Retrieval

Introduction to

Information Retrieval

CS276: Information Retrieval and Web Search

Christopher Manning and Pandu Nayak

Lecture 13: Latent Semantic Indexing

Today's topic

Latent Semantic Indexing

- Term-document matrices are very large
- But the number of topics that people talk about is small (in some sense)
  - Clothes, movies, politics, …
- Can we represent the term-document space by a lower

Linear Algebra

Eigenvalues & Eigenvectors

- **Eigenvectors** (for a square $m \times m$ matrix $\mathbf{S}$)

$$\mathbf{S}\mathbf{v} = \lambda \mathbf{v}$$

(right) eigenvector — eigenvalue

$\mathbf{v} \in \mathbb{R}^m \neq \mathbf{0}$ — $\lambda \in \mathbb{R}$

Example

$$\begin{pmatrix} 6 & -2 \\ 4 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \end{pmatrix} = \begin{pmatrix} 2 \\ 4 \end{pmatrix} = 2 \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

- How many eigenvalues are there at most?

$$\mathbf{S}\mathbf{v} = \lambda \mathbf{v} \iff (\mathbf{S} - \lambda \mathbf{I})\, \mathbf{v} = \mathbf{0}$$

only has a non-zero solution if $|\mathbf{S} - \lambda \mathbf{I}| = 0$

This is a mth order equation in λ which can have **at most m distinct solutions** (roots of the characteristic polynomial) – <u>can be complex even though S is real.</u>

Matrix–vector multiplication

$$S = \begin{bmatrix} 30 & 0 & 0 \\ 0 & 20 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

has eigenvalues 30, 20, 1 with corresponding eigenvectors

$$v_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \quad v_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \quad v_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

On each eigenvector, S acts as a multiple of the identity matrix: but as a different multiple on each.

Any vector (say $x = \begin{pmatrix} 2 \\ 4 \\ 6 \end{pmatrix}$) can be viewed as a combination of the eigenvectors: $x = 2v_1 + 4v_2 + 6v_3$

Matrix-vector multiplication

- Thus a matrix-vector multiplication such as Sx (S, x as in the previous slide) can be rewritten in terms of the eigenvalues/vectors:

$$Sx = S(2v_1 + 4v_2 + 6v_3)$$

$$Sx = 2Sv_1 + 4Sv_2 + 6Sv_3 = 2\lambda_1 v_1 + 4\lambda_2 v_2 + 6\lambda_3 v_3$$

$$Sx = 60v_1 + 80v_2 + 6v_3$$

- Even though x is an arbitrary vector, the action of S on x is determined by the eigenvalues/vectors.

Matrix-vector multiplication

- Suggestion: the effect of "small" eigenvalues is small.
- If we ignored the smallest eigenvalue (1), then instead of

$$\begin{pmatrix} 60 \\ 80 \\ 6 \end{pmatrix} \quad \text{we would get} \quad \begin{pmatrix} 60 \\ 80 \\ 0 \end{pmatrix}$$

- These vectors are similar (in cosine similarity, etc.)

Eigenvalues & Eigenvectors

For symmetric matrices, eigenvectors for distinct eigenvalues are **orthogonal**

$$Sv_{\{1,2\}} = \lambda_{\{1,2\}} v_{\{1,2\}}, \text{ and } \lambda_1 \neq \lambda_2 \Rightarrow v_1 \bullet v_2 = 0$$

All eigenvalues of a real symmetric matrix are **real**.

$$\text{for complex } \lambda, \text{ if } \; S - \lambda I$$

All eigenvalues of a positive semidefinite matrix are **non-negative**

$$\forall w \in \Re^n, w^T S w \geq 0, \text{ then if } Sv = \lambda v \Rightarrow \lambda \geq 0$$

Example

- Let $S = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$ ← Real, symmetric.
- Then $S - \lambda I = \begin{bmatrix} 2-\lambda & 1 \\ 1 & 2-\lambda \end{bmatrix} \Rightarrow$

$$|S - \lambda I| = (2-\lambda)^2 - 1 = 0.$$

- The eigenvalues are 1 and 3 (nonnegative, real).
- The eigenvectors are orthogonal (and real):

$$\begin{pmatrix} 1 \\ -1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

Plug in these values and solve for eigenvectors.

Eigen/diagonal Decomposition

- Let $\mathbf{S} \in \mathbb{R}^{m \times m}$ be a **square** matrix with ***m*** **linearly independent eigenvectors** (a "non-defective" matrix)
- **Theorem**: Exists an **eigen decomposition**

$$\mathbf{S} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^{-1}$$

diagonal

Unique for distinct eigen-values

  - (cf. matrix diagonalization theorem)
- Columns of **U** are the **eigenvectors** of **S**
- Diagonal elements of $\boldsymbol{\Lambda}$ are **eigenvalues** of $\mathbf{S}$

$$\boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_m), \quad \lambda_i \geq \lambda_{i+1}$$

Diagonal decomposition: why/how

Let **U** have the eigenvectors as columns: $U = \begin{bmatrix} v_1 & \dots & v_n \end{bmatrix}$

Then, **SU** can be written

$$SU = S\begin{bmatrix} v_1 & \dots & v_n \end{bmatrix} = \begin{bmatrix} \lambda_1 v_1 & \dots & \lambda_n v_n \end{bmatrix} = \begin{bmatrix} v_1 & \dots & v_n \end{bmatrix}\begin{bmatrix} \lambda_1 & & \\ & \dots & \\ & & \lambda_n \end{bmatrix}$$

Thus **SU=U**Λ, or **U**$^{-1}$**SU=**Λ

And **S=U**Λ**U**$^{-1}$.

Diagonal decomposition – example

Recall $S = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}; \lambda_1 = 1, \lambda_2 = 3.$

The eigenvectors $\begin{pmatrix} 1 \\ -1 \end{pmatrix}$ and $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$ form $U = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}$

Inverting, we have $U^{-1} = \begin{bmatrix} 1/2 & -1/2 \\ 1/2 & 1/2 \end{bmatrix}$

Recall $UU^{-1} = 1$.

Then, $\mathbf{S} = \mathbf{U}\Lambda\mathbf{U}^{-1} = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix}\begin{bmatrix} 1/2 & -1/2 \\ 1/2 & 1/2 \end{bmatrix}$

Example continued

Let's divide **U** (and multiply **U**$^{-1}$) by $\sqrt{2}$

Then, **S**=
$$\begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ -1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix}$$

Q Λ **(Q**$^{-1}$**= Q**^{T} **)**

Why? Stay tuned …

Symmetric Eigen Decomposition

- If $S \in \mathbb{R}^{m\times m}$ is a **symmetric** matrix:
- **Theorem**: There exists a (unique) **eigen decomposition** $S = Q\Lambda Q^T$
- where **Q** is **orthogonal:**
  - $\mathbf{Q}^{-1} = \mathbf{Q}^T$
  - Columns of **Q** are normalized eigenvectors
  - Columns are orthogonal.
  - (everything is real)

Exercise

- Examine the symmetric eigen decomposition, if any, for each of the following matrices:

$$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad \begin{bmatrix} 1 & 2 \\ -2 & 3 \end{bmatrix} \quad \begin{bmatrix} 2 & 2 \\ 2 & 4 \end{bmatrix}$$

Time out!

- I came to this class to learn about text retrieval and mining, not to have my linear algebra past dredged up again …
  - But if you want to dredge, Strang's Applied Mathematics is a good place to start.
- What do these matrices have to do with text?
- Recall M × N term-document matrices …
- But everything so far needs square matrices – so …

Similarity → Clustering

- We can compute the similarity between two document vector representations x_i and x_j by $x_i x_j^T$
- Let $X = [x_1 \ldots x_N]$
- Then XX^T is a matrix of similarities
- X_{ij} is symmetric
- So $XX^T = Q\Lambda Q^T$
- So we can decompose this similarity space into a set of orthonormal basis vectors (given in Q) scaled by the eigenvalues in Λ

Singular Value Decomposition

For an M × N matrix **A** of rank r there exists a factorization (Singular Value Decomposition = **SVD**) as follows:

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

M×M | M×N | V is N×N

(Not proven here.)

Singular Value Decomposition

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

M×M   M×N   V is N×N

- $AA^T = Q\Lambda Q^T$
- $AA^T = (U\Sigma V^T)(U\Sigma V^T)^T = (U\Sigma V^T)(V\Sigma U^T) = U\Sigma^2 U^T$

The columns of **U** are orthogonal eigenvectors of **AA^T**.

The columns of **V** are orthogonal eigenvectors of **A^TA**.

Eigenvalues $\lambda_1 \ldots \lambda_r$ of **AA^T** are the eigenvalues of **A^TA**.

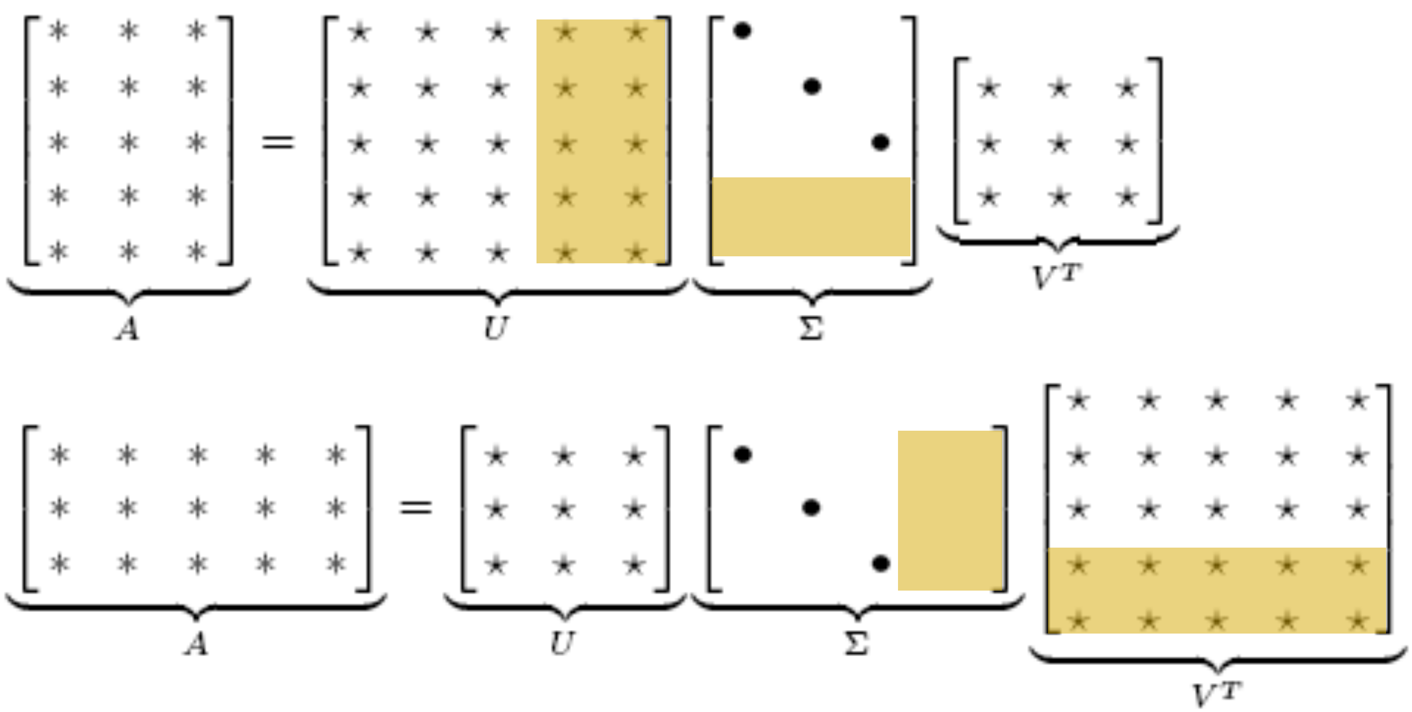
$$\sigma_i = \sqrt{\lambda_i}$$

$$\Sigma = diag(\sigma_1 \ldots \sigma_r)$$

Singular values

Singular Value Decomposition

- Illustration of SVD dimensions and sparseness



SVD example

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

Thus M=3, N=2. Its SVD is

$$\begin{bmatrix} 0 & 2/\sqrt{6} & 1/\sqrt{3} \\ 1/\sqrt{2} & -1/\sqrt{6} & 1/\sqrt{3} \\ 1/\sqrt{2} & 1/\sqrt{6} & -1/\sqrt{3} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & \sqrt{3} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{bmatrix}$$

Typically, the singular values arranged in decreasing order.

Low-rank Approximation

- SVD can be used to compute optimal **low-rank approximations**.
- Approximation problem: Find $\mathbf{A_k}$ of rank $\mathbf{k}$ such that

$$A_k = \min_{X:rank(X)=k} \| A - X \|_F$$

Frobenius norm

$$\|A\|_F \equiv \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} |a_{ij}|^2}\,.$$

A_k and X are both m×n matrices.

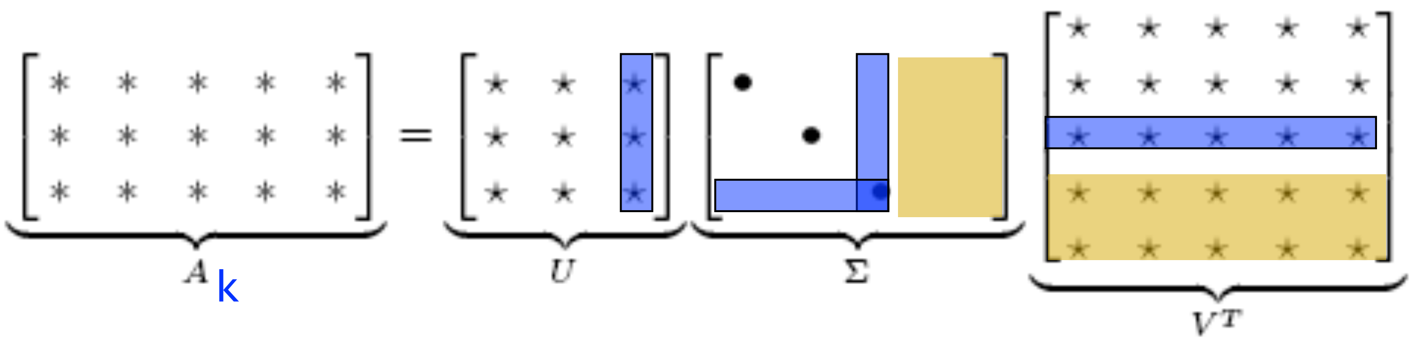
Typically, want k << r.

Low-rank Approximation

- Solution via SVD

$$A_k = U \operatorname{diag}(\sigma_1, \ldots, \sigma_k, \underbrace{0, \ldots, 0}) V^T$$

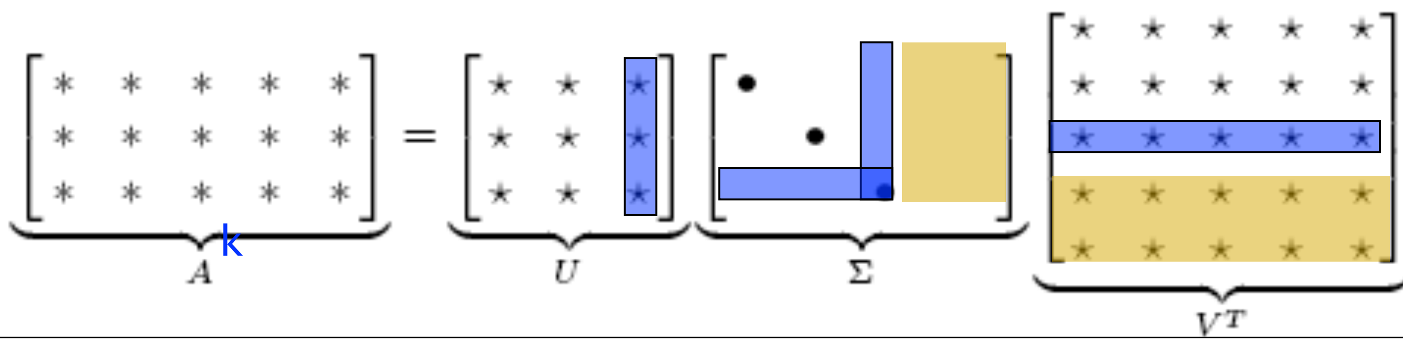
set smallest r-k singular values to zero



$$A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T$$

column notation: sum of rank 1 matrices

Reduced SVD

- If we retain only k singular values, and set the rest to 0, then we don't need the matrix parts in color
- Then Σ is k×k, U is M×k, V^T is k×N, and A_k is M×N
- This is referred to as the reduced SVD
- It is the convenient (space-saving) and … s
- 

Approximation error

- How good (bad) is this approximation?
- It's the best possible, measured by the Frobenius norm of the error:

$$\min_{X: rank(X)=k} A - X$$

where the σ_i are ordered such that $\sigma_i \geq \sigma_{i+1}$.

Suggests why Frobenius error drops as k increases.

SVD Low-rank approximation

- Whereas the term-doc matrix A may have M=50000, N=10 million (and rank close to 50000)
- We can construct an approximation A_{100} with rank 100.
  - Of all rank 100 matrices, it would have the lowest Frobenius error.
- Great … but why would we??
- Answer: Latent Semantic Indexing

C. Eckart, G. Young, *The approximation of a matrix by another of lower rank.* Psychometrika, 1, 211-218, 1936.

Latent Semantic

What it is

- From term-doc matrix A, we compute the approximation A_{k}.
- There is a row for each term and a column for each doc in A_k
- Thus docs live in a space of k<<r dimensions
  - These dimensions are not the original axes
- But why?

Vector Space Model: Pros

- **Automatic** selection of index terms
- **Partial matching** of queries and documents (dealing with the case where no document contains all search terms)
- **Ranking** according to **similarity score** (dealing with large result sets)
- **Term weighting** schemes (improves retrieval performance)
- Various extensions
  - Document clustering
  - Relevance feedback (modifying query vector)
- Geometric foundation

Problems with Lexical Semantics

- Ambiguity and association in natural language
  - **Polysemy**: Words often have a **multitude of meanings** and different types of usage (more severe in very heterogeneous collections).
  - The vector space model is unable to discriminate between different meanings of the same word.

$$\mathrm{sim}_{\mathrm{true}}(d, q) < \cos(\angle(\vec{d}, \vec{q}))$$

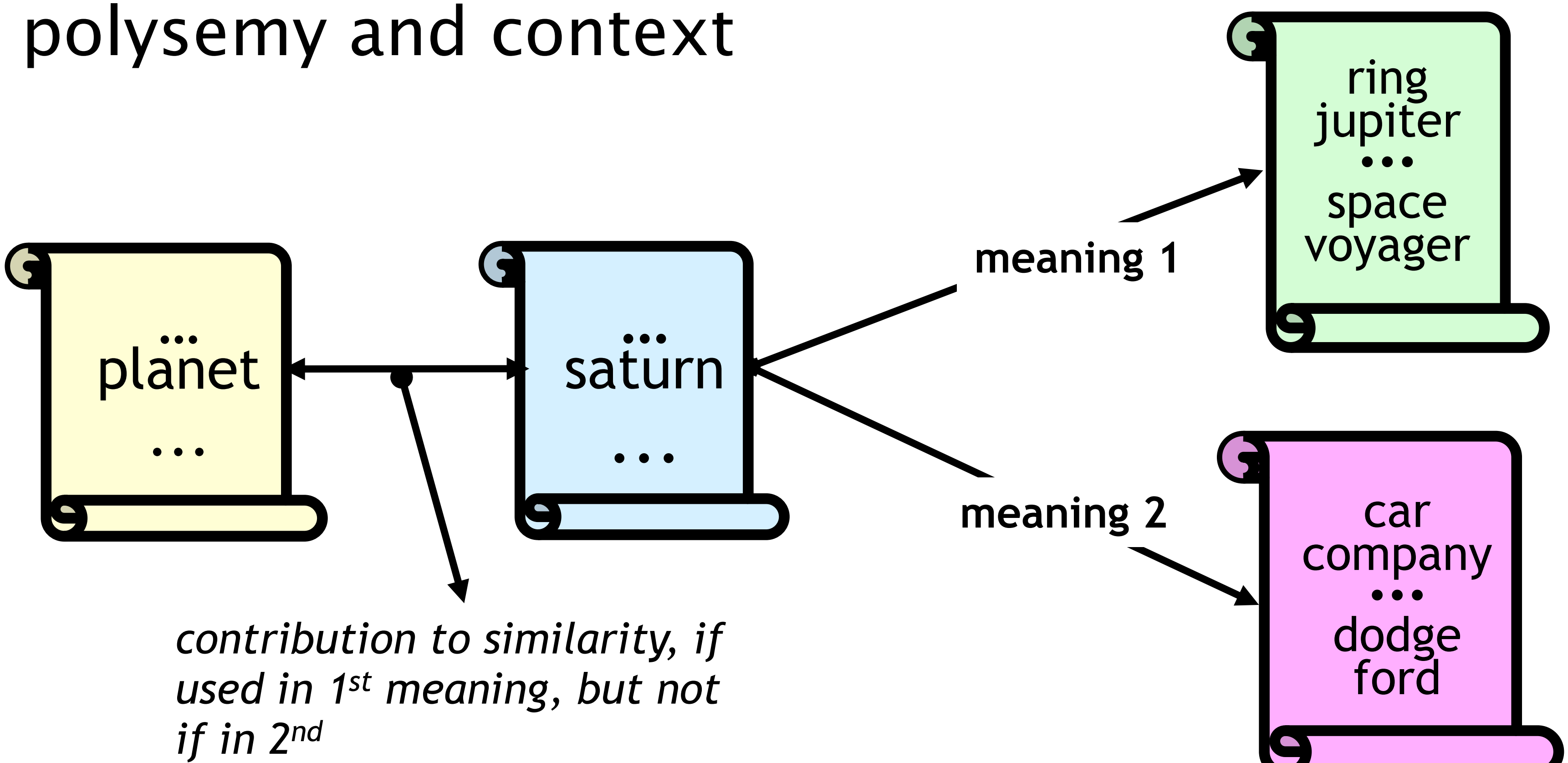
Problems with Lexical Semantics

- **Synonymy**: Different terms may have an **identical or a similar meaning** (weaker: words indicating the same topic).
- No associations between words are made in the vector space representation.

$$\mathrm{sim}_{\mathrm{true}}(d, q) > \cos(\angle(\vec{d}, \vec{q}))$$

Polysemy and Context

- Document similarity on single word level: polysemy and context



Latent Semantic Indexing (LSI)

- Perform a **low-rank approximation** of **document-term matrix** (typical rank **100-300**)
- General idea
  - Map documents (and terms) to a **low-dimensional** representation.
  - Design a mapping such that the low-dimensional space reflects **semantic associations** (latent semantic space).
  - Compute document similarity based on the **inner product** in this **latent semantic space**

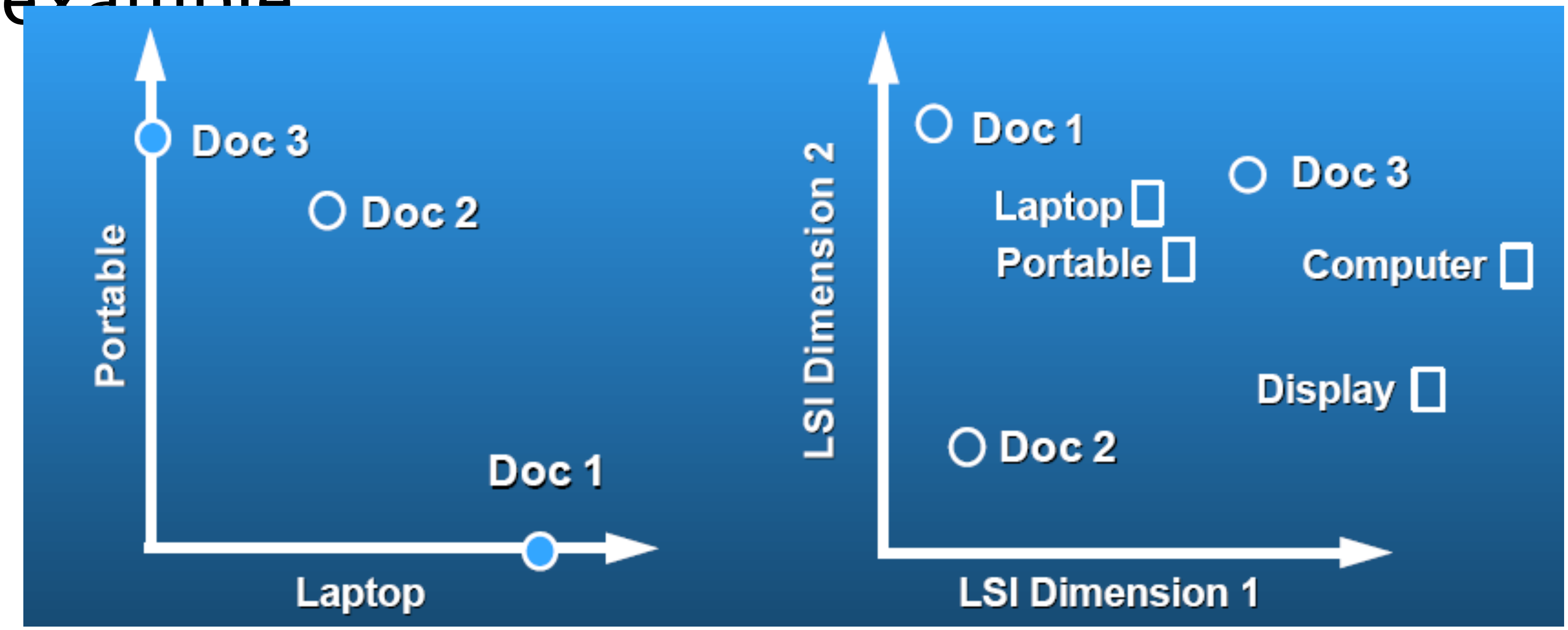
Goals of LSI

- LSI takes documents that are semantically similar (= talk about the same topics), but are not similar in the vector space (because they use different words) and re-represents them in a reduced vector space in which they have higher similarity.

- Similar terms map to similar location in low dimensional space
- Noise reduction by dimension reduction

Latent Semantic Analysis

- **Latent semantic space**: illustrating example



courtesy of Susan Dumais

Performing the maps

- Each row and column of A gets mapped into the k-dimensional LSI space, by the SVD.
- Claim – this is not only the mapping with the best (Frobenius error) approximation to A, but in fact improves retrieval.
- A query q is also mapped into this space, by

$$q_k = q^T U_k \Sigma_k^{-1}$$

  - Query NOT a sparse vector.

LSA Example

- A simple example term-document matrix (binary)

C	d_1	d_2	d_3	d_4	d_5	d_6
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
wood	1	0	0	1	1	0
tree	0	0	0	1	0	1

LSA Example

- Example of C = UΣVT: The matrix U

U	1	2	3	4	5
ship	−0.44	−0.30	0.57	0.58	0.25
boat	−0.13	−0.33	−0.59	0.00	0.73
ocean	−0.48	−0.51	−0.37	0.00	−0.61
wood	−0.70	0.35	0.15	−0.58	0.16
tree	−0.26	0.65	−0.41	0.58	−0.09

LSA Example

- Example of C = UΣVT: The matrix Σ

Σ	1	2	3	4	5
1	2.16	0.00	0.00	0.00	0.00
2	0.00	1.59	0.00	0.00	0.00
3	0.00	0.00	1.28	0.00	0.00
4	0.00	0.00	0.00	1.00	0.00
5	0.00	0.00	0.00	0.00	0.39

LSA Example

- Example of C = UΣV^T: The matrix V^T

V^T	d_1	d_2	d_3	d_4	d_5	d_6
1	−0.75	−0.28	−0.20	−0.45	−0.33	−0.12
2	−0.29	−0.53	−0.19	0.63	0.22	0.41
3	0.28	−0.75	0.45	−0.20	0.12	−0.33
4	0.00	0.00	0.58	0.00	−0.58	0.58
5	−0.53	0.29	0.63	0.19	0.41	−0.22

LSA Example: Reducing the dimension

U	1	2	3	4	5
ship	−0.44	−0.30	0.00	0.00	0.00
boat	−0.13	−0.33	0.00	0.00	0.00
ocean	−0.48	−0.51	0.00	0.00	0.00
wood	−0.70	0.35	0.00	0.00	0.00
tree	−0.26	0.65	0.00	0.00	0.00

Σ_2	1	2	3	4	5
1	2.16	0.00	0.00	0.00	0.00
2	0.00	1.59	0.00	0.00	0.00
3	0.00	0.00	0.00	0.00	0.00
4	0.00	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	0.00

V^T	d_1	d_2	d_3	d_4	d_5	d_6
1	−0.75	−0.28	−0.20	−0.45	−0.33	−0.12
2	−0.29	−0.53	−0.19	0.63	0.22	0.41
3	0.00	0.00	0.00	0.00	0.00	0.00
4	0.00	0.00	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	0.00	0.00

Original matrix C vs. reduced $C_2 = U\Sigma_2V^T$

C	d_1	d_2	d_3	d_4	d_5	d_6
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
wood	1	0	0	1	1	0
tree	0	0	0	1	0	1

C_2	d_1	d_2	d_3	d_4	d_5	d_6
ship	0.85	0.52	0.28	0.13	0.21	−0.08
boat	0.36	0.36	0.16	−0.20	−0.02	−0.18
ocean	1.01	0.72	0.36	−0.04	0.16	−0.21
wood	0.97	0.12	0.20	1.03	0.62	0.41
tree	0.12	−0.39	−0.08	0.90	0.41	0.49

Why the reduced dimension matrix is better

- Similarity of d2 and d3 in the original space: 0.
- Similarity of d2 and d3 in the reduced space: $0.52 * 0.28 + 0.36 * 0.16 + 0.72 * 0.36 + 0.12 * 0.20 + -0.39 * -0.08 \approx 0.52$

- Typically, LSA increases recall and hurts precision

Empirical evidence

- Experiments on TREC 1/2/3 – Dumais
- Lanczos SVD code (available on netlib) due to Berry used in these experiments
  - Running times of ~ one day on tens of thousands of docs [still an obstacle to use!]
- Dimensions – various values 250-350 reported. Reducing k improves recall.
  - (Under 200 reported unsatisfactory)
- Generally expect recall to improve – what about precision?

Empirical evidence

- Precision at or above median TREC precision
  - Top scorer on almost 20% of TREC topics
- Slightly better on average than straight vector spaces
- Effect of dimensionality:

Dimensions	Precision
250	0.367
300	0.371
346	0.374

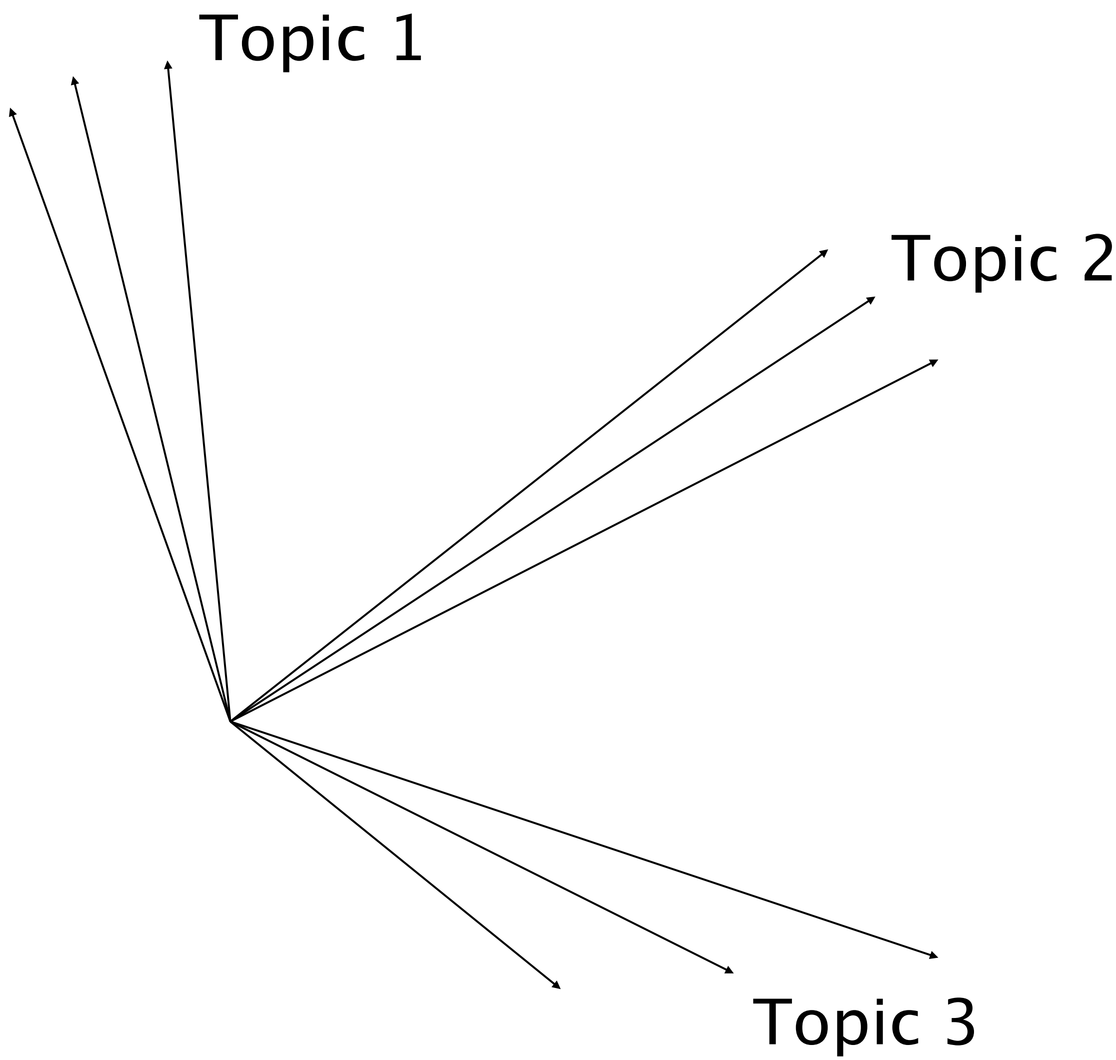
Failure modes

- Negated phrases
  - TREC topics sometimes negate certain query/terms phrases – precludes simple automatic conversion of topics to latent semantic space.
- Boolean queries
  - As usual, freetext/vector space syntax of LSI queries precludes (say) “Find any doc having to do with the following 5 companies”
- See Dumais for more.

But why is this clustering?

- We've talked about docs, queries, retrieval and precision here.
- What does this have to do with clustering?
- Intuition: Dimension reduction through LSI brings together "related" axes in the vector space.

Simplistic picture



Some wild extrapolation

- The "dimensionality" of a corpus is the number of distinct topics represented in it.
- More mathematical wild extrapolation:
  - if A has a rank k approximation of low Frobenius error, then there are no more than k distinct topics in the corpus.

LSI has many other applications

- In many settings in pattern recognition and retrieval, we have a feature-object matrix.
  - For text, the terms are features and the docs are objects.
  - Could be opinions and users …
  - This matrix may be redundant in dimensionality.
  - Can work with low-rank approximation.
  - If entries are missing (e.g., users' opinions), can recover if dimensionality is low.
- Powerful general analytical technique
  - Close, principled analog to clustering methods.

Resources

- IIR 18
- Scott Deerwester, Susan Dumais, George Furnas, Thomas Landauer, Richard Harshman. 1990. Indexing by latent semantic analysis. JASIS 41(6):391—407.