LSI, SVD and Data Management

Based on the Slides from CS276: Information Retrieval and Web Search

Christopher Manning and Pandu Nayak

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 dimensional latent space?

Linear Algebra Background

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

$$S = \begin{pmatrix} 30 & 0 & 0 \\ 0 & 20 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$ has eigenvalues $\lambda_1=30$, $\lambda_2=20$, $\lambda_3=1$ with corresponding eigenvectors

$$\vec{x_1} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \vec{x_2} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \text{ and } \vec{x_3} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

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

Any vector (say v= $\begin{pmatrix} 2 \\ 4 \\ 6 \end{pmatrix}$) can be viewed as a combination of the eigenvectors: $v = 2\vec{x}_1 + 4\vec{x}_2 + 6\vec{x}_3$

Matrix-vector multiplication

- Thus a matrix-vector multiplication such as Sv (S matrix, v a vector) can be rewritten in terms of the eigenvalues/vectors:

$$\begin{aligned} S\vec{v} &= S(2\vec{x_1} + 4\vec{x_2} + 6\vec{x_3}) \\ &= 2S\vec{x_1} + 4S\vec{x_2} + 6S\vec{x_3} \\ &= 2\lambda_1\vec{x_1} + 4\lambda_2\vec{x_2} + 6\lambda_3\vec{x_3} \\ &= 60\vec{x_1} + 80\vec{x_2} + 6\vec{x_3}. \end{aligned}$$

- Even though v is an arbitrary vector, the action of S on v 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**

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

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

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

$$\overbrace{w^T S w >= 0, \text{ for all } w}$$

Example

- Let $S = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$ $\longleftarrow$ $\boxed{\text{Real, symmetric.}}$

- 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):
$$\begin{pmatrix} 1 \\ -1 \end{pmatrix} \text{ and } \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

Eigen/diagonal Decomposition

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

  diagonal

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

  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 - example

Recall $S = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$ ith eigenvalues 3 and 1

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

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

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

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

Example continued

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

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

U $\not\subset$ **(U^{-1}= U^{T})**

Why? Stay tuned …

Symmetric Eigen Decomposition

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

Singular Value Decomposition

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

$$A = \underbrace{U}_{\boxed{M \times M}}\ \underbrace{\Sigma}_{\boxed{M \times N}}\ \underbrace{V^T}_{\boxed{V \text{ is } N \times N}}$$

(Not proven here.)

Singular Value Decomposition

$$A = \underset{\boxed{M\times M}}{U}\,\underset{\boxed{M\times N}}{\Sigma}\,\underset{\boxed{\text{V is } N\times N}}{V^T}$$

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

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

The columns of **V** are orthogonal eigenvectors of **A^{T}A**.

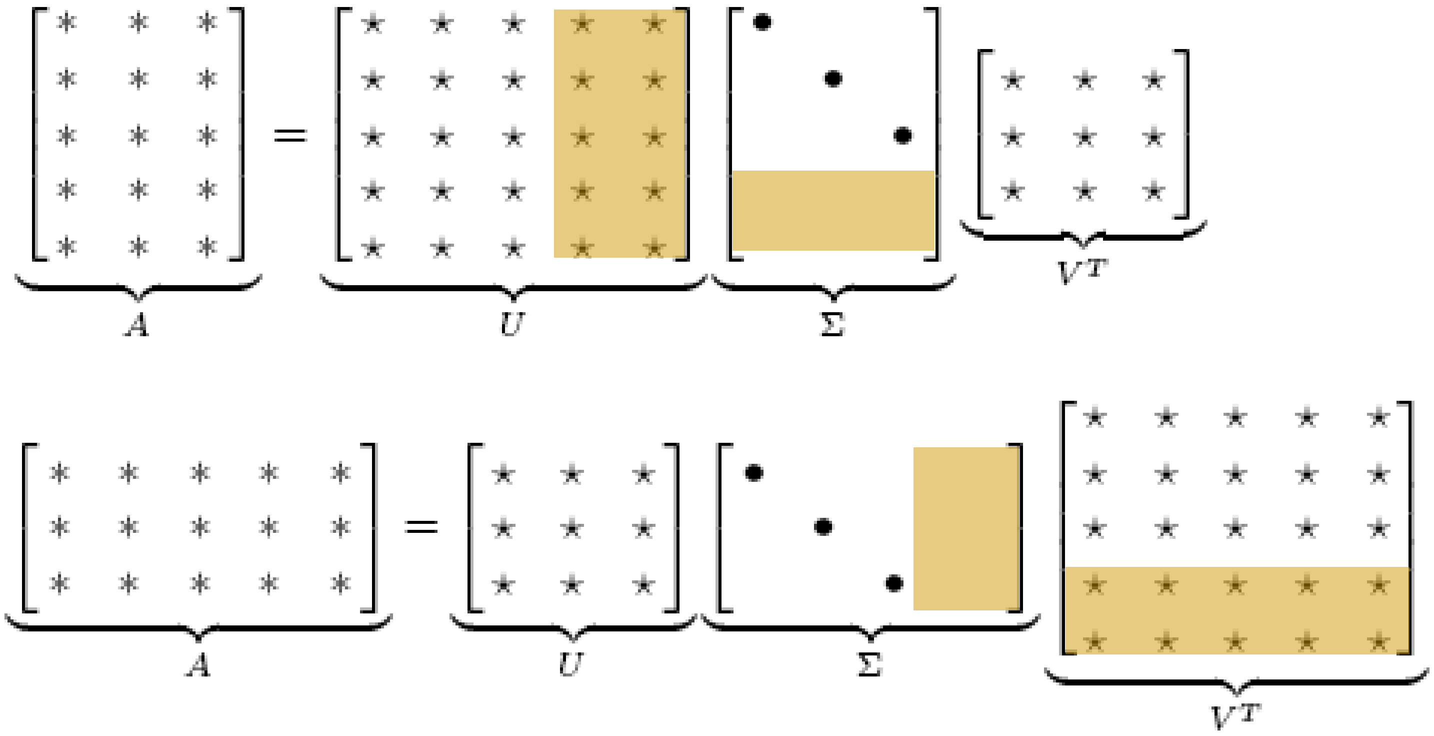
Eigenvalues $\lfloor_1 \ldots \lfloor_r$ of **AA^T** are the eigenvalues of **A^{T}A**.

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

$$\Sigma = diag(\sigma_1 \ldots \sigma_r) \longleftarrow \boxed{\text{Singular values}}$$

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

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

$\longleftarrow$ *Frobenius norm*

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

A_k and X are both m $\times$ n matrices.

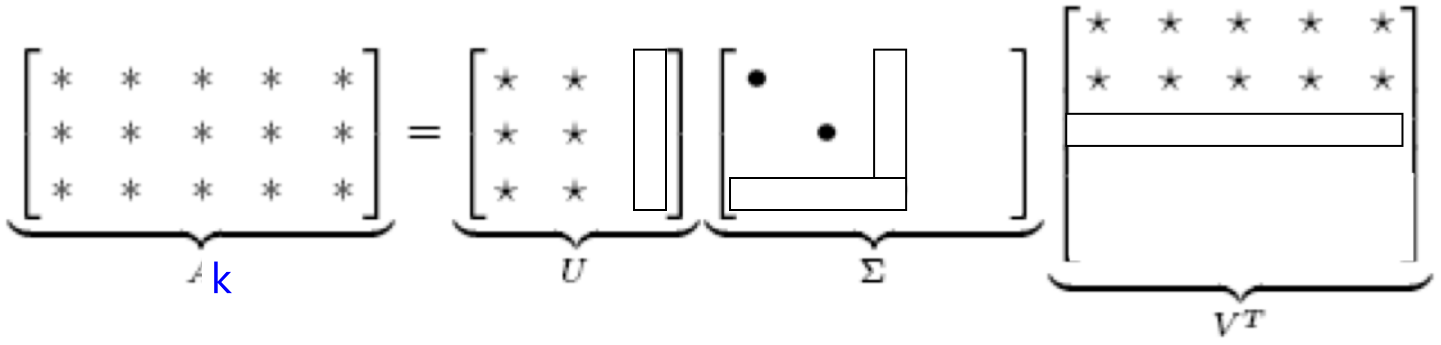
Typically, want k << r.

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

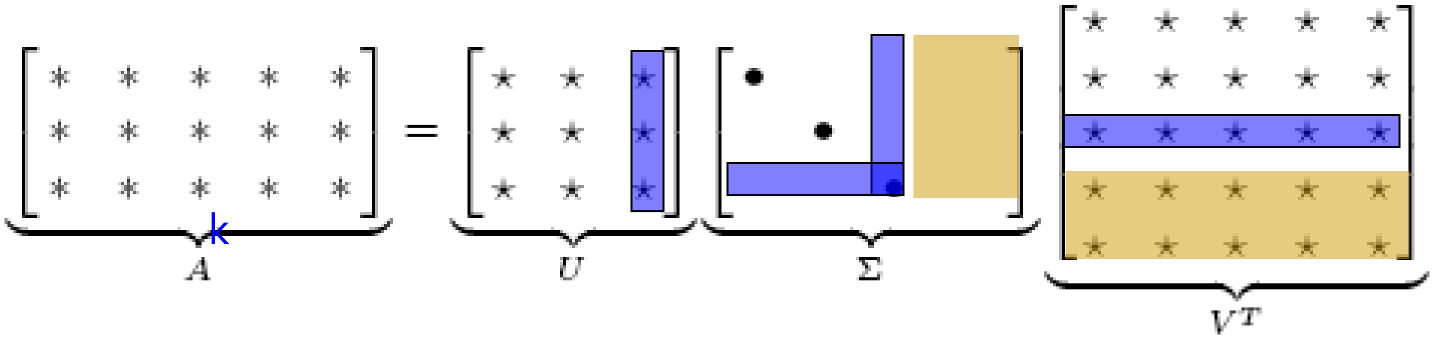


$$A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T$$

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 $k \times k$, U is $M \times k$, V^T is $k \times N$, and A_k is $M \times N$
- This is referred to as the reduced SVD
- It is the convenient (space-saving) and usual form for computational applications
- It's what Matlab gives you



Approximation error

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

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

where the $\int_i$ are ordered such that $\int_i \varepsilon \int_{i+1}$. 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.
  - <u>Of all rank 100 matrices, it would have the lowest Frobenius error.</u>
- Great … but why would we??
- Answer: Latent Semantic Indexing

Latent Semantic Indexing via the SVD

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.

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

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

Example of $C = U\Sigma V^T$: The matrix C

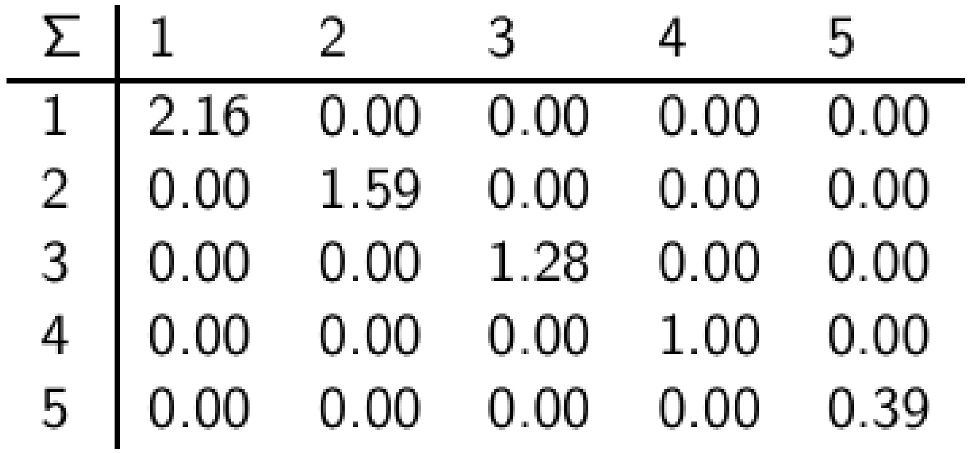
C	d_1	d_2	d_3	d_4	d_5	d_6
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
wood	1	0	0	1	1	0
tree	0	0	0	1	0	1

This is a typical term-document matrix. Actually, we use a non-weighted (binary) matrix here to simplify the example.

Example of $C = U\Sigma V^T$: 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

Example of $C = U\Sigma V^T$: The matrix Σ

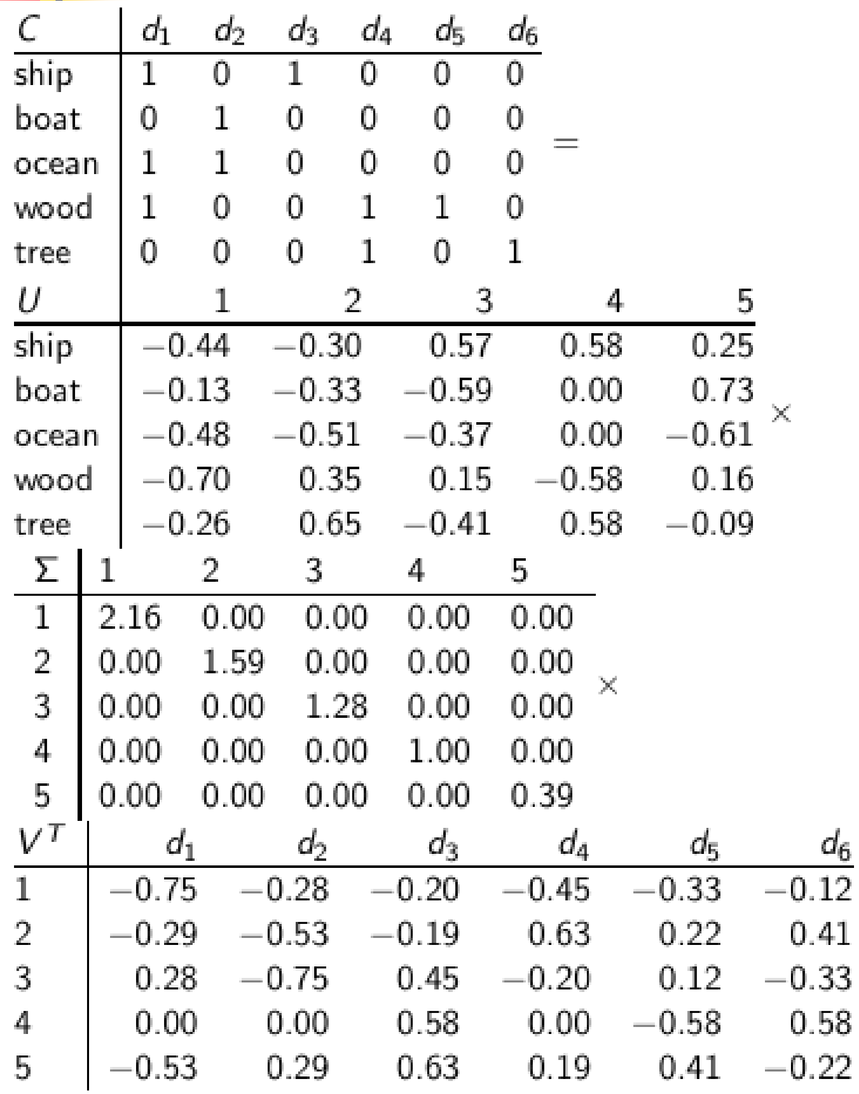


Σ	1	2	3	4	5
1	2.16	0.00	0.00	0.00	0.00
2	0.00	1.59	0.00	0.00	0.00
3	0.00	0.00	1.28	0.00	0.00
4	0.00	0.00	0.00	1.00	0.00
5	0.00	0.00	0.00	0.00	0.39

Example of $C = U\Sigma V^T$: 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

Example of $C = U\Sigma V^T$: All four matrices



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

$=$

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

$\times$

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

$\times$

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

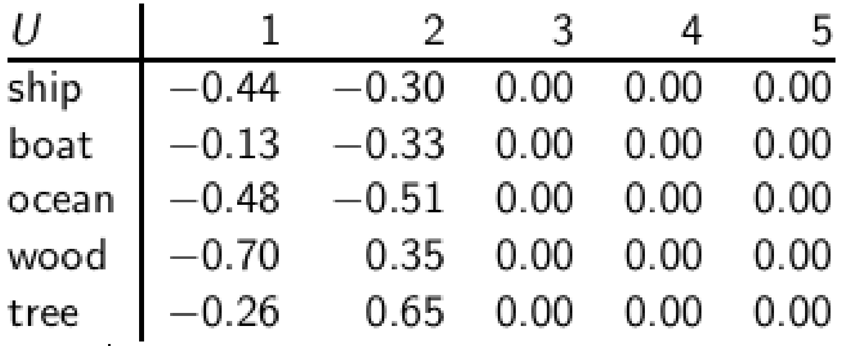
LSI: Summary

- We've decomposed the term-document matrix *C* into a product of three matrices.
- The term matrix *U* – consists of one (row) vector for each term
- The document matrix V^T – consists of one (column) vector for each document
- The singular value matrix Σ – diagonal matrix with singular values, reflecting importance of each dimension
- Next: Why are we doing this?

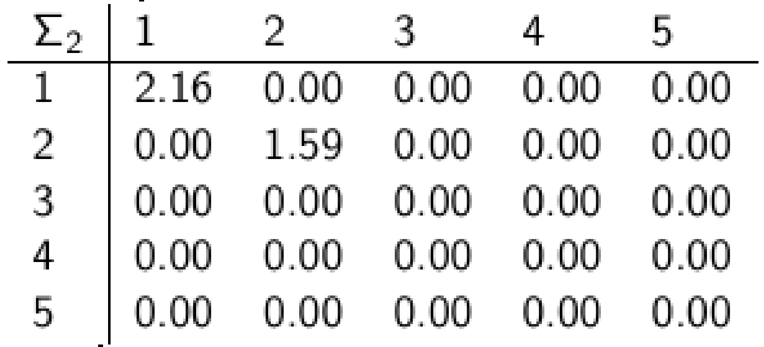
How we use the SVD in LSI

- Key property: Each singular value tells us how important its dimension is.
- By setting less important dimensions to zero, we keep the important information, but get rid of the "details".
- These details may
  - be noise – in that case, reduced LSI is a better representation because it is less noisy
  - make things dissimilar that should be similar – again reduced LSI is a better representation because it represents similarity better.
- Analogy for "fewer details is better"
  - Image of a bright red flower
  - Image of a black and white flower
  - Omitting color makes is easier to see similarity

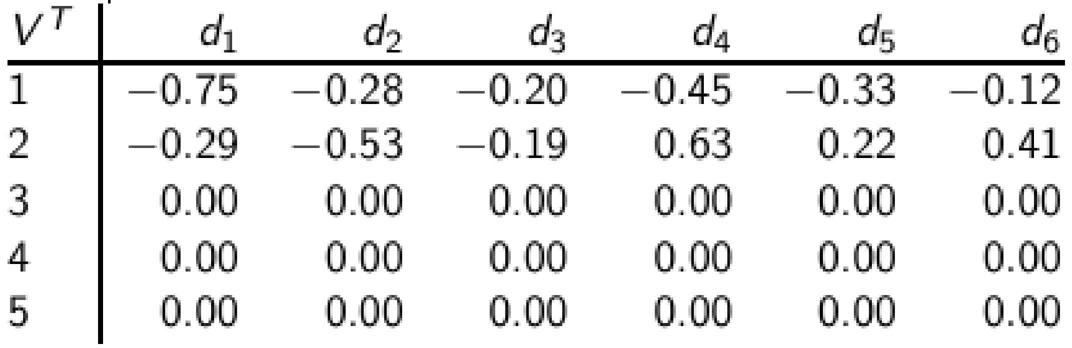
Reducing the dimensionality to 2



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



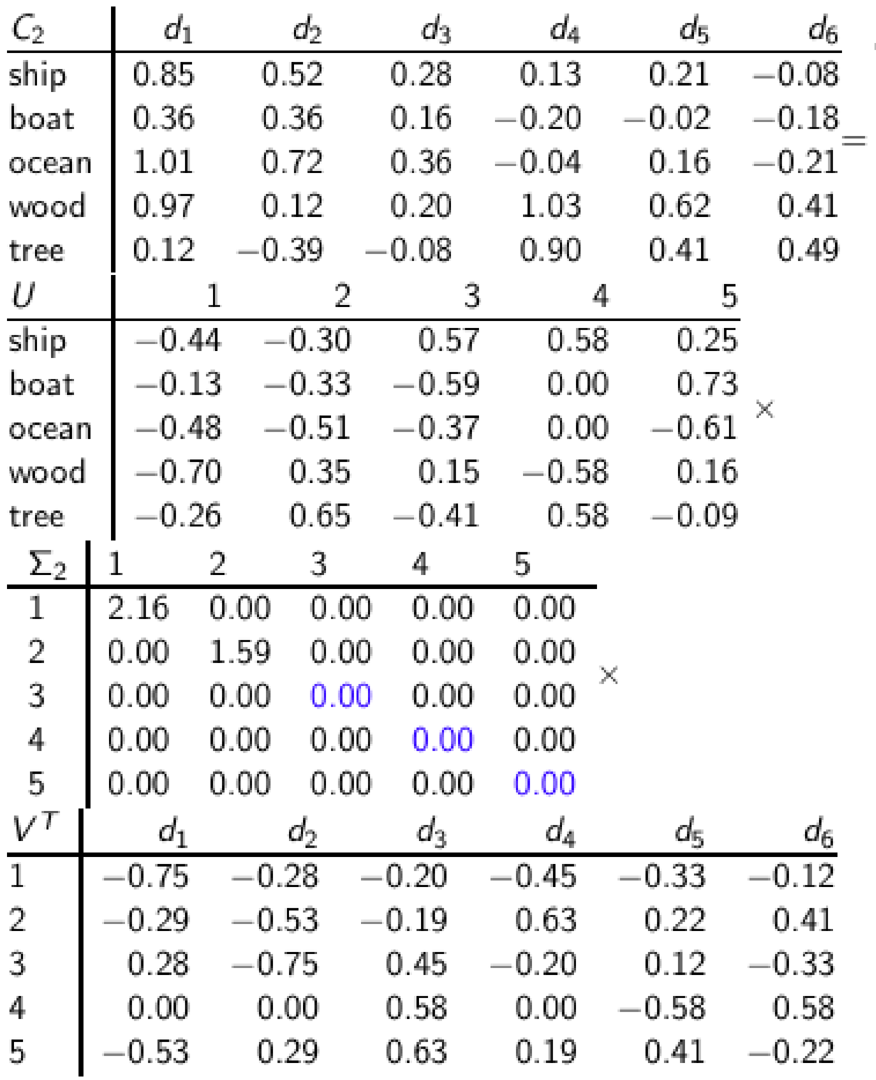
Σ_2	1	2	3	4	5
1	2.16	0.00	0.00	0.00	0.00
2	0.00	1.59	0.00	0.00	0.00
3	0.00	0.00	0.00	0.00	0.00
4	0.00	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	0.00



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

Actually, we only zero out singular values in Σ. This has the effect of setting the corresponding dimensions in U and V^T to zero when computing the product $C = U\Sigma V^T$.

Reducing the dimensionality to 2



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

$=$

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

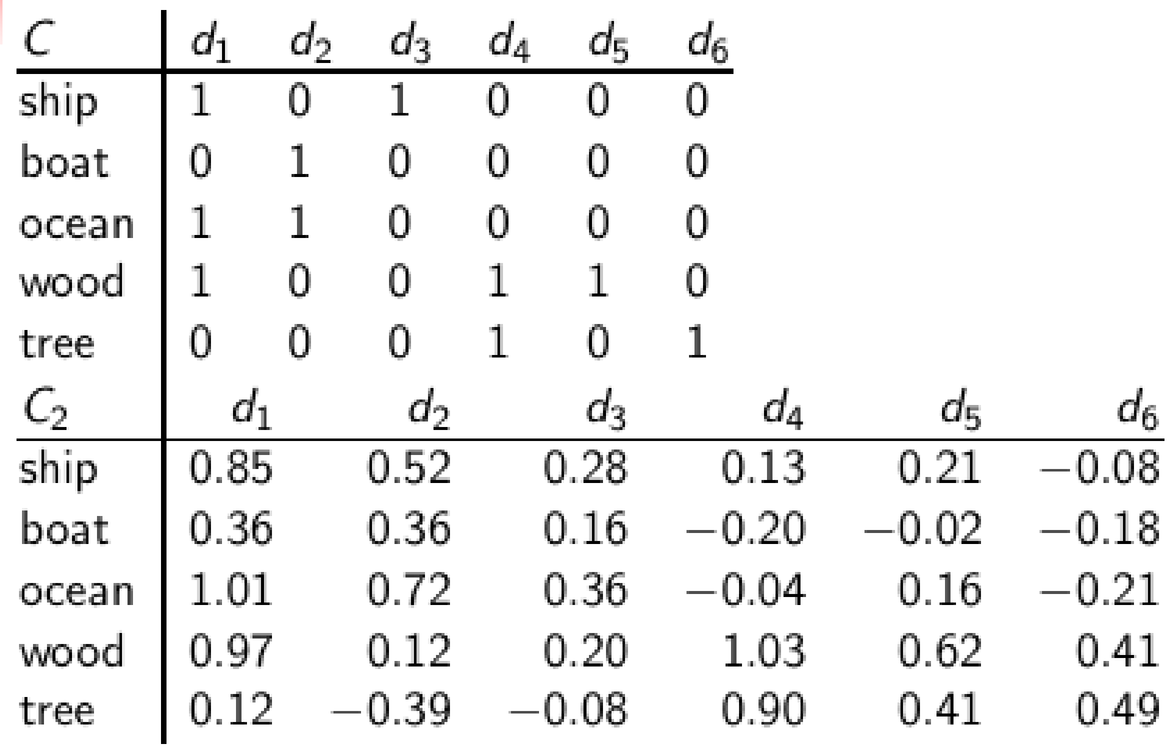
$\times$

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

$\times$

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

Original matrix C vs. reduced $C_2 = U\Sigma_2 V^T$



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

We can view C_2 as a two-dimensional representation of the matrix. We have performed a dimensionality reduction to two dimensions.

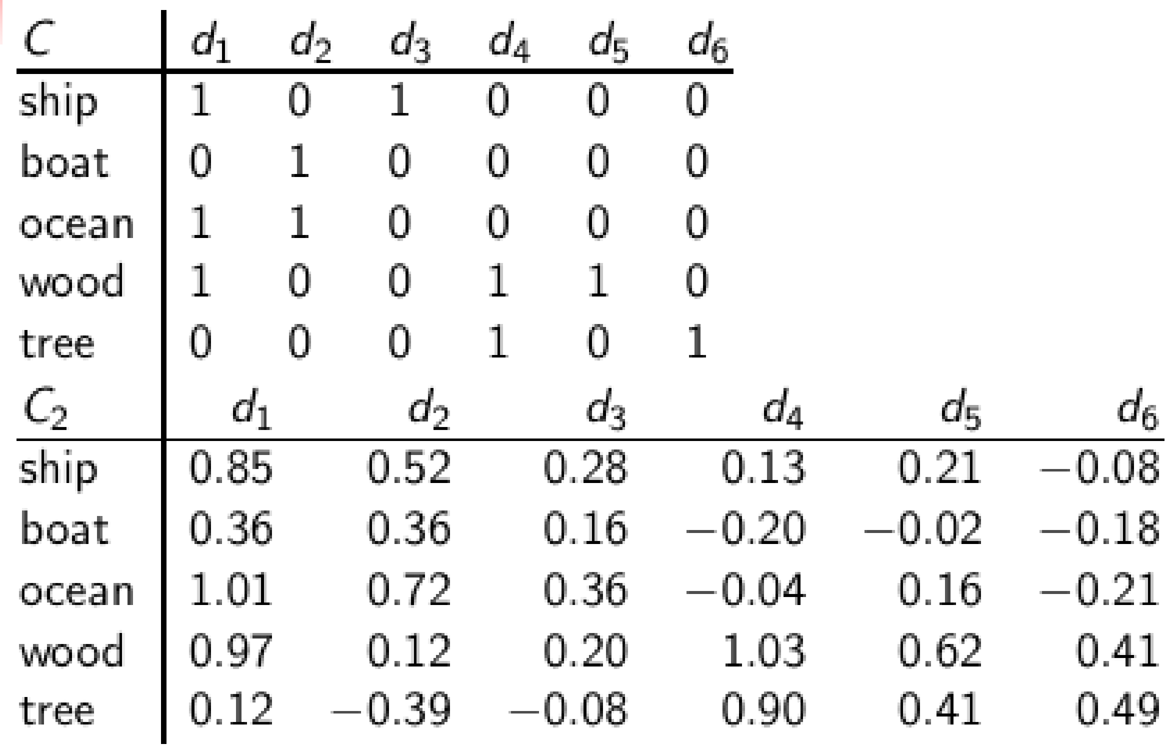
Why the reduced matrix is “better”

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

Similarity of d2 and d3 in the original space: 0.
Similarity of d2 und 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 ≈ 0.52

Why the reduced matrix is "better"



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

"boat" and "ship" are semantically similar. The "reduced" similarity measure reflects this.

Why we use LSI in information retrieval

- 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.
- Thus, LSI addresses the problems of synonymy and semantic relatedness.
- Standard vector space: Synonyms contribute nothing to document similarity.
- Desired effect of LSI: Synonyms contribute strongly to document similarity.

How LSI addresses synonymy and semantic relatedness

- The dimensionality reduction forces us to omit a lot of "detail".
- We have to map differents words (= different dimensions of the full space) to the same dimension in the reduced space.
- The "cost" of mapping synonyms to the same dimension is much less than the cost of collapsing unrelated words.
- SVD selects the "least costly" mapping (see below).
- Thus, it will map synonyms to the same dimension.
- But it will avoid doing that for unrelated words.

Implementation

- Compute SVD of term-document matrix
- Reduce the space and compute reduced document representations
- Map the query into the reduced space

$$\vec{q}_2^T = \Sigma_2^{-1} U_2^T \vec{q}^T.$$

- This follows from:

$$C_2 = U\Sigma_2 V^T \Rightarrow \Sigma_2^{-1} U^T C = V_2^T$$

- Compute similarity of q_2 with all reduced documents in V_2.
- Output ranked list of documents as usual
- Exercise: What is the fundamental problem with this approach?

Resources

- Chapter 18 of IIR
- Resources at http://ifnlp.org/ir
  - Original paper on latent semantic indexing by Deerwester et al.
  - Paper on probabilistic LSI by Thomas Hofmann
  - Word space: LSI for words