Information Retrieval and Web Search Engines

Lecture 5: Latent Semantic Indexing

Wolf-Tilo Balke
Muhammad Usman

Institut für Informationssysteme
Technische Universität Braunschweig

Independence

- Many information retrieval models assume **independent** (orthogonal) terms
- This is problematic (synonyms, …)
- What can we do?
  Use **independent "topics"** instead of terms!
- What do we need?
  - How to relate **single terms** to topics?
  - How to relate **documents** to topics?
  - How to relate **query terms** to topics?

Dealing with Topics

- Naïve approach:
  1. Find a **librarian** who knows the subject area of your document collection well enough
  2. Let him/her **identify independent topics**
  3. Let him/her **assign documents to topics**
     - A document about sports gets a weight of −1.1 with respect to the topic “politics”
     - A document about the vector space model gets a weight of 2.7 with respect to the topic “information retrieval”
  4. Find a method to **transform queries** over terms into queries over topics (e.g. by exploiting term/topic assignments provided by the librarian)

The difficult part…

Can it be automated?

The easy part…

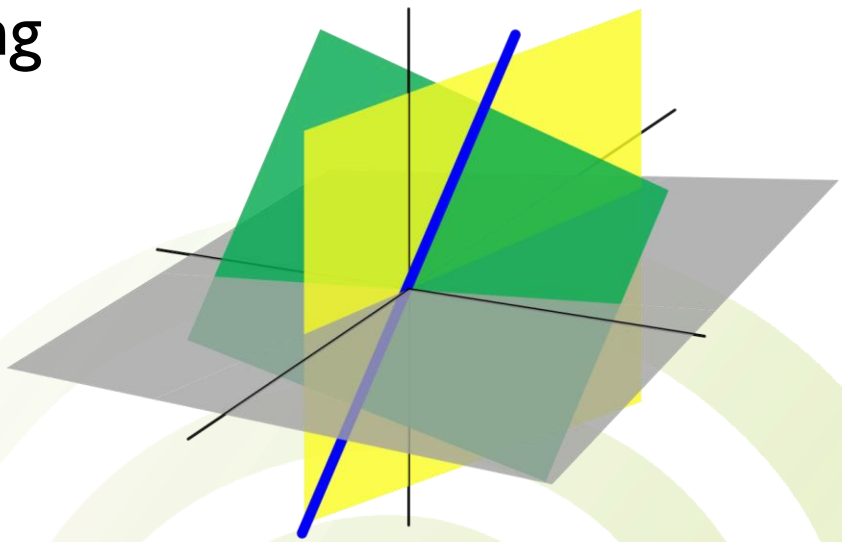
Latent Semantic Indexing

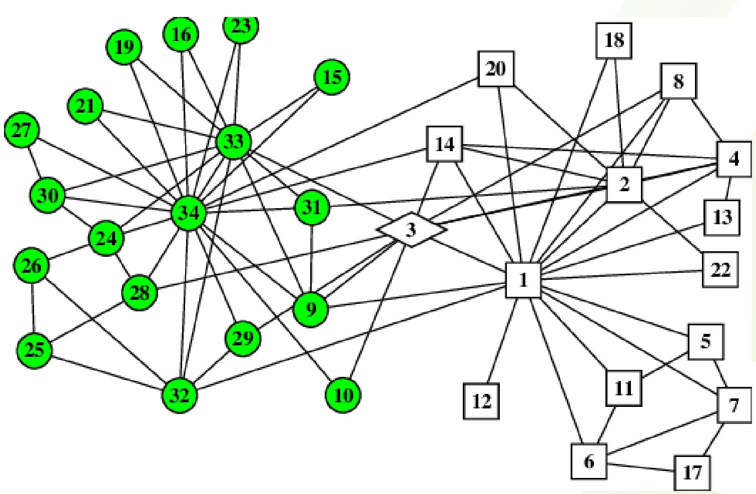
- **Latent Semantic Indexing** does the trick
- Proposed by Dumais *et al.* (1988)
- **Patented** in 1988 (US Patent 4,839,853)

- **Central idea:**
  Represent each document
  within a **"latent space of topics"**
- Use **singular value decomposition** (SVD)
  to derive the structure of this space
- The SVD is an important result from **linear algebra**

Latend Semantic Indexing

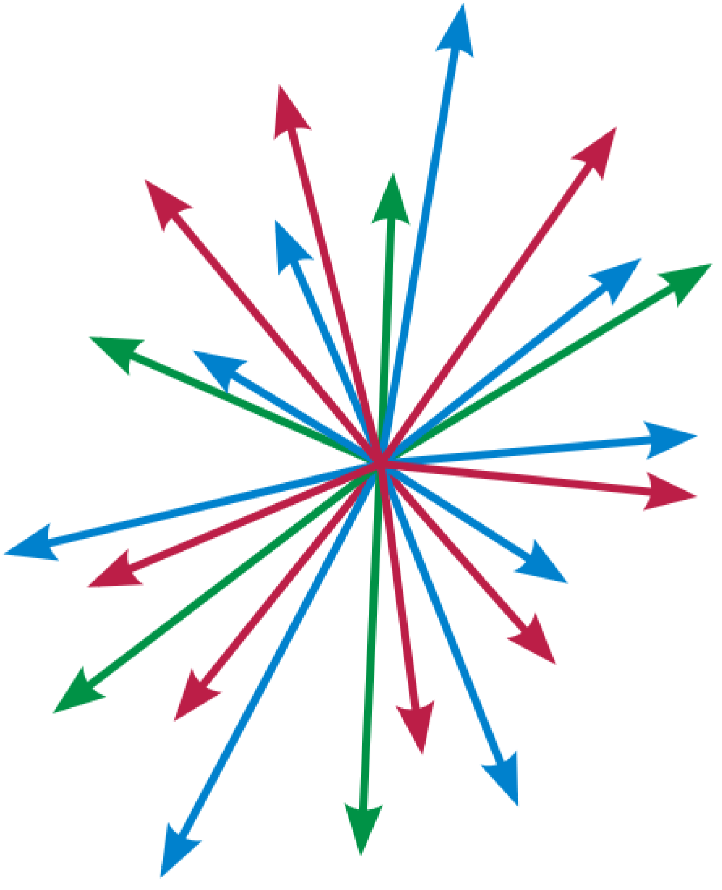
1. **Recap of Linear Algebra**
2. Singular Value Decomposition
3. Latent Semantic Indexing





Linear Algebra

- Linear algebra is the branch of mathematics concerned with the study of:
  - systems of linear equations,
  - **vectors,**
  - vector spaces, and
  - linear transformations (represented by **matrices**).



- **Important tool** in…
  - Information retrieval
  - Data compression
  - …

Vectors

- **Vectors** represent points in space
- There are:
  - Row vectors:

$$x = (x_1, x_2, x_3)$$

  - Column vectors:

Transpose

$$y = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = (y_1, y_2, y_3)^T$$

- All vectors (and matrices) considered in this course will be **real-valued**

Matrices

- Every $(m \times n)$-matrix A defines a **linear map** from $\mathbb{R}^n$ to $\mathbb{R}^m$ by sending the column vector $x \in \mathbb{R}^n$ to the column vector $Ax \in \mathbb{R}^m$:

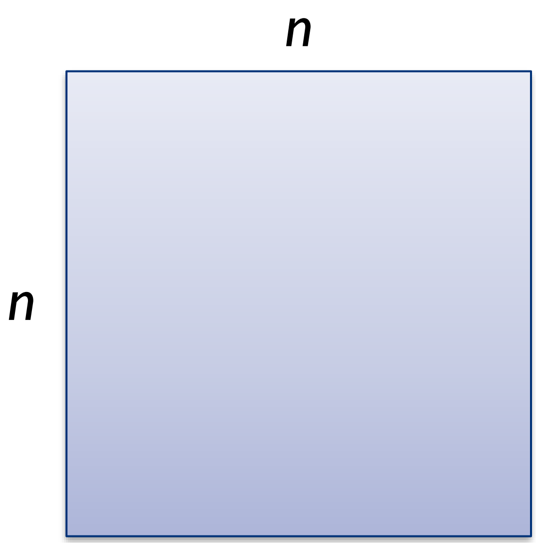
$$x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$$

$$A = (a_{i,j}) = \begin{pmatrix} a_{1,1} & \cdots & a_{1,n} \\ \vdots & \ddots & \vdots \\ a_{m,1} & \cdots & a_{m,n} \end{pmatrix} \qquad \begin{pmatrix} \sum_{j=1}^{n} a_{1,j} x_j \\ \vdots \\ \sum_{j=1}^{n} a_{m,j} x_j \end{pmatrix} = Ax$$

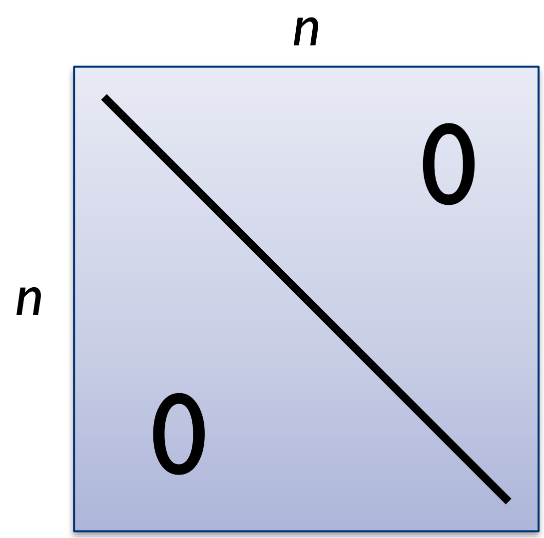
Row *i*

Column *j*

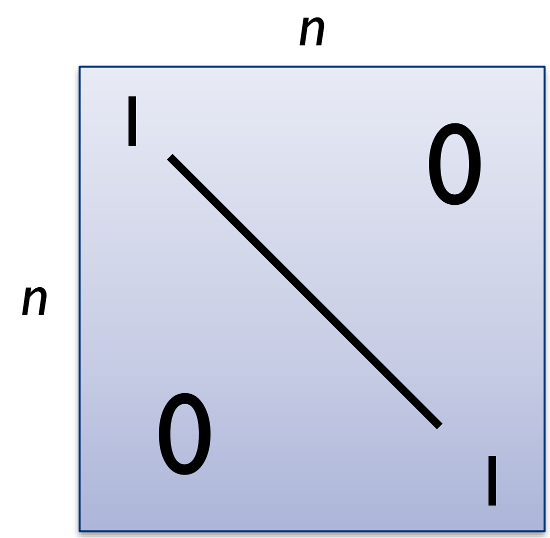
Matrix Gallery



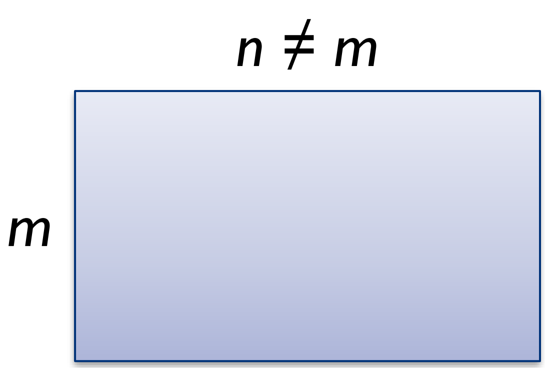
Square matrix



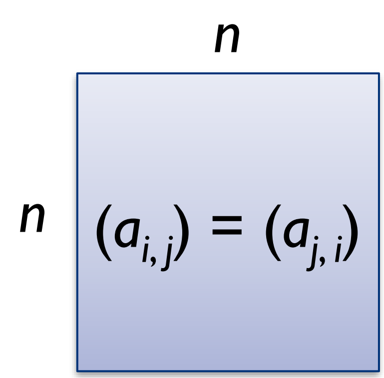
Diagonal matrix



Identity matrix



Rectangular matrix



Symmetric matrix

Linear Independence

- A set $\{x^{(1)}, \ldots, x^{(k)}\}$ of n-dimensional vectors is **linearly dependent** if there are real numbers $\lambda_1, \ldots, \lambda_k$, not all zero, such that

$$\lambda_1 \mathbf{x}^{(1)} + \cdots + \lambda_k \mathbf{x}^{(k)} = \mathbf{0}$$

Null vector

- Otherwise, this set is called **linearly independent**
- **Theorem:**
  If $k > n$, the set is linearly dependent

Linear Span

- Let $\{x^{(1)}, \ldots, x^{(k)}\}$ be a set of n-dimensional vectors
- The **linear span** (aka linear hull) of this set is defined as:

$$\operatorname{span}\left\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(k)}\right\} = \left\{\underbrace{\lambda_1 \mathbf{x}^{(1)} + \cdots + \lambda_k \mathbf{x}^{(k)}}_{\text{Linear combination}} \;\middle|\; \lambda_1, \ldots, \lambda_k \in \mathbb{R}\right\}$$

- **Idea:**
  The linear span is the set of all points in $\mathbb{R}^n$ that can be expressed by **linear combinations** of $x^{(1)}, \ldots, x^{(k)}$
- The linear span is a **subspace** of $\mathbb{R}^n$ with **dimension** at most k

Linear Span (2)

- The span of $\{x^{(1)}, \ldots, x^{(k)}\}$ can be:
  - A single point (0-dimensional)
  - A line (1-dimensional)
  - A plane (2-dimensional)
  - …

- **Example:**
  span$\{(1, 2, 3), (2, 4, 6), (3, 6, 9)\}$ is a line in $\mathbb{R}^3$

- **Example:**
  span$\{(1, 0, 0), (0, 1, 0), (0, 0, 1)\} = \mathbb{R}^3$

Basis

- Let $\{x^{(1)}, \ldots, x^{(k)}\}$ be a set of **linearly independent** *n*-dimensional vectors

- **Theorem:**
  span$\{x^{(1)}, \ldots, x^{(k)}\}$ is a ***k*-dimensional** subspace of $\mathbb{R}^n$

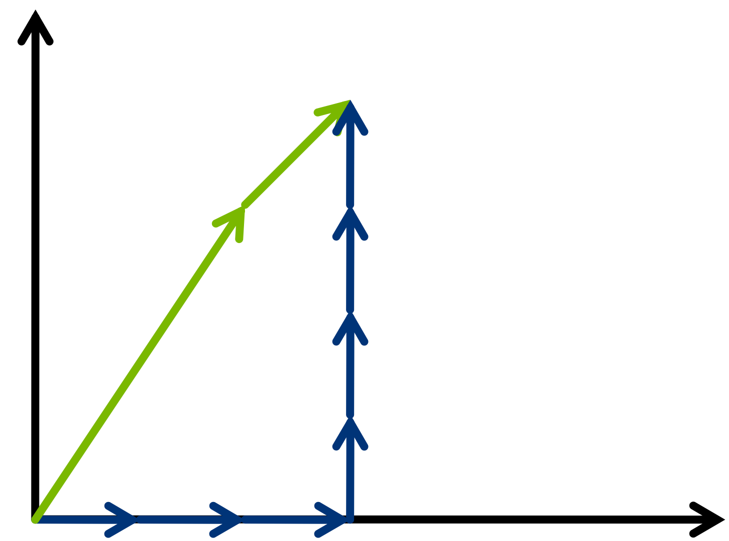
- **Theorem:**
  Any point in span$\{x^{(1)}, \ldots, x^{(k)}\}$ is generated by a **unique linear combination** of $x^{(1)}, \ldots, x^{(k)}$

- $\{x^{(1)}, \ldots, x^{(k)}\}$ is called a **basis** of the subset it spans

Example

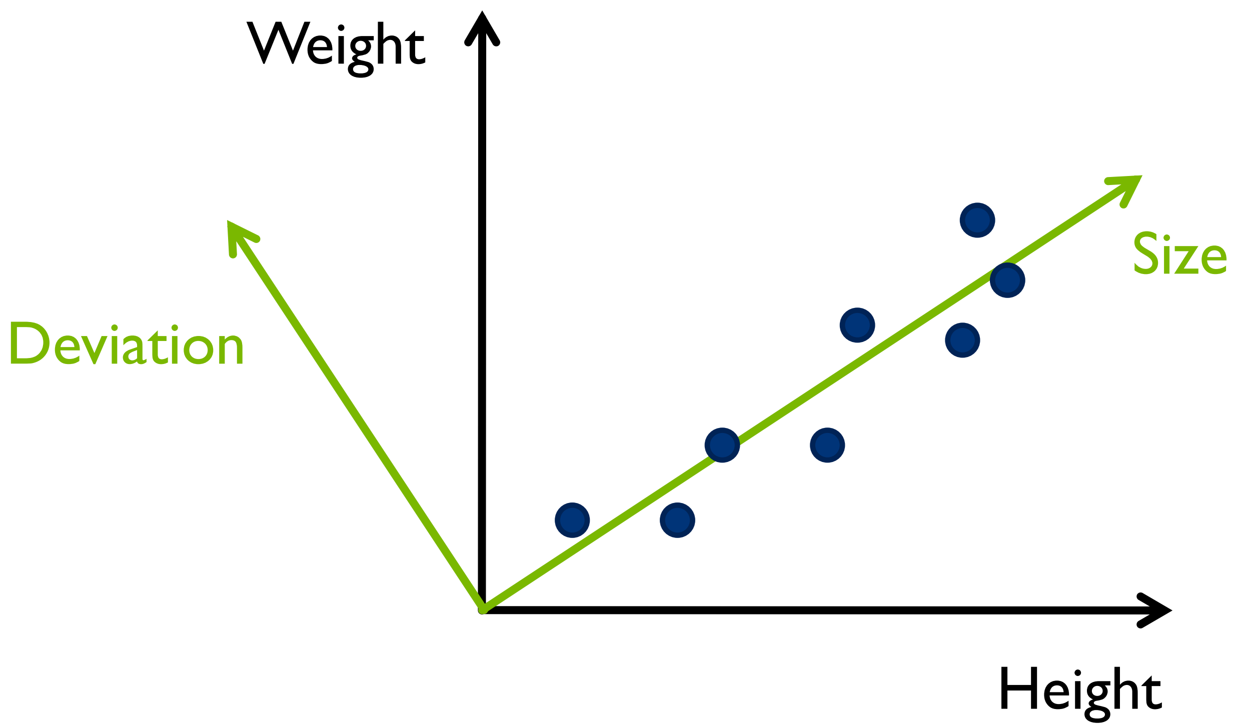
- Two bases of $\mathbb{R}^2$:
  - $B_1 = \{(1, 0), (0, 1)\}$ (standard basis)
  - $B_2 = \{(1, 1), (2, 3)\}$

- What are the coordinates of standard basis' point (3, 4) with respect to basis B_2?
  - B_1: $3 \cdot (1, 0) + 4 \cdot (0, 1) = (3, 4)$
  - B_2: $1 \cdot (1, 1) + 1 \cdot (2, 3) = (3, 4)$



Non-Standard Bases

- Often it is useful to represent data using a non-standard basis:



Change of Basis

- Let $B_1 = \{x^{(1)}, \ldots, x^{(k)}\}$ and $B_2 = \{y^{(1)}, \ldots, y^{(k)}\}$
  be **two bases of the same subspace** $V \subseteq \mathbb{R}^n$,
  i.e., span $B_1 = V =$ span B_2

- **Theorem:**
  There is a unique **transformation matrix** T such that
  $Tx^{(i)} = y^{(i)}$, for any $i = 1, \ldots, k$

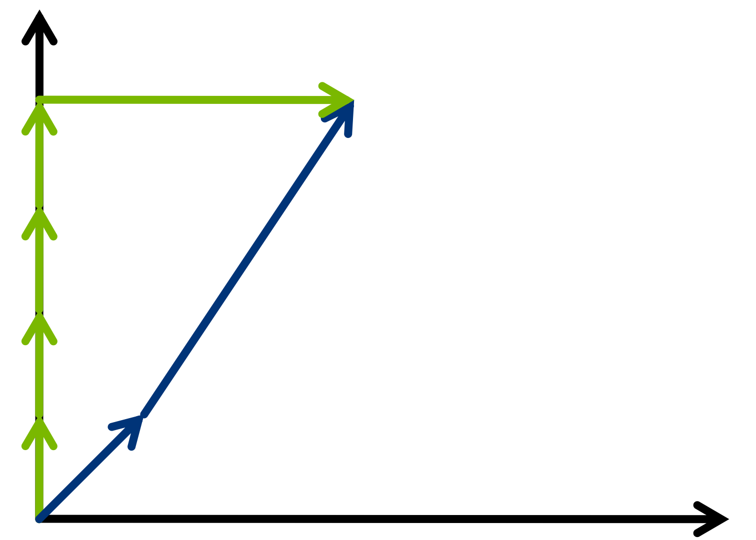
- T can be used to transform the coordinates of points given with respect to base B_1 into the corresponding coordinates with respect to base B_2

Example

- Two bases of $\mathbb{R}^2$:
  - $B_1 = \{(1, 1), (2, 3)\}$
  - $B_2 = \{(0, 1), (3, 0)\}$
- Given a point p with coordinates (1, 1) wrt. base B_1
- What are p's coordinates wrt. base B_2?

$$T = \begin{pmatrix} 1 & 3 \\ \frac{1}{3} & \frac{2}{3} \end{pmatrix}$$

- $T \cdot (1, 1)^T = (4, 1)$

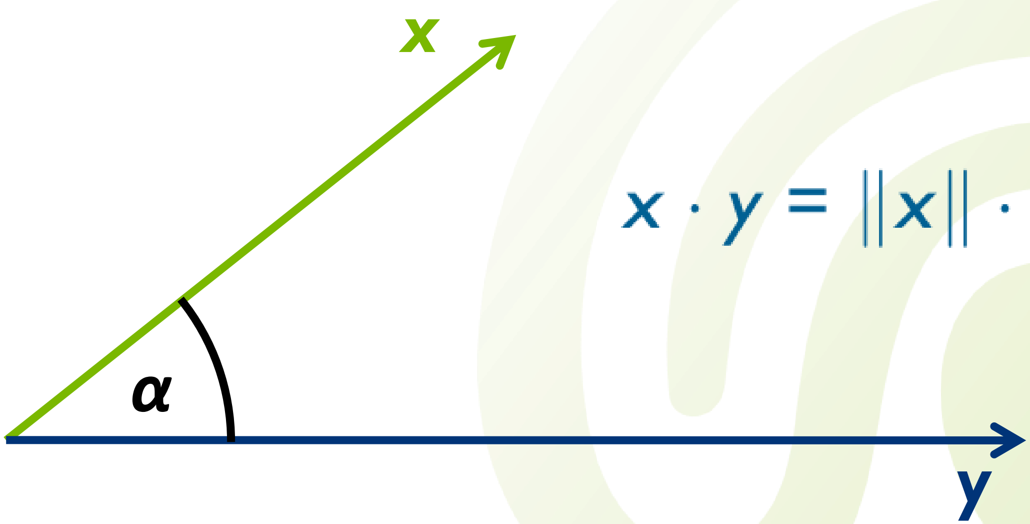


Orthogonality

- **Scalar product** (aka dot product) of vectors $x, y \in \mathbb{R}^n$ and **length** (norm) of a vector $x \in \mathbb{R}^n$:

$$x \cdot y = \sum_{i=1}^{n} x_i y_i \qquad \|x\| = \sqrt{\sum_{i=1}^{n} x_i^2} = \sqrt{x \cdot x}$$

- Two vectors $x, y \in \mathbb{R}^n$ are **orthogonal** if $x \cdot y = 0$



$$x \cdot y = \|x\| \cdot \|y\| \cdot \cos(\alpha)$$

Orthonormality

- **Theorem:**
  Any set of mutually orthogonal vectors
  is linearly independent

- A set of *n*-dimensional vectors is **orthonormal**
  if all vectors are of length 1 and are mutually orthogonal

- A matrix is **column-orthonormal**
  if its set of column vectors is orthonormal
  (row-orthonormality is defined analogously)

Rank of a Matrix

- The **rank** of a matrix is the number of linearly independent rows in it (or columns; it's the same)
- The rank of a matrix A can also be defined as the **dimension of the image** of the linear map $f(x) = Ax$
- **Theorem:**
  The rank of a diagonal matrix is
  equal to the number of its nonzero diagonal entries

Example

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

is row- and column-orthonormal;
its rank is 4

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

is row-orthonormal;
its rank is 3

Eigenvectors and Eigenvalues

- Let A be a square $(n \times n)$-matrix
- Let $x \in \mathbb{R}^n$ be a non-zero vector
- x is an **eigenvector** of A
  if it satisfies the equation
  $Ax = \lambda x$, for some real number λ
- Then, λ is called an **eigenvalue** of A corresponding to the eigenvector x

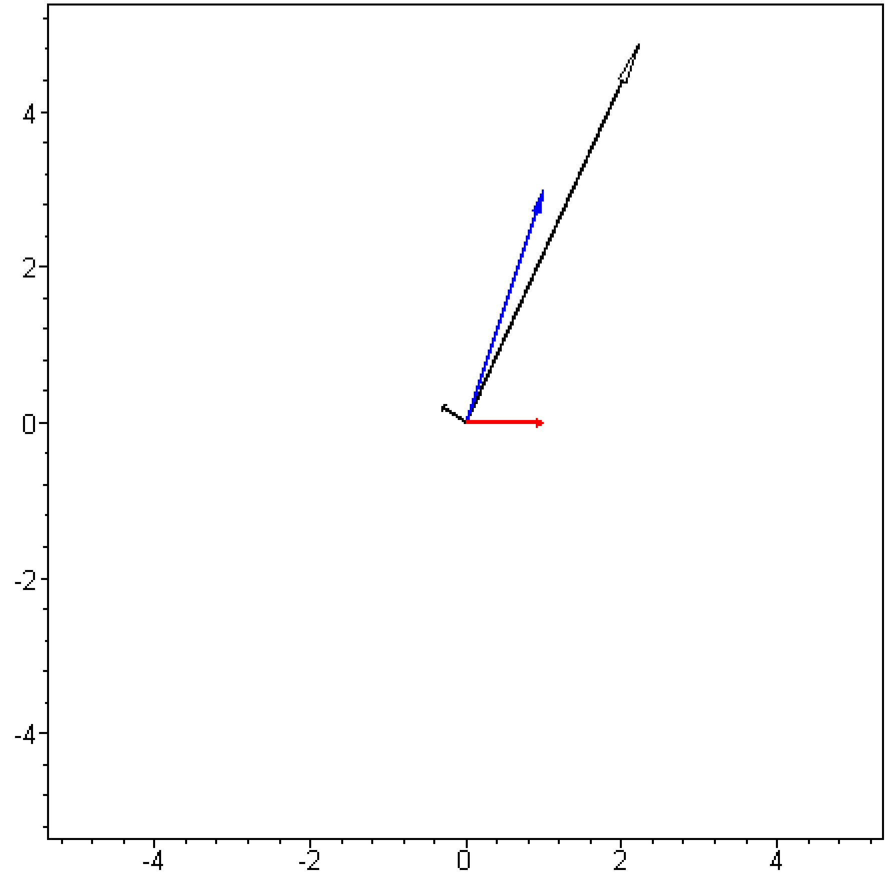
Manfred Eigen

- **Idea:**
  - Eigenvectors are **mapped to itself** (possibly scaled)
  - Eigenvalues are the corresponding **scaling factors**

Example

- Unit vector *x*
- Vector *Ax* (image of *x*)
- Eigenvectors multiplied by eigenvalues

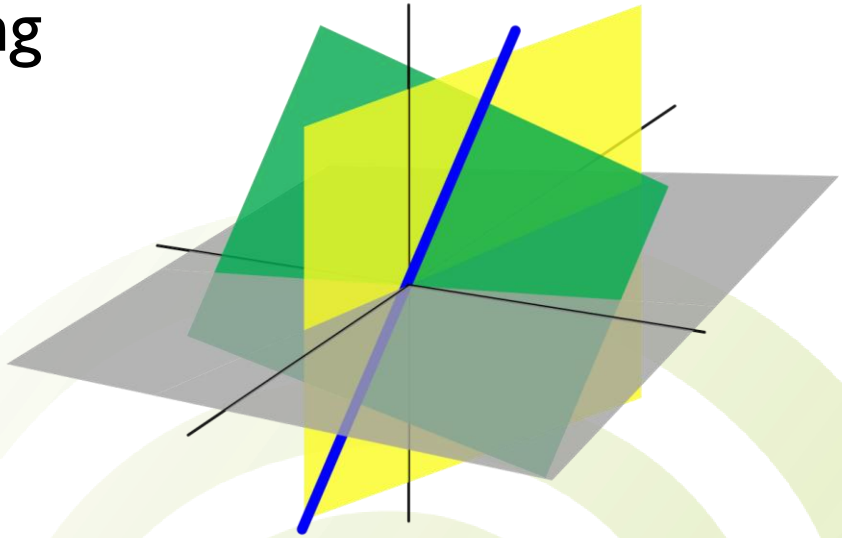
- It could be useful to change the basis to the set of eigenvectors…

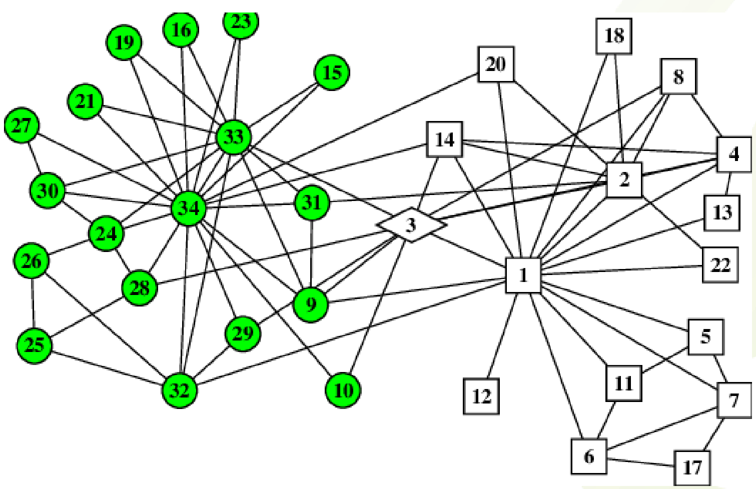


Source: http://centaur.maths.qmul.ac.uk/Lin_Alg_I

Latend Semantic Indexing

1. Recap of Linear Algebra
2. **Singular Value Decomposition**
3. Latent Semantic Indexing

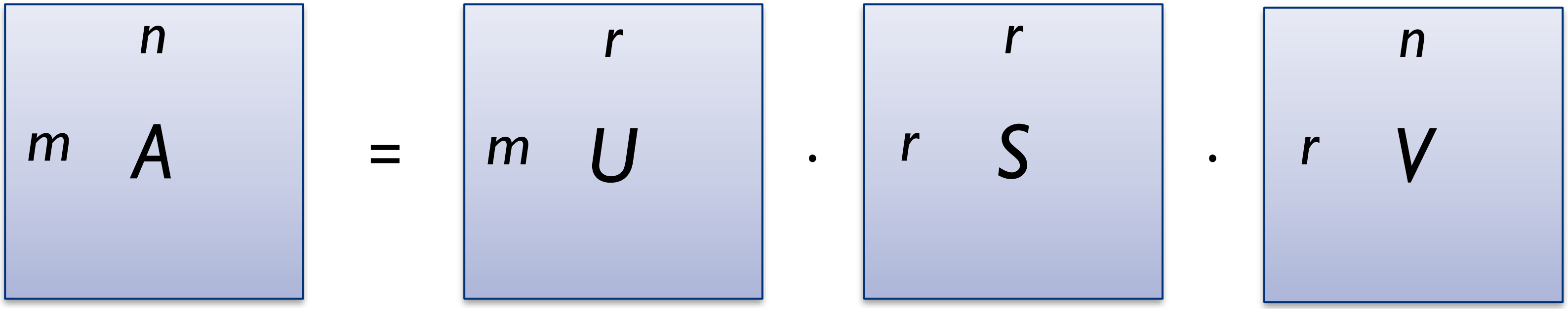




Singular Value Decomposition

- Let A be an $(m \times n)$-matrix (rectangular!)
- Let r be the rank of A
- **Theorem:**
  A can be decomposed such that $A = U \cdot S \cdot V$, where
  - U is a column-orthonormal $(m \times r)$-matrix
  - V is a row-orthonormal $(r \times n)$-matrix
  - S is a diagonal matrix such that $S = \text{diag}(s_1, \ldots, s_r)$ and $s_1 \geq s_2 \geq \cdots \geq s_r > 0$
- The columns of U are called **left singular vectors**
- The rows of V are called **right singular vectors**
- s_i is referred to as A's i-th **singular value**

Singular Value Decomposition



column-orthonormal, left singular vectors, rank r

diagonal, singular values, rank r

row-orthonormal, right singular vectors, rank r

- The linear map A can be split into three mapping steps:
  - Given $x \in \mathbb{R}^n$, it is $Ax = USVx$
    - V maps x into space $\mathbb{R}^r$,
    - S scales the components of Vx
    - U maps SVx into space $\mathbb{R}^m$
  - The same holds for $y \in \mathbb{R}^m$; it is $yA = yUSV$

Example

- We measured the height and weight of several persons:

	Person 1	Person 2	Person 3	Person 4	Person 5
Height	170cm	175cm	182cm	183cm	190cm
Weight	69kg	77kg	77kg	85kg	89kg

- Compute the SVD of this data matrix:

$$\begin{pmatrix} 170 & 175 & 182 & 183 & 190 \\ 69 & 77 & 77 & 85 & 89 \end{pmatrix}$$

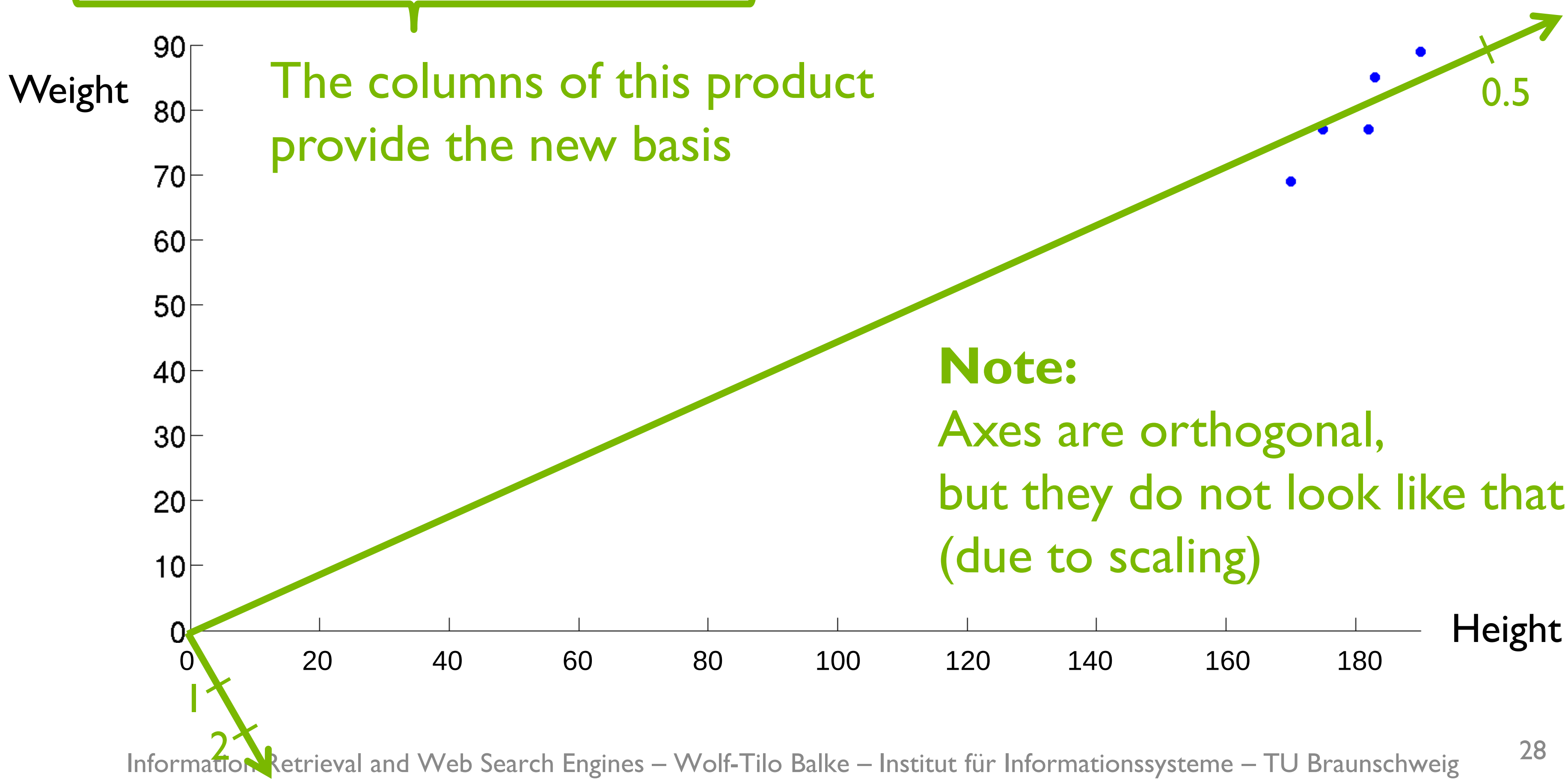
$$= \underbrace{\begin{pmatrix} 0.9146 & 0.4043 \\ 0.4043 & -0.9146 \end{pmatrix}}_{U} \cdot \underbrace{\begin{pmatrix} 440.3705 & 0 \\ 0 & 8.7638 \end{pmatrix}}_{S} \cdot \underbrace{\begin{pmatrix} 0.4164 & 0.4342 & 0.4487 & 0.4581 & 0.4763 \\ 0.6418 & 0.0375 & 0.3605 & -0.4283 & -0.5228 \end{pmatrix}}_{V}$$

Example

$$\begin{pmatrix} 170 & 175 & 182 & 183 & 190 \\ 69 & 77 & 77 & 85 & 89 \end{pmatrix}$$

$$= \begin{pmatrix} 0.9146 & 0.4043 \\ 0.4043 & -0.9146 \end{pmatrix} \cdot \begin{pmatrix} 440.3705 & 0 \\ 0 & 8.7638 \end{pmatrix} \cdot \begin{pmatrix} 0.4164 & 0.4342 & 0.4487 & 0.4581 & 0.4763 \\ 0.6418 & 0.0375 & 0.3605 & -0.4283 & -0.5228 \end{pmatrix}$$

The columns of this product provide the new basis



Note:
Axes are orthogonal,
but they do not look like that
(due to scaling)

Low Rank Approximation

- $A = USV$
  - $U \in \mathbb{R}^{m \times r}$: column-orthonormal
  - $S \in \mathbb{R}^{r \times r}$: diagonal
  - $V \in \mathbb{R}^{r \times n}$: row-orthonormal
- Since S is diagonal,
  A can be written as a **sum of matrices:**

$$A = s_1 u_{\star,1} v_{1,\star} + \cdots + s_r u_{\star,r} v_{r,\star}$$

First singular value

First left singular vector (column vector)

First right singular vector (row vector)

Low Rank Approximation

$$A = s_1 u_{\star,1} v_{1,\star} + \cdots + s_r u_{\star,r} v_{r,\star}$$

- The *i*-th summand is scaled by s_i
- Remember: $s_1 \geq s_2 \geq \cdots \geq s_r > 0$
  - The first summands are most important
  - The last ones have low impact on A (if their s_i's are small)

- **Idea:**
  Get an **approximation** of A
  by removing some less important summands
- This saves space and could remove small noise in the data

Low Rank Approximation

- **Rank-*k* approximation** of A (for any $k = 0, \ldots, r$):

$$A_k = s_1 u_{\star,1} v_{1,\star} + \cdots + s_k u_{\star,k} v_{k,\star}$$

- Let U_k denote the matrix U after removing the columns $k + 1$ to r
- Let S_k denote the matrix S after removing both the rows and columns $k + 1$ to r
- Let V_k denote the matrix V after removing the rows $k + 1$ to r
- Then it is $A_k = U_k \cdot S_k \cdot V_k$

Low Rank Approximation

- **Rank-*k* approximation** of *A* (for any *k* = 0, …, *r*):

$$A_k = s_1 u_{\star,1} v_{1,\star} + \cdots + s_k u_{\star,k} v_{k,\star}$$

- How large is the approximation error?
- The error can be measured using the Frobenius distance
- The **Frobenius distance** of two matrices $A, B \in \mathbb{R}^{m \times n}$ is:

$$d_F(A, B) = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} (a_{i,j} - b_{i,j})^2}$$

- Roughly the same as the mean squared entry-wise error

Low Rank Approximation

$$A_k = s_1 u_{\star,1} v_{1,\star} + \cdots + s_k u_{\star,k} v_{k,\star}$$

$$d_F(A, B) = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} (a_{i,j} - b_{i,j})^2}$$

- **Theorem:**
  For any $(m \times n)$-matrix B of rank at most k,
  it is $d_F(A, B) \geq d_F(A, A_k)$

- Therefore, A_k is an **optimal** rank-k approximation of A

Low Rank Approximation

$$A_k = s_1 u_{\star,1} v_{1,\star} + \cdots + s_k u_{\star,k} v_{k,\star}$$

$$d_F(A, B) = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} (a_{i,j} - b_{i,j})^2}$$

- **Theorem:**
  It is

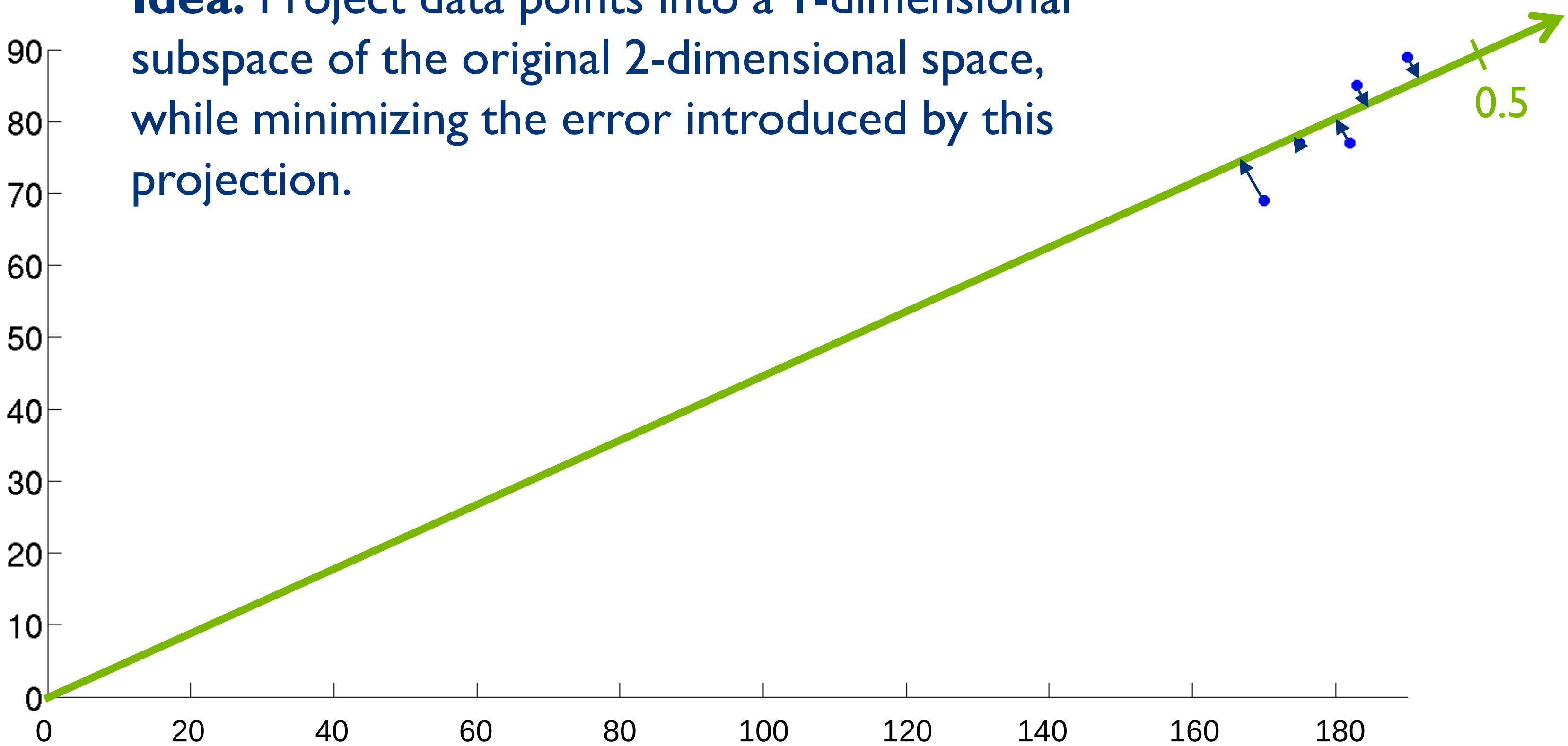
$$d_F(A, A_k) = \sqrt{s_{k+1}^2 + \cdots + s_r^2} \leq \sqrt{r - k} \cdot s_{k+1}$$

- If the singular values starting at s_{k+1} are "small enough," the approximation A_k is "good enough"

Example

- Let's ignore the second axis…

Idea: Project data points into a 1-dimensional subspace of the original 2-dimensional space, while minimizing the error introduced by this projection.



Example

- SVD:

$$\begin{pmatrix} 170 & 175 & 182 & 183 & 190 \\ 69 & 77 & 77 & 85 & 89 \end{pmatrix}$$

$$= \begin{pmatrix} 0.9146 & 0.4043 \\ 0.4043 & -0.9146 \end{pmatrix} \cdot \begin{pmatrix} 440.3705 & 0 \\ 0 & 8.7638 \end{pmatrix} \cdot \begin{pmatrix} 0.4164 & 0.4342 & 0.4487 & 0.4581 & 0.4763 \\ 0.6418 & 0.0375 & 0.3605 & -0.4283 & -0.5228 \end{pmatrix}$$

- Rank-1 approximation:

$$\begin{pmatrix} 0.9146 \\ 0.4043 \end{pmatrix} \cdot 440.3705 \cdot \begin{pmatrix} 0.4164 & 0.4342 & 0.4487 & 0.4581 & 0.4763 \end{pmatrix}$$

$$= \begin{pmatrix} 167.7261 & 174.8671 & 180.7228 & 184.5177 & 191.8526 \\ 74.1440 & 77.3007 & 79.8892 & 81.5668 & 84.8092 \end{pmatrix}$$

Connection to Eigenvectors

- Let A be an $(m \times n)$-matrix and $A = USV$ its SVD
- Then:

$$AA^{\mathsf{T}} = USV(USV)^{\mathsf{T}} = USVV^{\mathsf{T}}S^{\mathsf{T}}U^{\mathsf{T}} = US^2U^{\mathsf{T}}$$

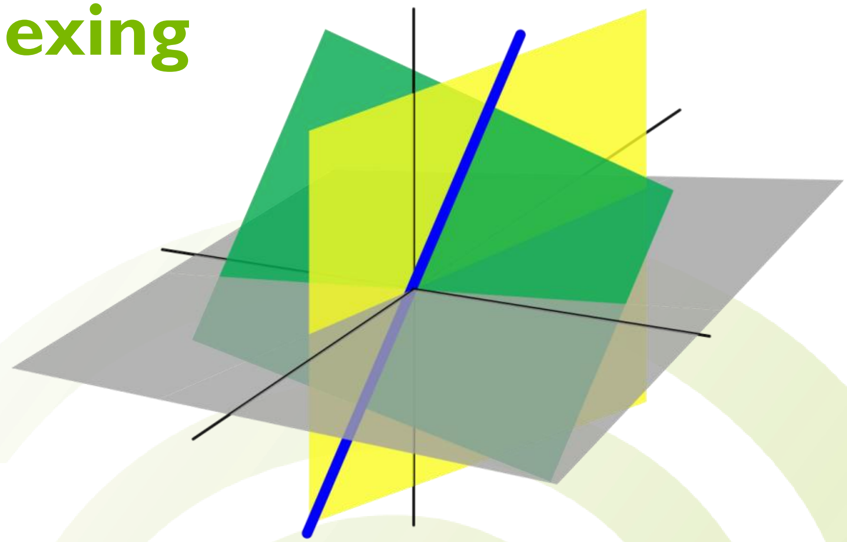
***V* is row-orthonormal, i.e. $VV^{\mathsf{T}} = I$**

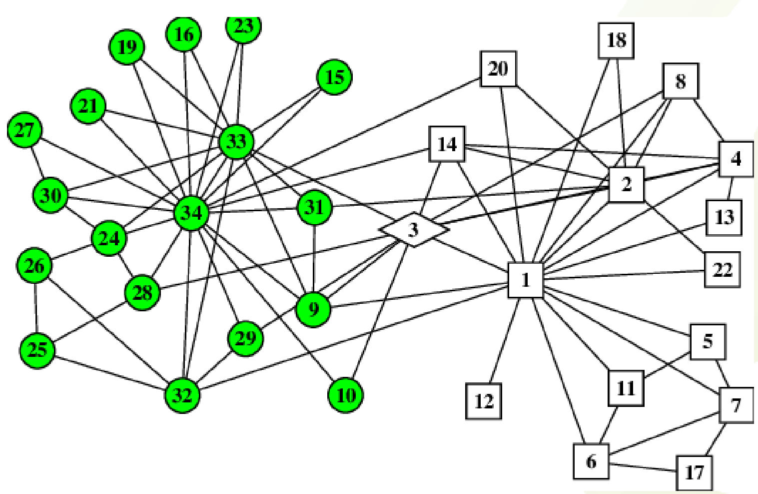
S^2 is still diagonal (entries got squared)

- **Theorem:**
  U's columns are the **eigenvectors** of AA^{T},
  the matrix S^2 contains the corresponding **eigenvalues**
- Similarly, V's rows are the eigenvectors of $A^{\mathsf{T}}A$,
  S^2 again contains the eigenvalues

Latend Semantic Indexing

1. Recap of Linear Algebra
2. Singular Value Decomposition
3. **Latent Semantic Indexing**





Latent Semantic Indexing

- **Idea** of Dumais *et al.* (1988):
  Apply the SVD to a term–document matrix!
- The *r* intermediate dimensions correspond to "topics"
  - Terms that usually occur together get bundled (synonyms)
  - Terms having several meanings get assigned to several topics (polysemes)
- Discarding dimensions having small singular values removes "noise" from the data…
  - Low rank approximations enhance data quality!

Example

- Example from (Berry *et al.*, 1995):
- A small collection of book titles

Label	Titles
B1	A Course on Integral Equations
B2	Attractors for Semigroups and Evolution Equations
B3	Automatic Differentiation of Algorithms: Theory, Implementation, and Application
B4	Geometrical Aspects of Partial Differential Equations
B5	Ideals, Varieties, and Algorithms – An Introduction to Computational Algebraic Geometry and Commutative Algebra
B6	Introduction to Hamiltonian Dynamical Systems and the N-Body Problem
B7	Knapsack Problems: Algorithms and Computer Implementations
B8	Methods of Solving Singular Systems of Ordinary Differential Equations
B9	Nonlinear Systems
B10	Ordinary Differential Equations
B11	Oscillation Theory for Neutral Differential Equations with Delay
B12	Oscillation Theory of Delay Differential Equations
B13	Pseudodifferential Operators and Nonlinear Partial Differential Equations
B14	Sinc Methods for Quadrature and Differential Equations
B15	Stability of Stochastic Differential Equations with Respect to Semi-Martingales
B16	The Boundary Integral Approach to Static and Dynamic Contact Problems
B17	The Double Mellin-Barnes Type Integrals and Their Applications to Convolution Theory

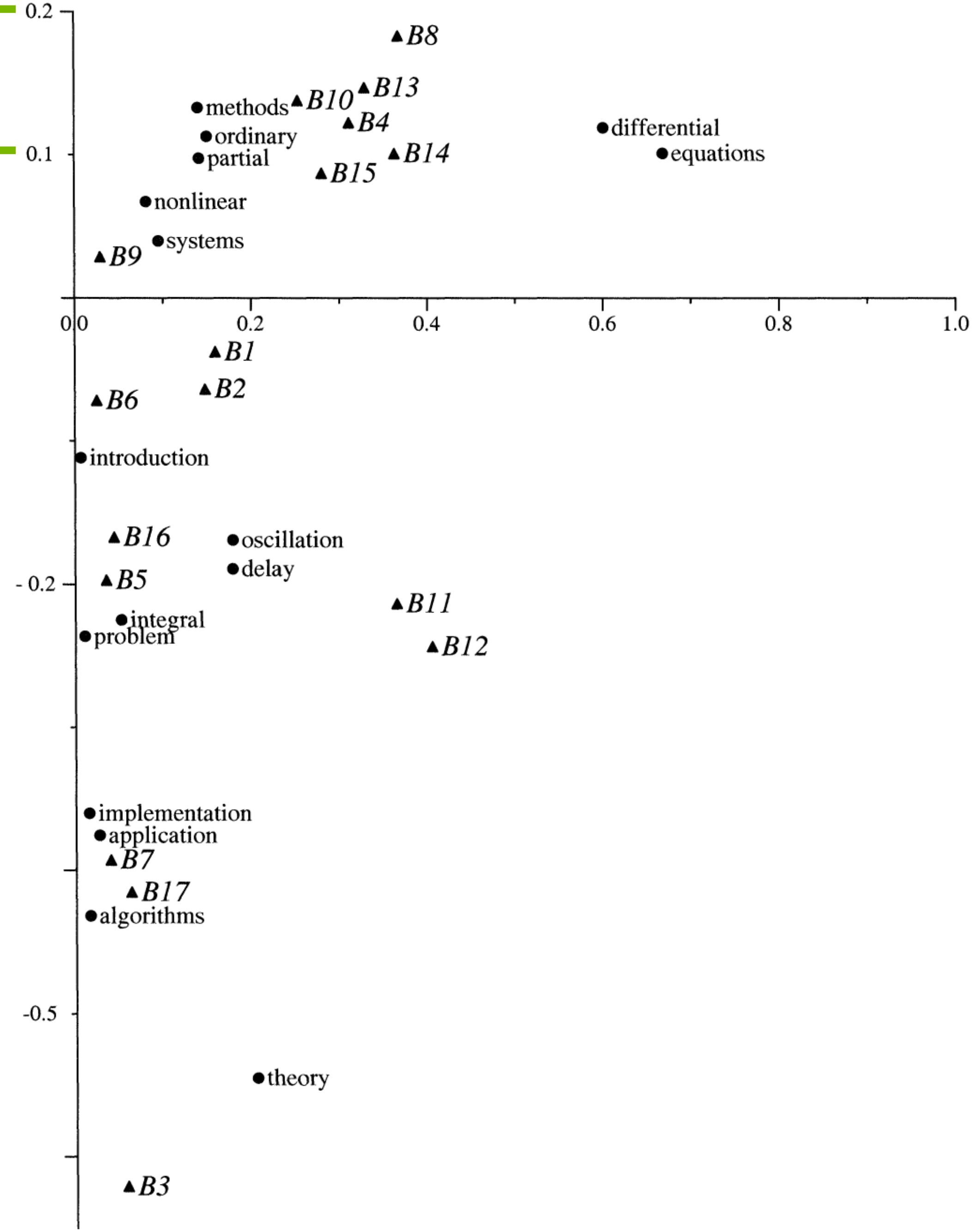
Example

- Term–document matrix
  (binary, since no term occured more than once):

Terms	Documents																
	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15	B16	B17
algorithms	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0
application	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
delay	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0
differential	0	0	0	1	0	0	0	1	0	1	1	1	1	1	1	0	0
equations	1	1	0	1	0	0	0	1	0	1	1	1	1	1	1	0	0
implementation	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0
integral	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
introduction	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
methods	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
nonlinear	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0
ordinary	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0
oscillation	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0
partial	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0
problem	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	1	0
systems	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0
theory	0	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	1

Example

- The first two dimensions of the SVD:
- Books and terms are plotted using the new basis' coordinates
- Similar terms have similar coordinates



Mapping into Latent Space

- How to exactly map documents and terms into the latent space?
- Recall: $A_k = U_k S_k V_k$
- To **get rid of the scaling factors** (singular values), S_k usually is split up and moved into U_k and V_k:
  - Let $S_k^{1/2}$ denote the matrix that results from extracting square roots from S_k (entry-wise)
  - Define $U_k' = U_k S_k^{1/2}$ and $V_k' = S_k^{1/2} V_k$, which gives $A_k = U_k' V_k'$
- Then:
  - The latent space coordinates of the *j*-th document are given by the *j*-th column of V_k'
  - The *i*-th term's coordinates are given by the *i*-th row of U_k'

Processing Queries

- How does querying work?
- **Idea:** Map the query vector $q \in \mathbb{R}^m$ into the latent space
- But: How to map **new documents/queries** into the latent space?
- Let $q' \in \mathbb{R}^k$ denote the query's (yet unknown) coordinates in latent space
- Assuming that q and q' are column vectors, we know that the following must be true (by definition of the SVD):

$$q = U_k' \cdot q'$$

Processing Queries

$$q = U_k' \cdot q' = U_k S_k^{1/2} \cdot q'$$

- Now, let's **solve this equation** with respect to *q*':
  - Multiply by U_k^{T} on the left-hand side:

$$U_k^{\mathrm{T}} \cdot q = U_k^{\mathrm{T}} U_k S_k^{1/2} \cdot q' = S_k^{1/2} \cdot q'$$

  - Multiply by $S_k^{-1/2}$ (the entry-wise reciprocal of $S^{1/2}$):

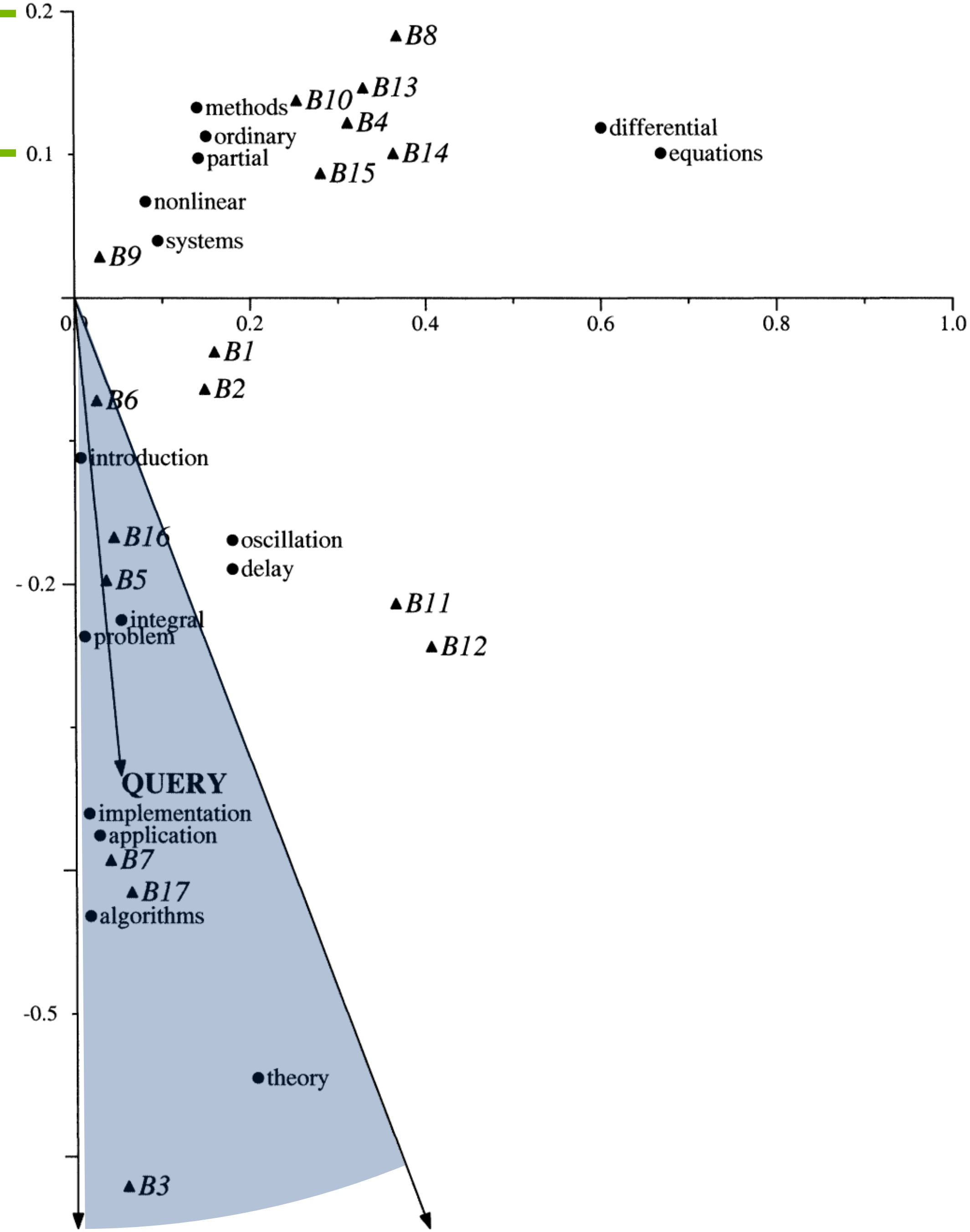
$$S_k^{-1/2} U_k^{\mathrm{T}} \cdot q = S_k^{-1/2} S_k^{1/2} \cdot q' = q'$$

- Thus, finally:

$$q' = S_k^{-1/2} U_k^{\mathrm{T}} \cdot q = S_k^{-1} U_k'^{\mathrm{T}} \cdot q$$

Example

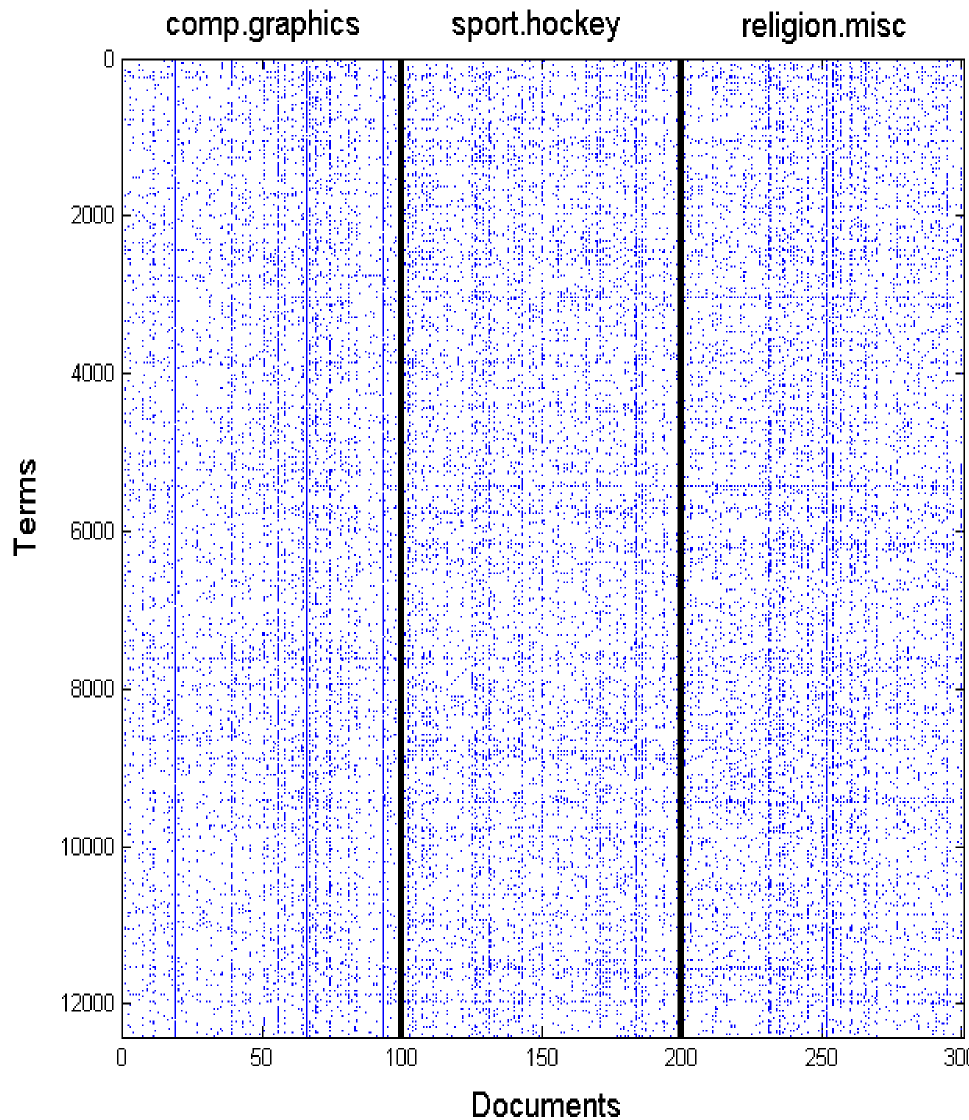
- Query = "application theory"
- All books within the shaded area have a cosine similarity to the query of at least 0.9



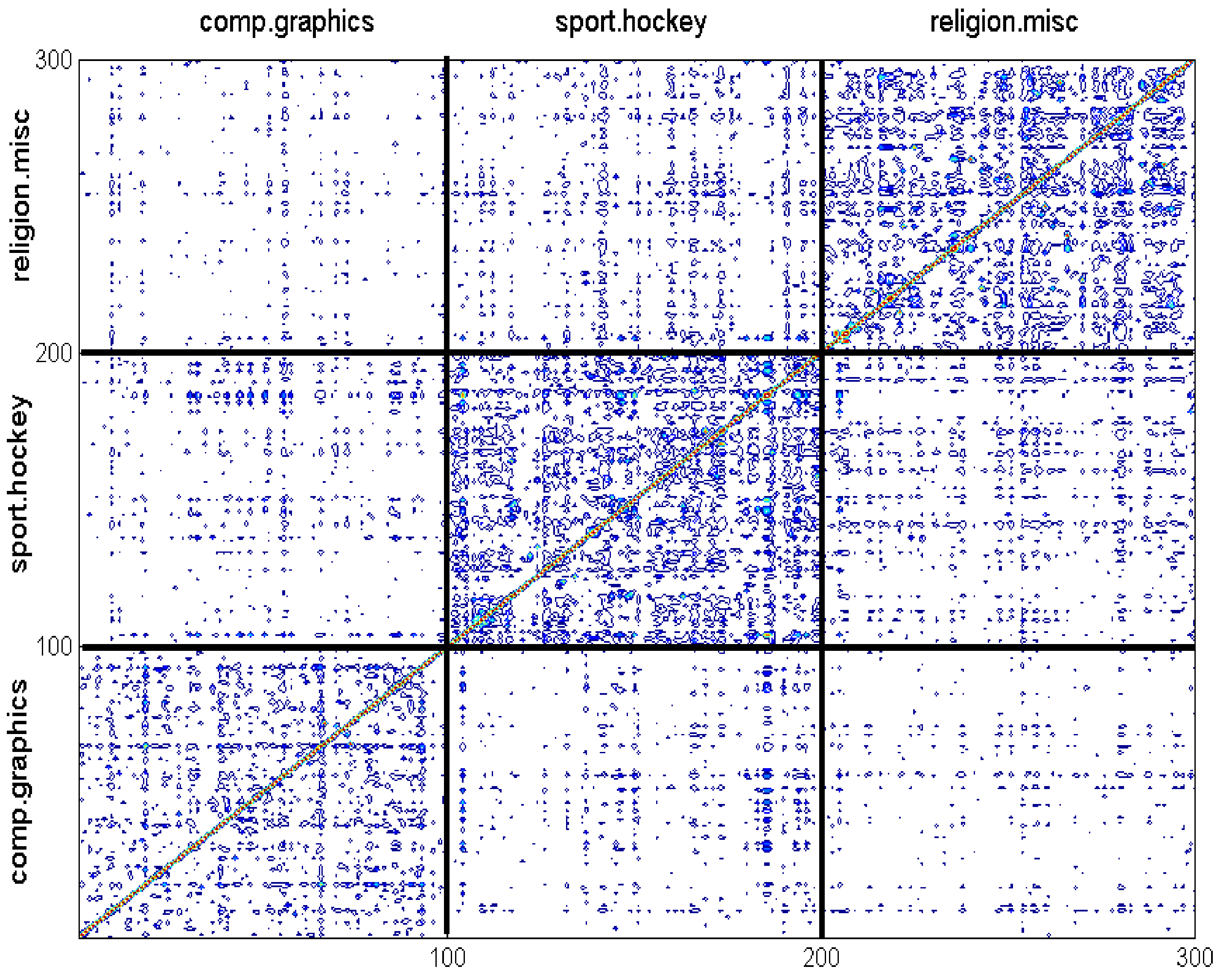
Another Example

- Example by Mark Girolami (University of Glasgow)

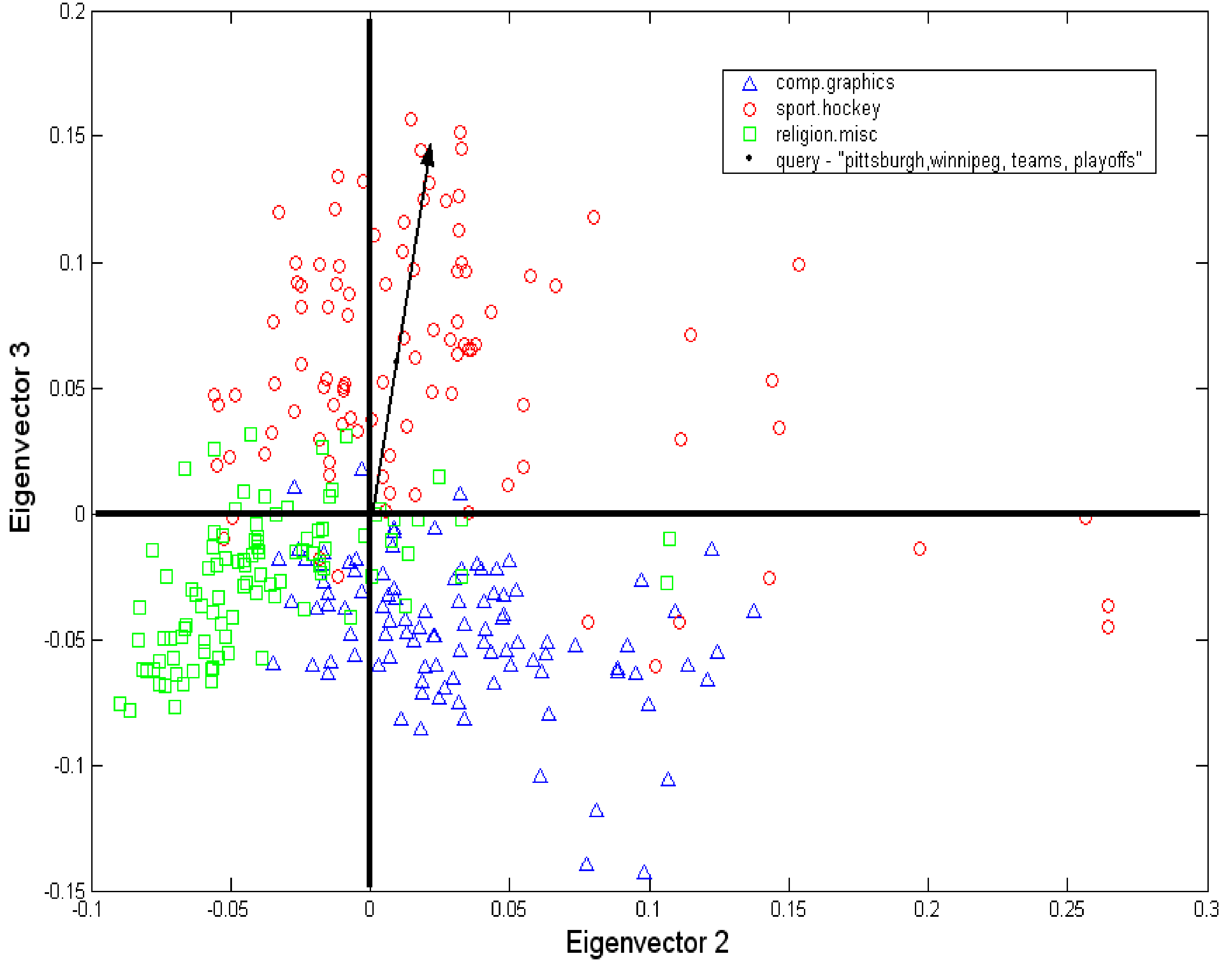
- Documents from a collection of Usenet postings



Another Example



Another Example



Yet Another Example

- Reuters-21578 collection
  - 21578 short newswire messages from 1987
- Top-3 results when querying for **taxes reagan** using LSI:

FITZWATER SAYS REAGAN STRONGLY AGAINST TAX HIKE
WASHINGTON, March 9 - White House spokesman Marlin Fitzwater said President Reagan's record in opposing tax hikes is "long and strong" and not about to change.

ROSTENKOWSKI SAYS WILL BACK U.S. TAX HIKE, BUT DOUBTS PASSAGE WITHOUT REAGAN SUPPORT

WHITE HOUSE SAYS IT OPPOSED TO TAX INCREASE AS UNNECESSARY

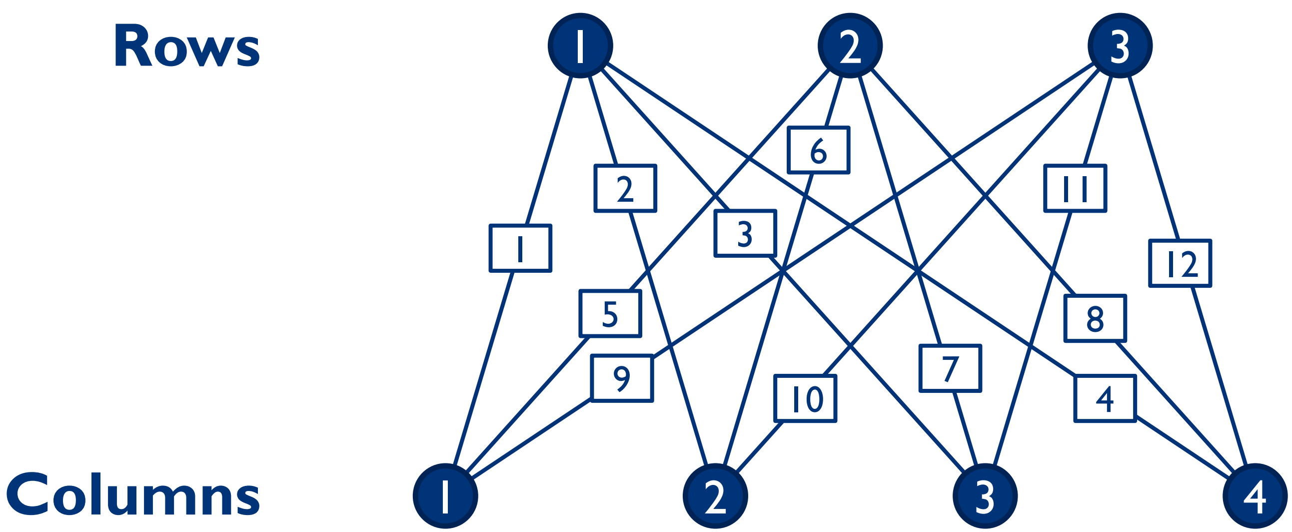
  - **The last document doesn’t mention the term “reagan”!**

A Different View on LSI

Detour

- Use a model similar to **neural networks**
- **Example:**
  $m = 3$, $n = 4$

$$A = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix}$$



A Different View on LSI

Detour

- SVD representation:

rank(A) = 2

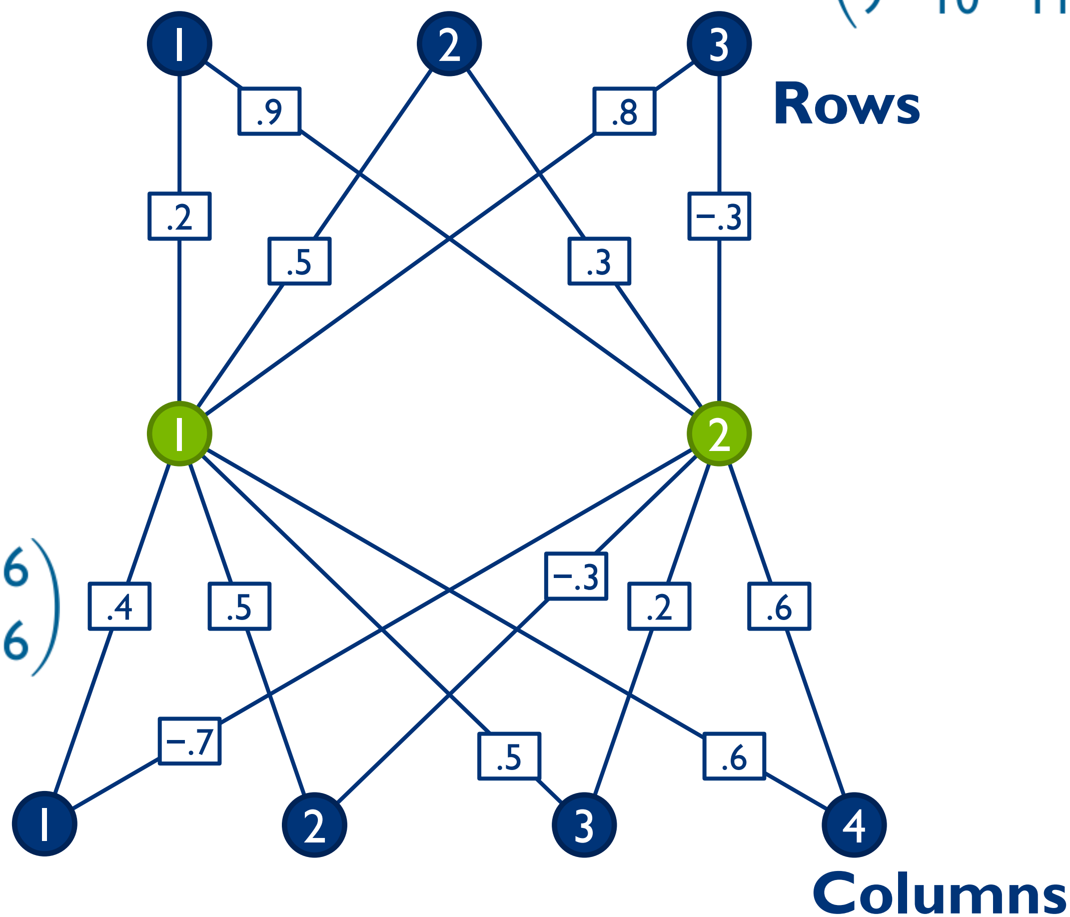
$$A = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix}$$

$$U \approx \begin{pmatrix} 0.2 & 0.9 \\ 0.5 & 0.3 \\ 0.8 & -0.3 \end{pmatrix}$$

$$S \approx \begin{pmatrix} 25.4 & 0 \\ 0 & 1.7 \end{pmatrix}$$

$$V \approx \begin{pmatrix} 0.4 & 0.5 & 0.5 & 0.6 \\ -0.7 & -0.3 & 0.2 & 0.6 \end{pmatrix}$$

For a given column, its rows in V represent the column's connections' strength to the topics



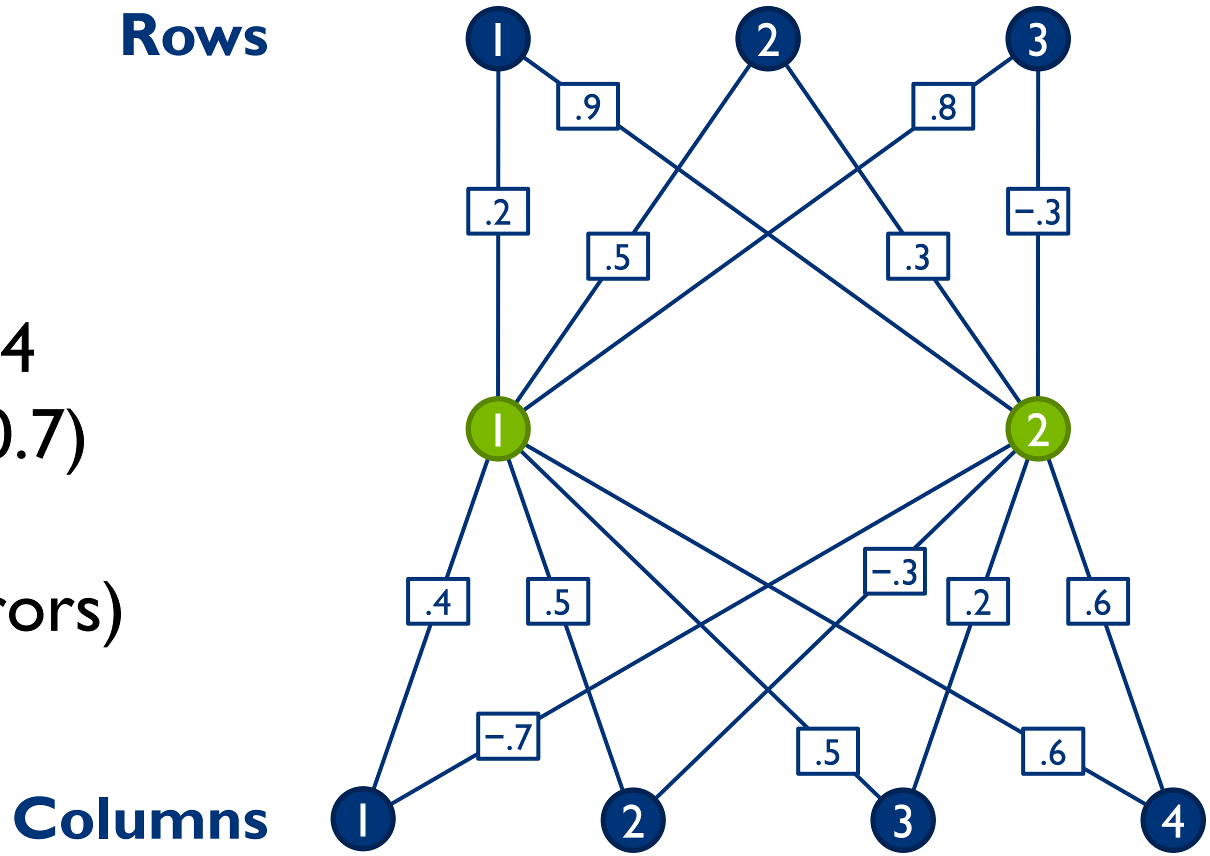
A Different View on LSI

Detour

- Reconstruction of *A* by multiplication:

$$A = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix}$$

$$S \approx \begin{pmatrix} 25.4 & 0 \\ 0 & 1.7 \end{pmatrix}$$



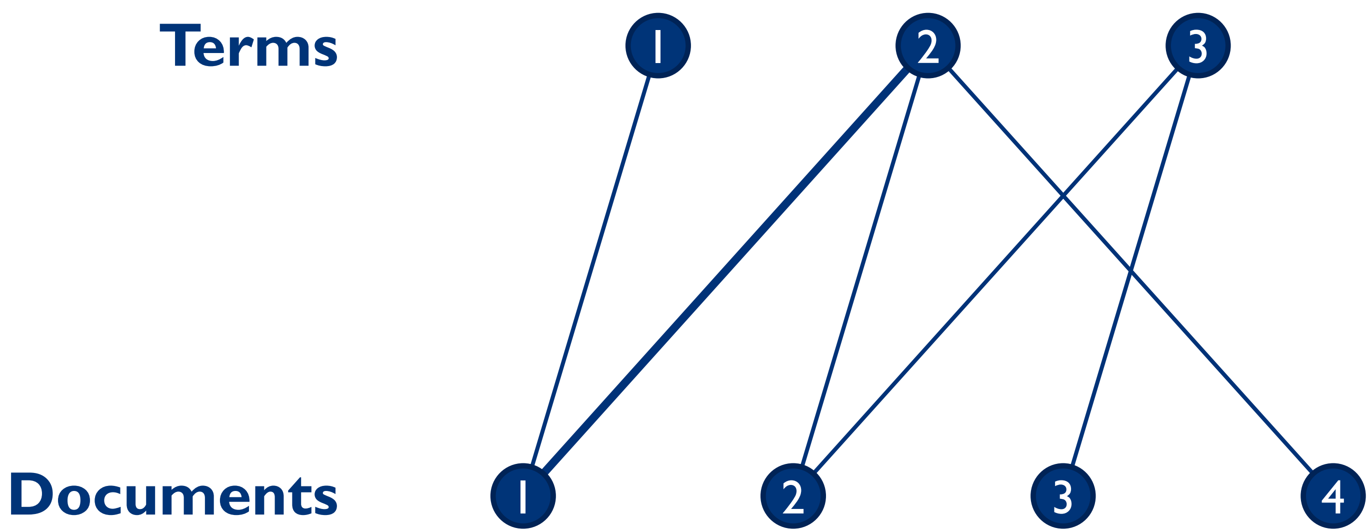
- $a_{2,1}$
  $= \quad 0.5 \cdot 25.4 \cdot 0.4$
  $+ 0.3 \cdot 1.7 \cdot (-0.7)$
  $= 5.08 - 0.357$
  ≈ 5 (rounding errors)

A Different View on LSI

Detour

- What does this mean for term–document matrices?

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

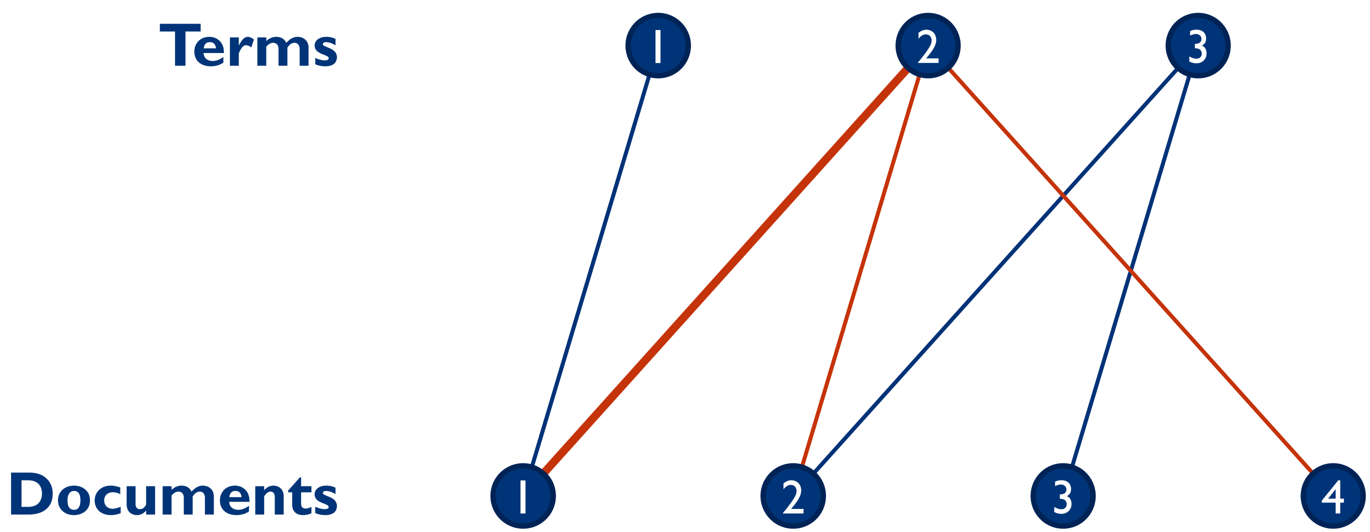


A Different View on LSI

Detour

- What documents contain **term 2?**

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

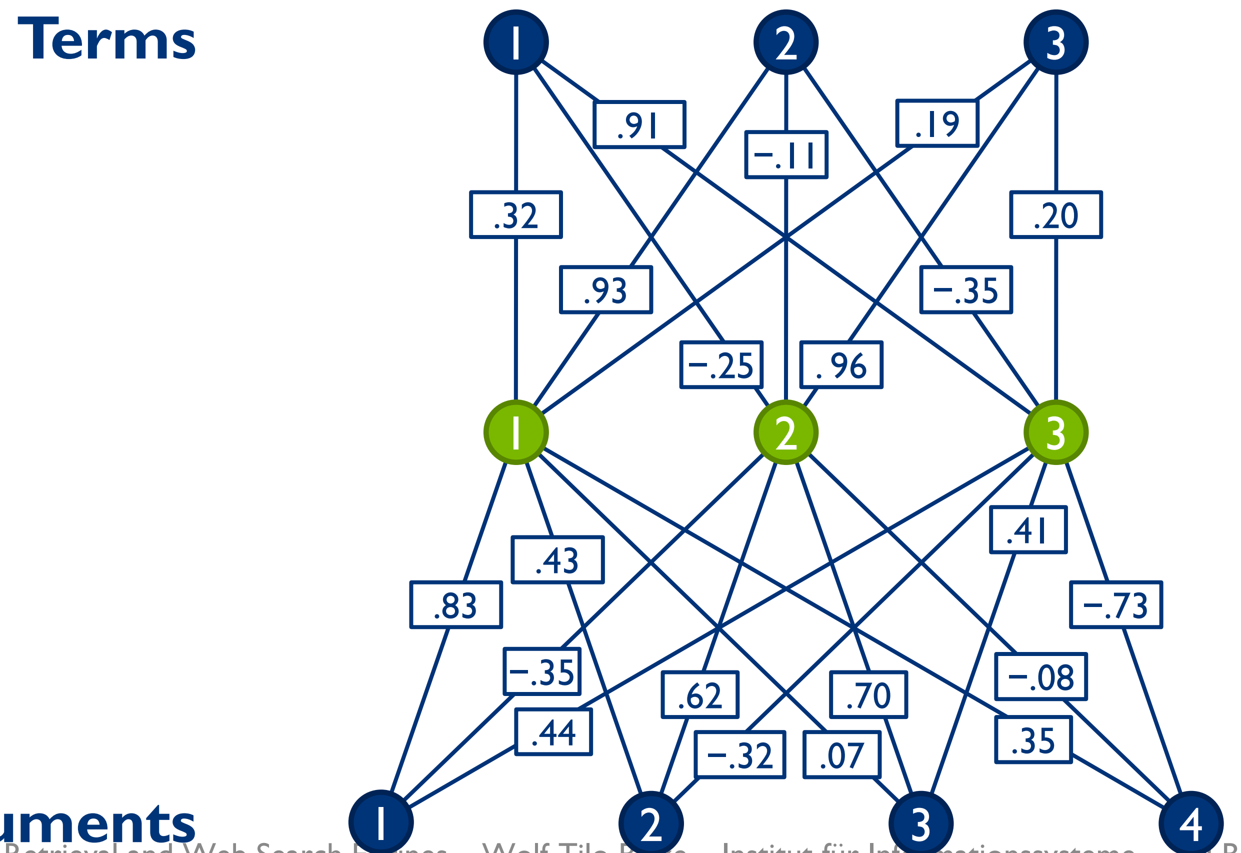


A Different View on LSI

Detour

- The SVD introduces an **intermediate layer:**

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 0.32 & -0.25 & 0.91 \\ 0.93 & -0.11 & -0.35 \\ 0.19 & 0.96 & 0.20 \end{pmatrix} \begin{pmatrix} 2.62 & 0 & 0 \\ 0 & 1.37 & 0 \\ 0 & 0 & 0.48 \end{pmatrix} \begin{pmatrix} 0.83 & 0.43 & 0.07 & 0.35 \\ -0.35 & 0.62 & 0.70 & -0.08 \\ 0.44 & -0.32 & 0.41 & -0.73 \end{pmatrix}$$

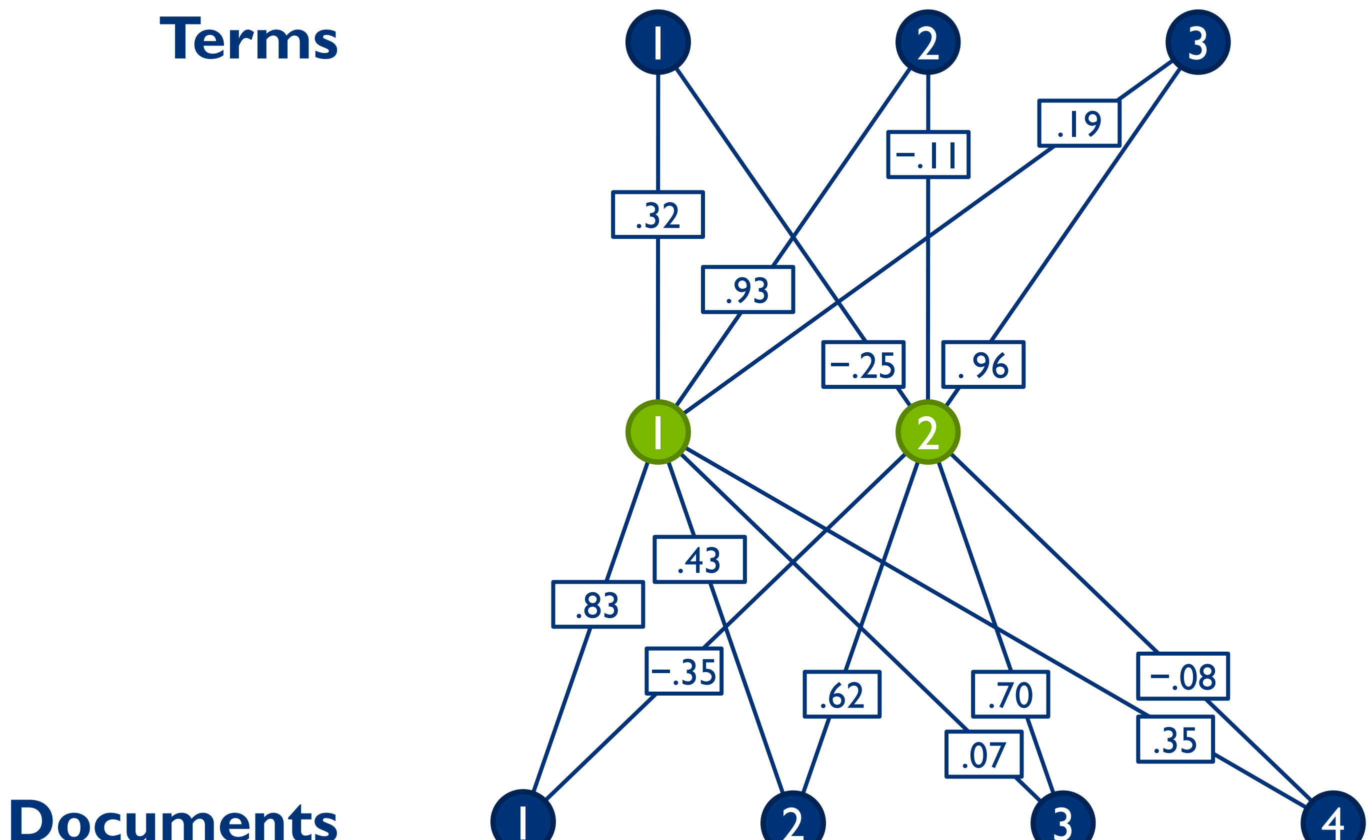


A Different View on LSI

Detour

- Remove unimportant topics:

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 0.32 & -0.25 & 0.91 \\ 0.93 & -0.11 & -0.35 \\ 0.19 & 0.96 & 0.20 \end{pmatrix} \begin{pmatrix} 2.62 & 0 & 0 \\ 0 & 1.37 & 0 \\ 0 & 0 & 0.48 \end{pmatrix} \begin{pmatrix} 0.83 & 0.43 & 0.07 & 0.35 \\ -0.35 & 0.62 & 0.70 & -0.08 \\ 0.44 & -0.32 & 0.41 & -0.73 \end{pmatrix}$$



Computing the SVD

- Computing the SVD on large matrices
  is at least **very difficult**
  - Traditional algorithms require matrices to be kept in memory
  - There are more specialized algorithm available,
    but computations still takes a long time on large collections
  - We have not been able to find any LSI experiment
    involving more than 1,000,000 documents…
  - Alternative: Compute LSI on a **subset** of the data…
- Recently, quite simple approximation algorithms have been developed that require much less memory and are relatively fast
  - For example, based on gradient descent
  - Maybe those approaches will make LSI easier to use in the future

What's the *k*?

- A central question remains:
  How many dimensions *k* should be used?
- It's a tradeoff:
  - Too many dimensions make computation expensive and lead to performance degradation in retrieval (no noise gets filtered out)
  - Too few dimensions also lead to performance degradation since important topics are left out
- The "right" *k* depends on the collection:
  - How specialized is it?
  - Are there special types of documents?

Pros and Cons

- **Pros**
  - Very good retrieval quality
  - Reasonable mathematical foundations
  - General tool for different purposes

- **Cons**
  - Latent dimensions found
    might be difficult to interpret
  - High computational requirements
  - The “right” *k* is hard to find

The Netflix Prize

Detour

- Netflix: Large DVD rental service
- The Netflix Prize
  - http://www.netflixprize.com
  - Win **$1,000,000**
- Dataset of customers' DVD ratings:
  - 480,189 customers
  - 17,700 movies
  - 100,480,507 ratings (scale: 1–5)
  - Density of rating matrix: 0.012

- Task: Estimate 2,817,131 ratings not published by Netflix

The Netflix Prize

Detour

- Computing a (sort of) SVD on the rating matrix has been proved to be highly successful
- Main problem here: The matrix is very **sparse!**
  - Sparse means **missing knowledge** (in contrast to LSI!)

	Matrix	Alien	Star Wars	Casablanca	Titanic
Joe	1	1	1	0	0
Jim	3	3	3	0	0
John	4	4	4	0	0
Jack	5	5	5	0	0
Jill	0	0	0	4	4
Jenny	0	0	0	5	5
Jane	0	0	0	2	2

$$= \begin{bmatrix} .14 & 0 \\ .42 & 0 \\ .56 & 0 \\ .70 & 0 \\ 0 & .60 \\ 0 & .75 \\ 0 & .30 \end{bmatrix} \begin{bmatrix} 12.4 & 0 \\ 0 & 9.5 \end{bmatrix} \begin{bmatrix} .58 & .58 & .58 & 0 & 0 \\ 0 & 0 & 0 & .71 & .71 \end{bmatrix}$$

SVD on Rating Data

Detour

- Each **movie** can be represented as a **point** in some *k*-dimensional **coordinate space**
- Many interesting applications
- Finding similar movies:

Rocky (1976)	Dirty Dancing (1987)	The Birds (1963)
Rocky II (1979)	Pretty Woman (1990)	Psycho (1960)
Rocky III (1982)	Footloose (1984)	Vertigo (1958)
Hoosiers (1986)	Grease (1978)	Rear Window (1954)
The Natural (1984)	Ghost (1990)	North By Northwest (1959)
The Karate Kid (1984)	Flashdance (1983)	Dial M for Murder (1954)

SVD on Rating Data

Detour

- Automatically reweighting genre assignments:

Movie	IMDb's genres	Reweighted genres
Back to the Future III (1990)	Adventure | Comedy | Family | Sci-Fi | Western	Adventure Comedy Family Sci-Fi Western
Rocky (1976)	Drama | Romance | Sport	Drama Romance Sport
Star Trek (1979)	Action | Adventure | Mystery | Sci-Fi	Action Adventure Mystery Sci-Fi
Titanic (1997)	Adventure | Drama | History | Romance	Adventure Drama History Romance

Next Lecture

1. Language models
2. Evaluation of retrieval quality