |C|/2]$

or

P[C|w] = $P[w_{t-j}, \ldots, w_{t+j}$ for $1 \le j \le |C|/2 \mid w_t]$

CBOW model (continuous bag of words)

continuous **Skip-Gram** model

https://code.google.com/p/word2vec/

Word2Vec: Learning Task

Objective: represent term w as vector $\vec{w}$ such that
for training corpus with term sequence $w_1 \dots w_T$:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{j\in C(t)} \log \frac{exp\,(\overrightarrow{w_j}^T\,\overrightarrow{w_t}\,)}{\sum_{v\in V} exp(\vec{v}^T\,\overrightarrow{w_t})} = \max!$$

softmax function based on (shallow) neural network

Approximate solution:
advanced machine learning methods (non-convex optimization)

Output:
distributional vector $\vec{w}$ for each term w (word or phrase or …)

Word2Vec: Examples

for given term w with vector $\vec{w}$ find closest $\vec{u}$ (e.g using cos)
$\rightarrow$ u is interpreted as most related term of w

term vector	**nearest vectors**
Redmond	Redmond Washington, Microsoft
graffiti	spray paint, grafitti, taggers
San_Francisco	Los_Angeles, Golden_Gate, Oakland, Seattle
Chinese_river	Yangtze_River, Yangtze, Yangtze_tributary

2 term vectors	**nearest vectors**
Czech + currency	koruna, Czech crown, Polish zloty, CTK
Vietnam + capital	Hanoi, Ho Chi Minh City, Viet Nam, Vietnamese
German + airlines	airline Lufthansa, carrier Lufthansa
Russian + river	Moscow, Volga River, upriver, Russia
French + actress	Juliette Binoche, Charlotte Gainsbourg

Word2Vec: Compositionality

Simply use linear algebra: vector addition and subtraction

Word2vec power
largely comes from data

X is to X' like Y to Y'
$\rightarrow$ vec(X) – vec(X') = vec(Y) – vec(Y')
$\rightarrow$ given Y, Y', X, solve for X': vec(X') = vec(Y) – vec(Y') +vec(X)

Y:Y'	**X:X'**
France:Paris	Italy:Rome, Japan:Tokyo
big:bigger	small:larger, cold:colder, quick:quicker
Einstein:scientist	Messi:midfielder, Mozart:violinist, Picasso:painter
Microsoft:Windows	Google:Android, IBM:Linux, Apple:iPhone
Sarkozy:France	Berlusconi:Italy, Merkel:Germany
Japan:sushi	Germany:bratwurst, France:tapas, USA:pizza

Can also be used to **automatically mine** linguistic regularities, e.g.:
vec(woman) – vec(man) = vec(queen) – vec(king) = vec(aunt)) – vec(uncle)

Summary of Section 13.4

- **Latent-Topic Models** can capture word correlations like synonymy in an implicit manner:
  - docs belong to (mixes of) latent topics, topics create words
- **LSI** is based on **spectral decomposition (SVD)** of term-doc matrix
  - elegant, effective, not scalable to Web size
- **pLSI** and **LDA** use non-negative, probabilistic decomposition
  - parameter estimation and query processing complex & expensive
- Other interesting models: **co-clustering**, **word2vec**, …
- none of these scales to Bios. of docs and Web workload
- all have a **model selection issue**: # topics (aspects)

Additional Literature for Section 13.4

- M.W. Berry, S.T. Dumais, G.W. O‘Brien: Using Linear Algebra for Intelligent Information Retrieval, SIAM Review Vol.37 No.4, 1995
- S. Deerwester, S.T. Dumais, G.W. Furnas, T.K. Landauer, R. Harshman: Indexing by Latent Semantic Analysis, JASIS 41(6), 1990
- H. Bast, D. Majumdar: Why Spectral Retrieval Works, SIGIR 2005
- T. Hofmann: Unsupervised Learning by Probabilistic Latent Semantic Analysis, Machine Learning 42, 2001
- T. Hofmann: Matrix Decomposition Techniques in Machine Learning and Information Retrieval, Tutorial Slides, ADFOCS 2004
- D. Blei, A. Ng, M. Jordan: Latent Dirichlet Allocation, Journal of Machine Learning Research 3, 2003
- D. Blei: Probabilistic Topic Models, CACM 2012
- X. Wei, W.B. Croft: LDA-based Document Models for Ad-hoc Retrieval, SIGIR‘06
- I.S. Dhillon, S. Mallela, D.S. Modha: Information-theoretic Co-clustering. KDD‘03
- W. Xu, X. Liu, Y. Gong: Document Clustering based on Non-negative Matrix Factorization, SIGIR 2003
- T. Mikolov et al.: Distributed Representations of Words and Phrases and their Compositionality. NIPS 2013

Outline

13.1 IR Effectiveness Measures

13.2 Probabilistic IR

13.3 Statistical Language Model

13.4 Latent-Topic Models

13.5 Learning to Rank

13.5 Learning to Rank

Why?

- **Increasing complexity of combining all feature groups: doc contents, source authority, freshness, geo-location, language style, author's online behavior, etc. etc.**
- **High dynamics of contents and user interests**

How?

- **exploit user feedback on search-result quality**
- **train a machine-learning predictor: scoring function f (query features, doc features)**
- **use the learned scoring function (weights) to rank the answers of new queries**
- **re-train the scoring function periodically**

Learning-to-Rank (LTR) Framework

Treat scoring as a **function** of different m **input signals (feature families)** $\mathbf{x_i}$ with **weights (hyper-parameters)** $\boldsymbol{\alpha_i}$

$$\boldsymbol{score\ (d,q) = f\ (\ x_1, \ldots, x_m, \alpha_1, \ldots, \alpha_m\ )}$$

where the weights α_i need to be learned and
the x_i are derived from d and q and the context
(e.g. tokens and bigrams of d and q,
last update of d, age of d‘s Internet domain,
user‘s preceding query, last clicked doc, etc. etc.)

Training data: set of **queries** each with info about docs

- **pointwise:** set of (q,d) points with relevant and irrelevant docs
- **pairwise:** set of (d,d‘) pairs where d is preferred over d‘
- **listwise:** list of ranked docs in desc. order of relevance

Objective function for learning task varies with setting
and quality measure to optimize (e.g. precision, F1, NDCG, …)

Regression for Parameter Fitting

Linear Regression

Estimate $r(x) = E[Y \mid X_1=x_1 \wedge \ldots \wedge X_m=x_m]$ using a linear model

$Y = r(x) + \varepsilon = \beta_0 + \sum_{i=1}^{m} \beta_i x_i + \varepsilon$ with error ε with $E[\varepsilon]=0$

given n sample points $(x_1^{(i)}, \ldots, x_m^{(i)}, y^{(i)})$, i=1..n, the least-squares estimator (LSE) minimizes the quadratic error:

$$\sum_{i=1..n} \left(\left(\sum_{k=0..m} \beta_k x_k^{(i)} \right) - y^{(i)} \right)^2 =: E(\beta_0, \ldots, \beta_m) \qquad \text{(with } x_o^{(i)}=1\text{)}$$

Solve linear equation system: $\frac{\partial E}{\partial \beta_k} = 0$ for k=0, ..., m

equivalent to MLE $\vec{\beta} = (X^T X)^{-1} X^T Y$

with $Y = (y^{(1)} \ldots y^{(n)})^T$ and
$$X = \begin{pmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \ldots & x_m^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & \ldots & x_m^{(2)} \\ \ldots \\ 1 & x_1^{(n)} & x_2^{(n)} & \ldots & x_m^{(n)} \end{pmatrix}$$

Regression for Parameter Fitting
Logistic Regression

Estimate r(x) = E[Y | X=x] for **Bernoulli Y** using a logistic model

$$Y = r(x) + \varepsilon = \frac{e^{\beta_0 + \sum_{i=1}^{m} \beta_i x_i}}{1 + e^{\beta_0 + \sum_{i=1}^{m} \beta_i x_i}} + \varepsilon \qquad \text{log-linear}$$

with error ε with E[ε]=0

$\rightarrow$ solution for MLE for β_i values
based on **numerical gradient-descent** methods

Pointwise LTR with Linear Regression

given n samples (x_1,y_1), (x_2,y_2), …

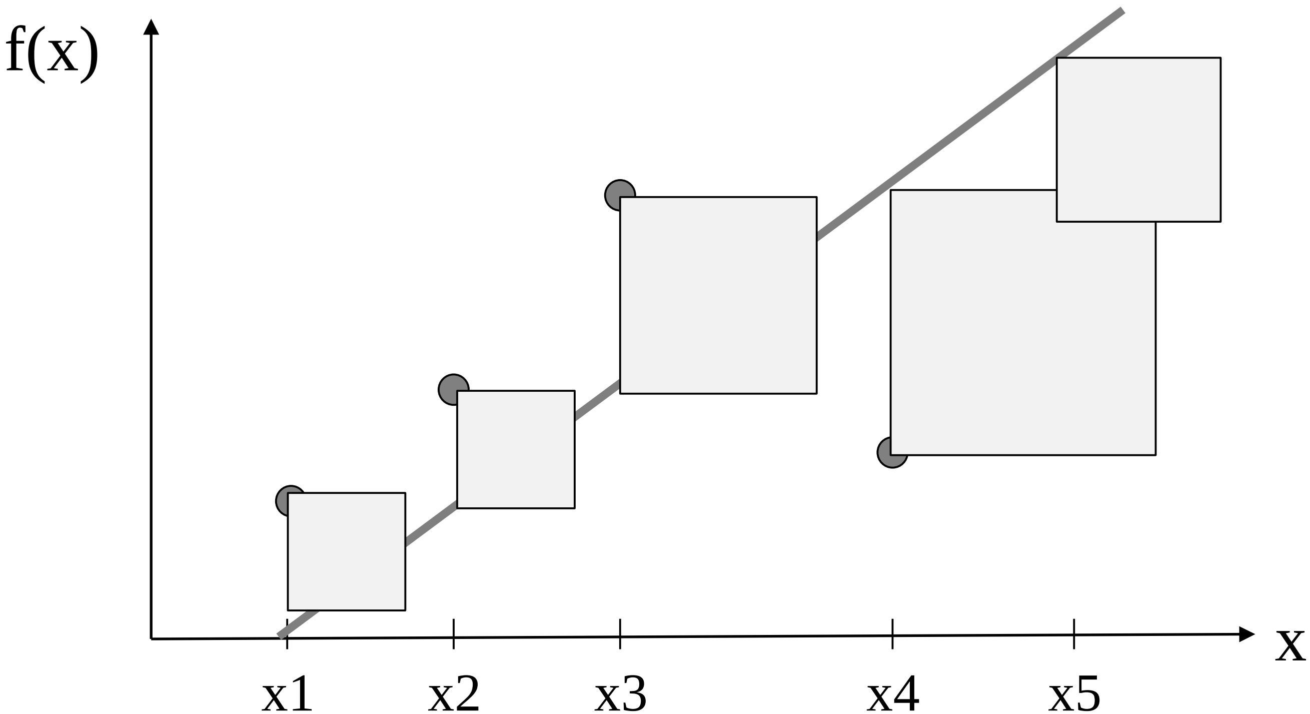
find **linear function f(x)** with **smallest L2 error** ~ $\mathbf{\Sigma_i\ (f(x_i)-y_i)^2}$

(method of least squares)

solve linear equation system (or SVD) over (x_i,y_i) matrix

generalizes to m-dimensional input $(x_{i1}, x_{i2}, \ldots, x_{im}, y_i)$, …

f(x1) = 0.4
f(x2) = 0.6
f(x3) = 0.9
f(x4) = 0.5
f(x5) = 0.8



Pointwise LTR with Logistic Regression

given n m-dim. samples $(x_{i1}, x_{i2}, \ldots, x_{im}, y_i)$ with $y_i \in \{0,1\}$

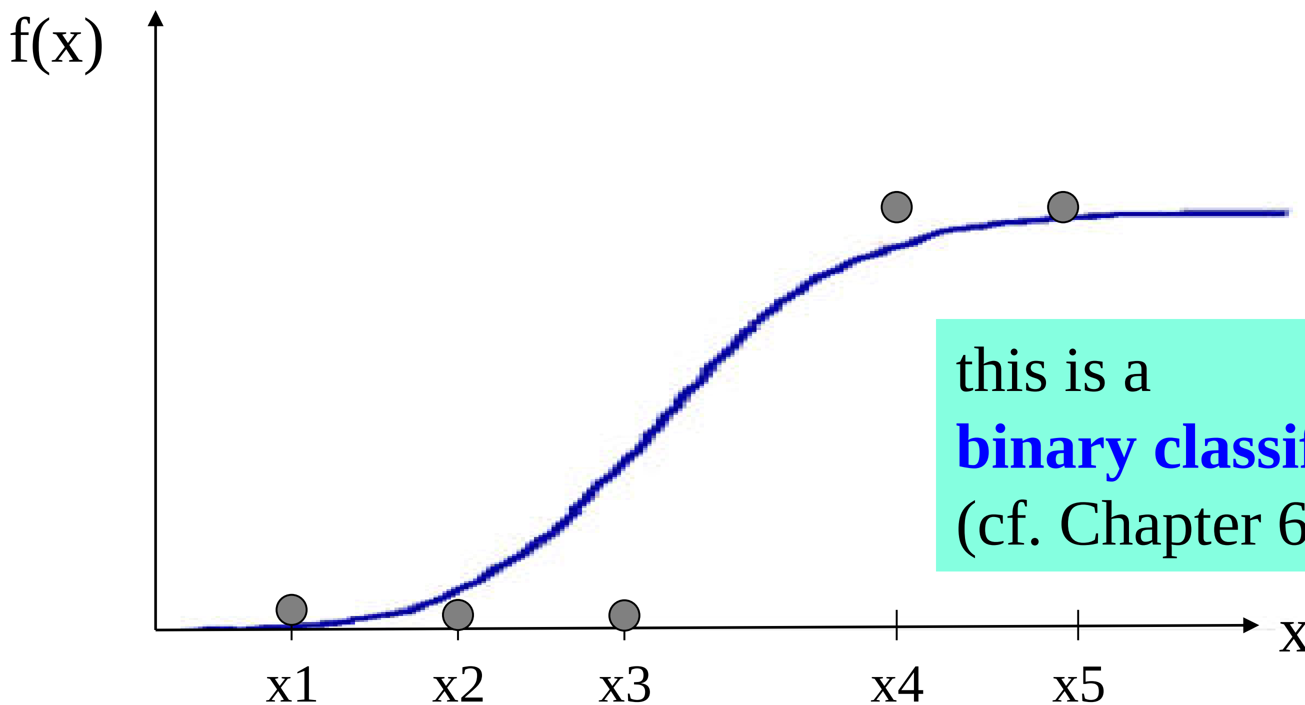
find **coefficient vector β** of logistic function f(x) with

smallest log-linear error $\sim \sum_{i=1..n} (y_i \beta^T x_i - \log(1+e^{\beta^T x_i})) + \|\beta\|_1$

data error (log-likelihood) **model complexity (regularizer)**

solve numerically by iterative gradient-descent methods

$f(x1) = \bar{R} = 0$
$f(x2) = \bar{R} = 0$
$f(x3) = \bar{R} = 0$
$f(x4) = \boldsymbol{R = 1}$
$f(x5) = \boldsymbol{R = 1}$



this is a **binary classifier** (cf. Chapter 6)

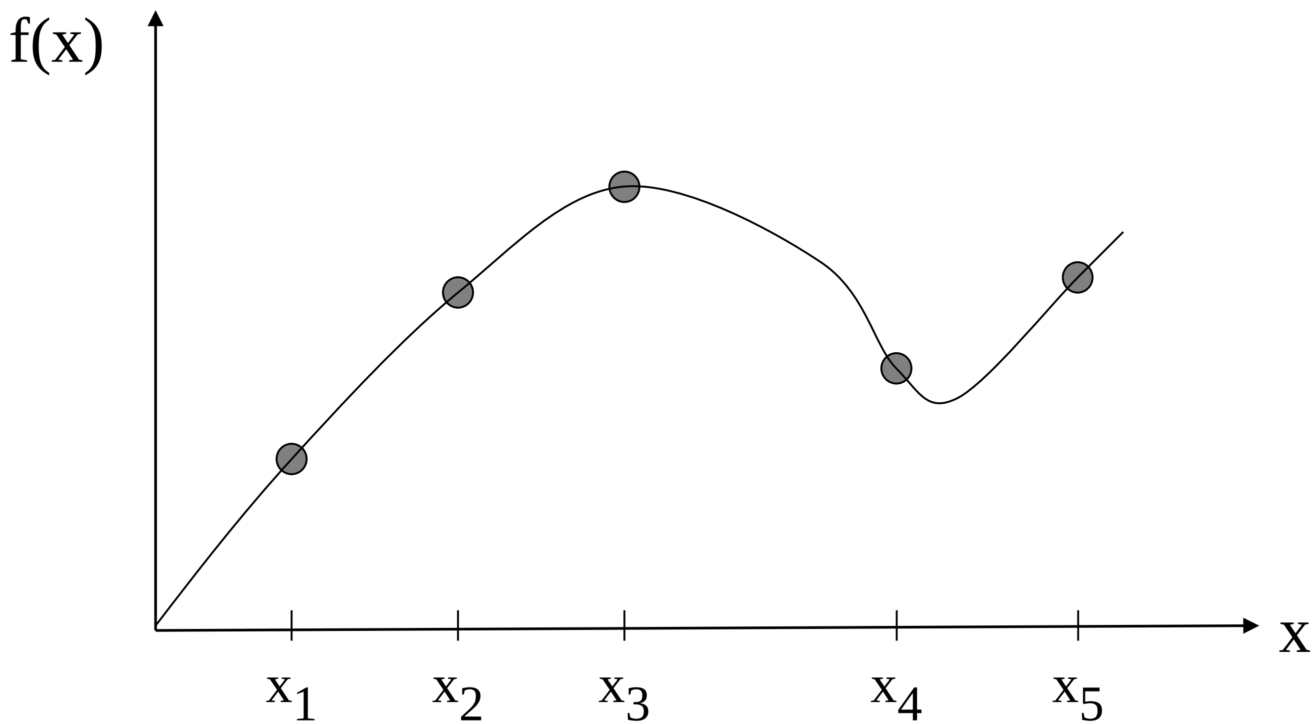
Pairwise LTR with Ordinal Regression

given $x_1, x_2, x_3, \ldots$ and preferences $x_i <_p x_j$ („x_i is better than x_j“)
find function f(x) with low violation of preference inequalities
$\rightarrow$ minimize ranking loss ~ $\Sigma_{i,j} L(x_i,x_j) + \ldots$ where

$L(x_i,x_j)=1$ if $x_i <_p x_j$ and $f(x_i) > f(x_j)$ or $x_i >_p x_j$ and $f(x_i) < f(x_j)$, 0 else

$\rightarrow$ advanced optimization methods (e.g. SVM-Rank [T. Joachims et al. 2005])

$x_1 <_p x_2$
$x_1 <_p x_3$
$x_4 <_p x_2$
$x_3 <_p x_5$
$x_4 <_p x_5$



Additional Literature for Section 13.5

- Tie-Yan Liu: Learning to Rank for Information Retrieval, Springer 2011, also in: Foundations and Trends in Information Retrieval 3 (3): 225–331, 2009
- R. Herbrich, T. Graepel, K. Obermayer: Large margin rank boundaries for ordinal regression. In: Advances in Large Margin Classifiers, MIT Press, 1999
- T. Joachims: Optimizing Search Engines using Clickthrough Data, KDD 2002
- T. Joachims, F. Radlinski: Query Chains: Learning to Rank from Implicit Feedback, KDD 2005
- T. Joachims et al.: Accurately Interpreting Clickthrough Data as Implicit Feedback, SIGIR 2005
- C.J.C. Burges et al.: Learning to rank using gradient descent. ICML 2005

Summary of Chapter 13

- **Probabilistic IR** and **Statistical Language Models** are the state-of-the-art ranking methods

- LMs are very versatile and composable

- **Latent Topic Models (LSI, LDA)** are powerful for consideration of term-term (cor)relations, but do not scale to Web

- **Learning-to-Rank** is very powerful and used for Web search, for training hyper-parameters of different feature groups and scoring models