Linear Algebra for AI/ML

Jiaul Paik

Latent Semantic Analysis of Text with SVD

- 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?

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

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

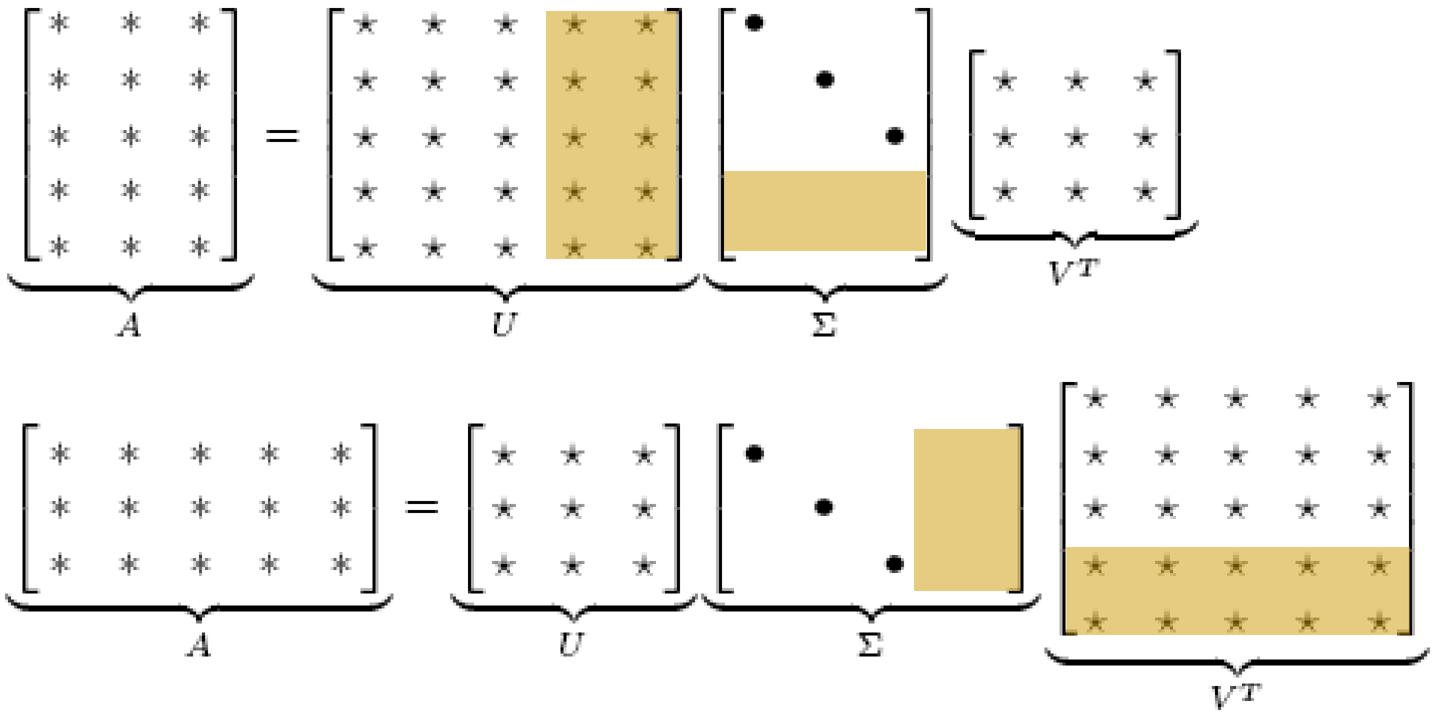
M·M

M·N

V is N·N

Singular Value Decomposition

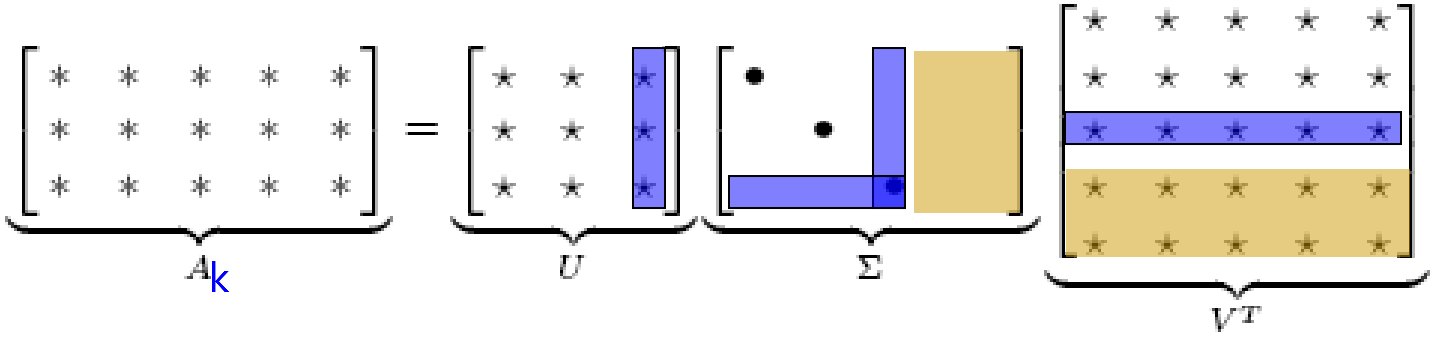
- **Illustration of SVD dimensions and sparseness**



Low-rank Approximation

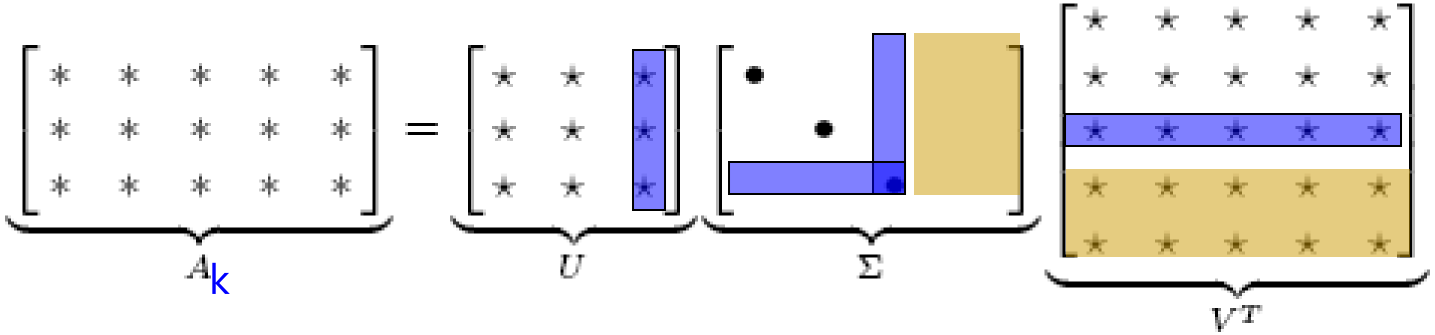
- **Solution via SVD**

set smallest r-k singular values to zero



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 usual form for computational applications



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?**

Text Mining/NLP: Vector Space Model

- **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

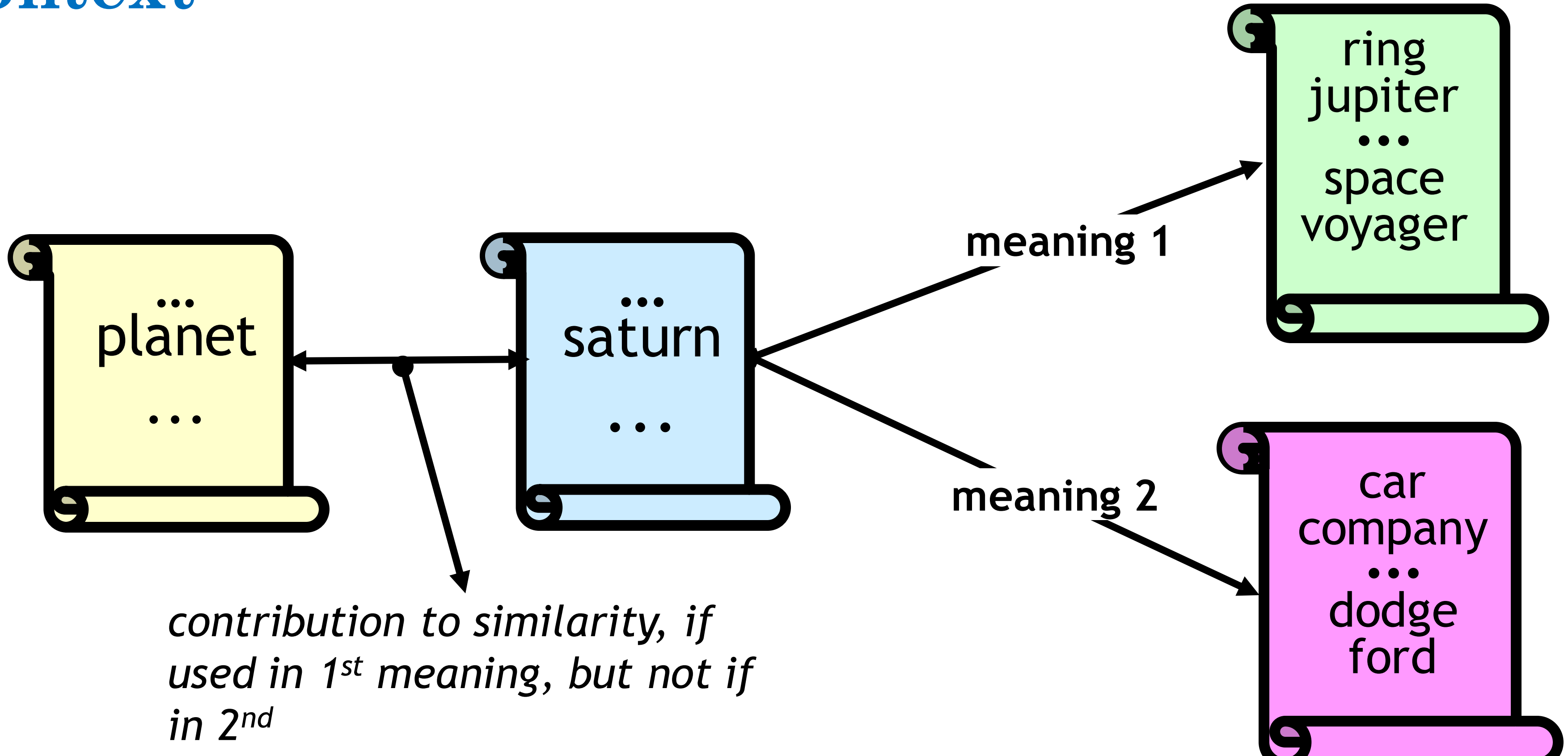
- **Synonymy: Different terms may have an identical or a similar meaning (weaker: words indicating the same topic).**

- **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**



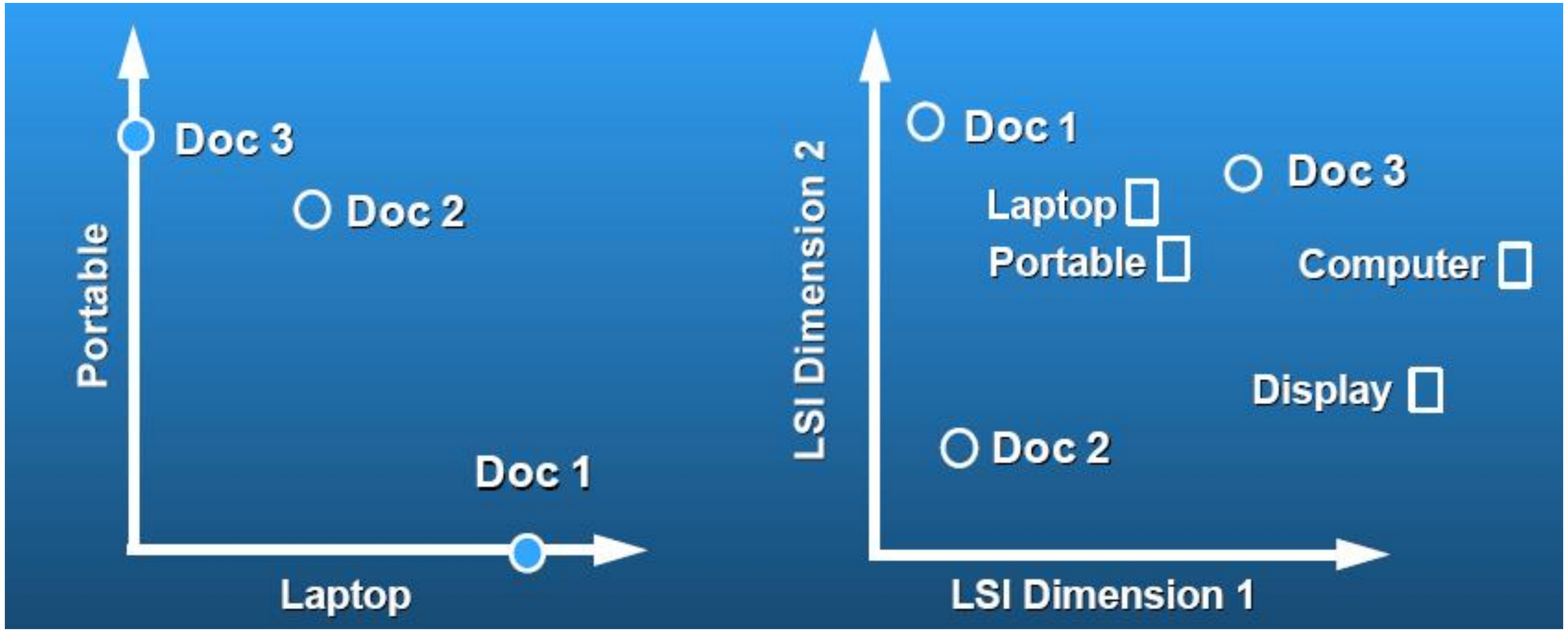
Latent Semantic Analysis (LSA)

- **Perform a low-rank approximation of document-term matrix**

- **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**

Latent Semantic Analysis

- **Latent semantic space: illustrating example**



courtesy of Susan Dumais

LSA Example

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 = UΣV^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

LSA Example

- **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

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. $C_2 = U\Sigma_2 V^T$

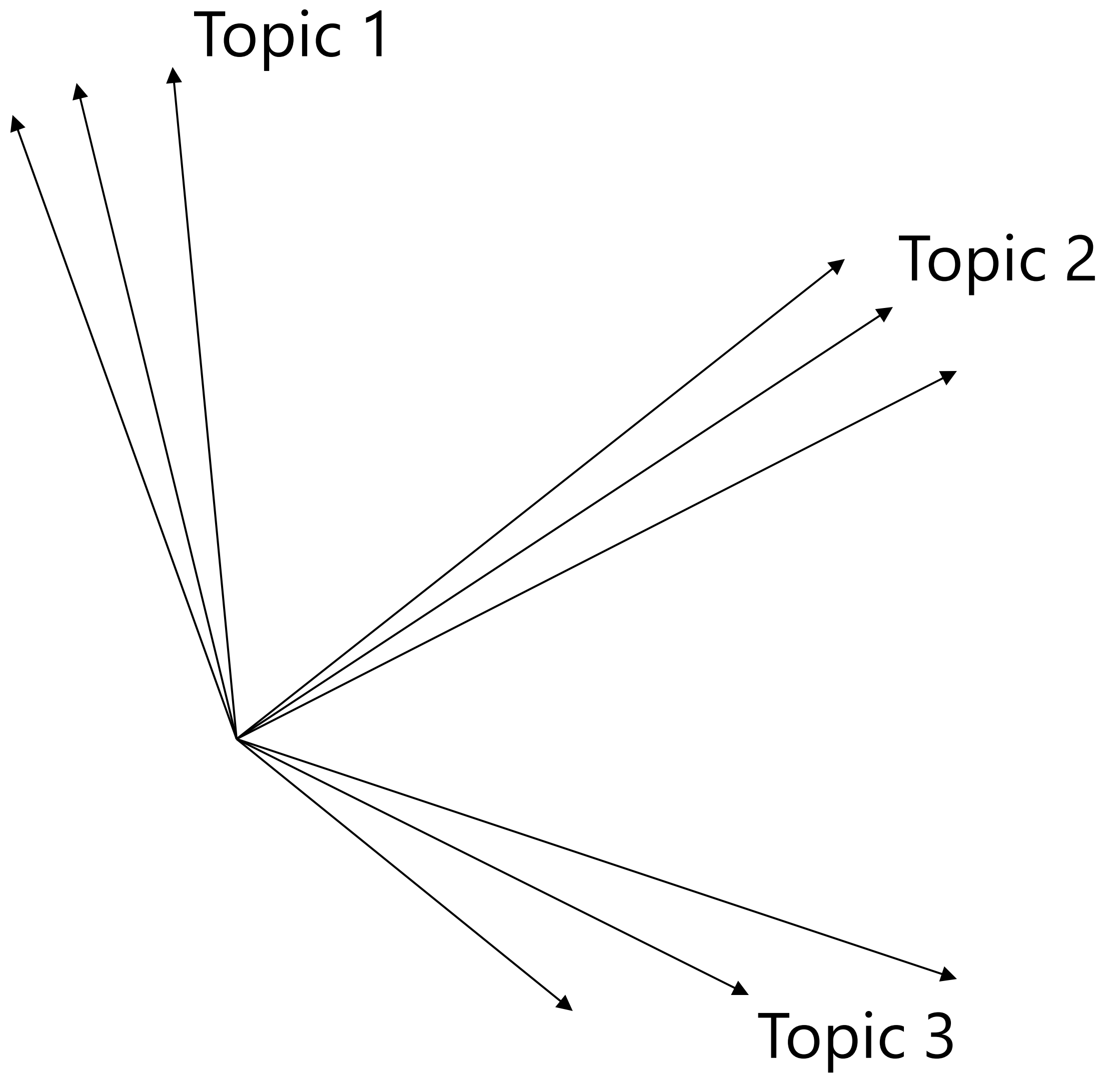
C	d_1	d_2	d_3	d_4	d_5	d_6
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
wood	1	0	0	1	1	0
tree	0	0	0	1	0	1

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

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:**
  - $0.52 * 0.28 + 0.36 * 0.16 + 0.72 * 0.36 + 0.12 * 0.20 + -0.39 * -0.08 \approx 0.52$

Simplistic picture



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.

Thank you!