

Algebraic models to improve ranking and query expansion

Latent Semantic Indexing, Word
Embeddings

Is there anything more advanced than co-occurrences to learn word correlations?

- Traditional IR uses Term matching, → # of times the doc says “Albuquerque” – not fully appropriate
- We can use a **different approach**: compare all-pairs of query-document **terms**, → # of terms in the doc **that relate to Albuquerque**
- To detect these similarities:
 - **Latent Semantic Indexing**
 - Word embeddings (a.k.o. deep method – emerging technology)

Albuquerque is the most populous city in the U.S. state of New Mexico. The high-altitude city serves as the county seat of Bernalillo County, and it is situated in the central part of the state, straddling the Rio Grande. The city population is 557,169 as of the July 1, 2014, population estimate from the United States Census Bureau, and ranks as the 32nd-largest city in the U.S. The Metropolitan Statistical Area (or MSA) has a population of 902,797 according to the United States Census Bureau's most recently available estimate for July 1, 2013.

Passage *about* Albuquerque

Allen suggested that they could program a BASIC interpreter for the device; after a call from Gates claiming to have a working interpreter, MITS requested a demonstration. Since they didn't actually have one, Allen worked on a simulator for the Altair while Gates developed the interpreter. Although they developed the interpreter on a simulator and not the actual device, the interpreter worked flawlessly when they demonstrated the interpreter to MITS in Albuquerque, New Mexico in March 1975; MITS agreed to distribute it, marketing it as Altair BASIC.

Passage not about Albuquerque

The problem

- With the standard term-document matrix encoding, each term is a vector and dimensions are documents
- Different terms have no inherent similarity, e.g.:
Search: [0 2 0 0 0 0 0 0 0 0 1 0 0 0 0]
Information retrieval: [0 0 0 0 0 0 0 3 0 0 0 0 1 0 0]
- If query on *search* and document has *information retrieval*, then our query and document vectors are orthogonal. **Dot product is zero.** But these two words are very related!

Can we directly learn term relations?

- Basic IR is scoring on $\mathbf{q}^T \cdot \mathbf{d} / K$ (dot product of query and document vectors)
$$\frac{\vec{q} \cdot \vec{d}}{|\vec{q}| |\vec{d}|}$$
- No treatment of synonyms; no machine learning
- Can we learn a matrix \mathbf{W} to rank via $\mathbf{q}^T \mathbf{W} \mathbf{d}$, rather than $\mathbf{q}^T \cdot \mathbf{d}$?

Handwritten example illustrating term relations between "search ranking" and "information retrieval ranking".

"search ranking" (q^T): $(1 \ 0 \ 0 \ 1 \ 0)$ (words: se, in, re, ra, or)

W (matrix):

$$\begin{pmatrix} 1 & 0.7 & 0.5 & 0 & 0 \\ 0.3 & 1 & 0.2 & 0 & 0 \\ 0.5 & 0.2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0.7 \\ 0 & 0 & 0 & 0.7 & 1 \end{pmatrix}$$

"information retrieval ranking" (d): $\begin{pmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 0 \end{pmatrix}$ (words: se, in, re, ra, or (dering))

Result: $re = 2.2$

- Where W is a matrix that captures **similarity between words** (e.g., "search" and "information retrieval")?

Latent Semantic Indexing

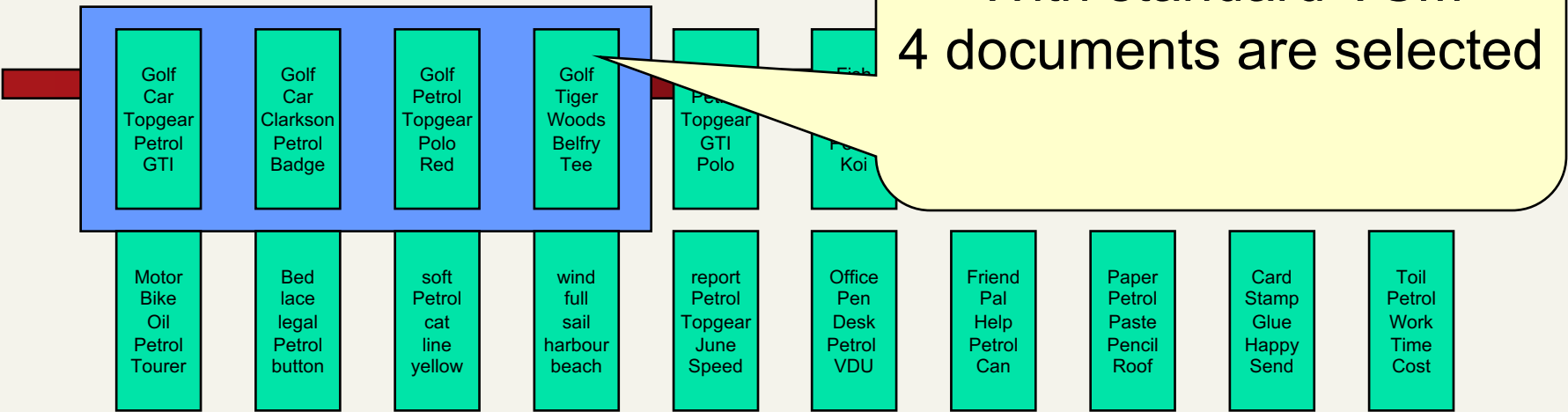
Latent Semantic Indexing

- Term-document matrices are very large, though most cells are “zeros”
- But the number of **topics** that people talk about is smaller (in some sense)
 - Clothes, movies, politics, ...
 - Each topic can be represented as a cluster of (semantically) related terms, e.g.: clothes=golf, jacket, shoe..
- Can we represent the term-document space by a lower dimensional “latent” space (**latent space=set of topics**)?

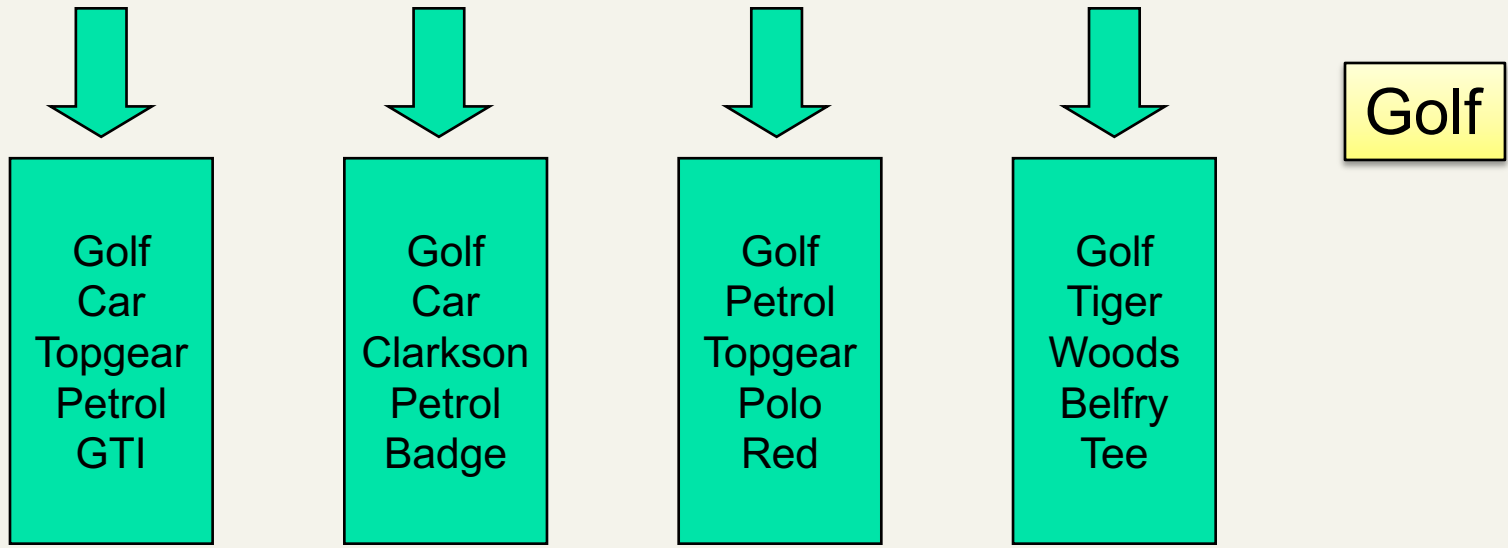
Searching with latent topics

- Given a collection of documents, LSI learns clusters of frequently co-occurring terms (ex: `information retrieval, ranking and web`)
- If you query with `ranking, information retrieval` LSI “automatically” extends the search to documents including also (and even ONLY) `web`

Document base (20)



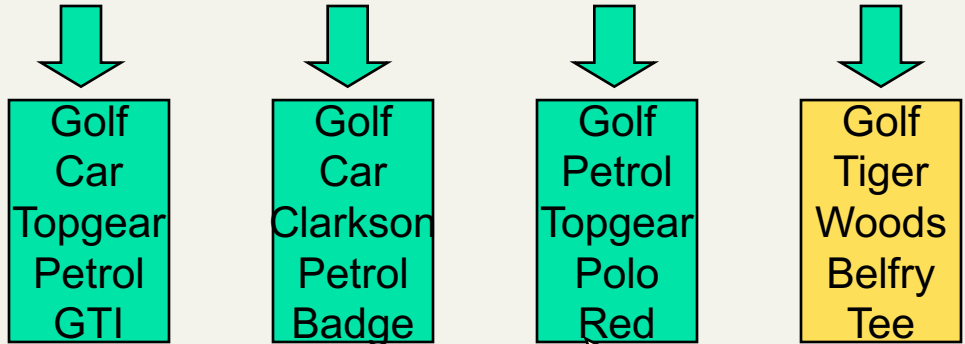
Selection based on 'Golf'



20 documents

Golf Car Topgear Petrol GTI	Golf Car Clarkson Petrol Badge	Golf Petrol Topgear Polo Red	Golf Tiger Woods Belfry Tee	Car Petrol Topgear GTI Polo	Fish Pond gold Petrol Koi	PC Dell RAM Petrol Floppy	Core Petrol Apple Pip Tree	Pea Pod Fresh Green French	Lupin Petrol Seed May April
Motor Bike Oil Petrol Tourer	Bed lace legal Petrol button	soft Petrol cat line yellow	wind full sail harbour beach	report Petrol Topgear June Speed	Office Pen Desk Petrol VDU	Friend Pal Help Petrol Can	Paper Petrol Paste Pencil Roof	Card Stamp Glue Happy Send	Toil Petrol Work Time Cost

Selection based on 'Golf'



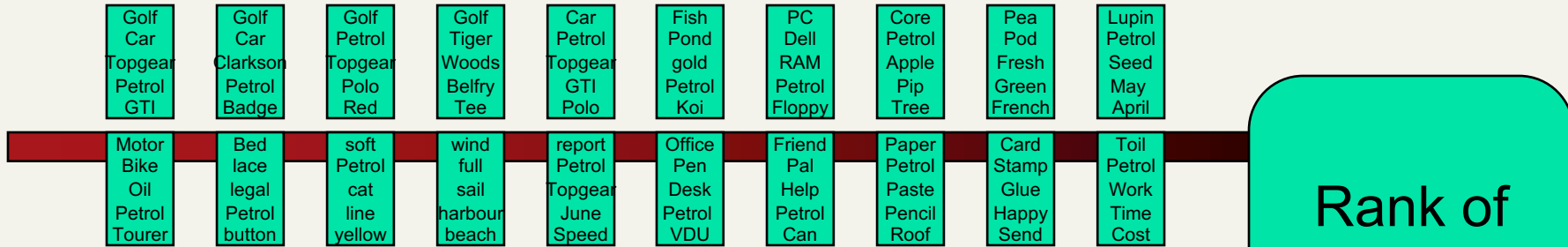
Rank of selected documents

tf.idf

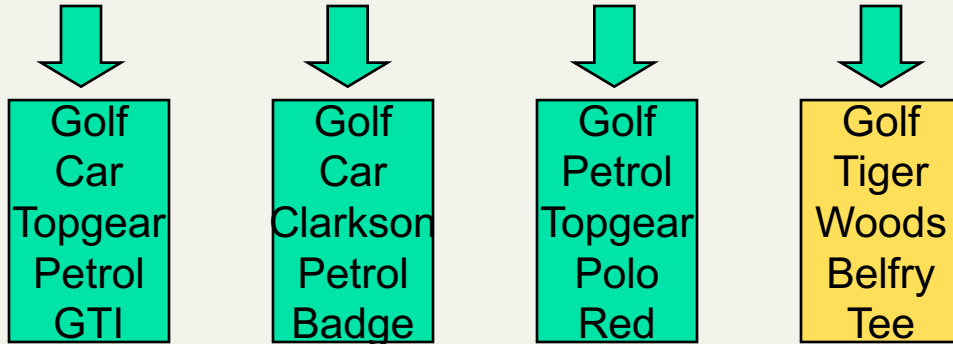
- Car
 $2 * (20/3) = 13$
- Topgear
 $2 * (20/3) = 13$
- Petrol
 $3 * (20/16) = 4$

The most relevant words associated with golf in these docs are:
Car, Topgear and Petrol

20 docs



Selezione basata su 'Golf'

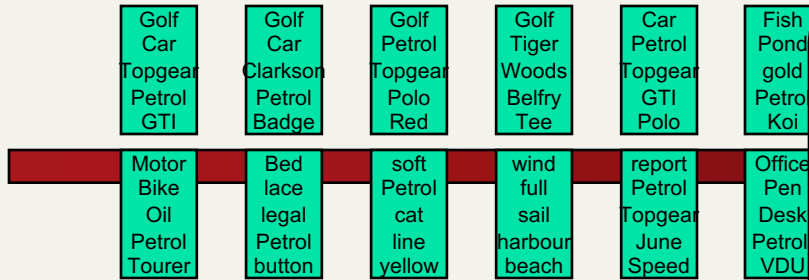


Rank of selected docs

If we consider the co-occurring terms with higher $tf \cdot idf$, **car** e **topgear** turn out to be related to **Golf** more than **petrol**, **wood**,...

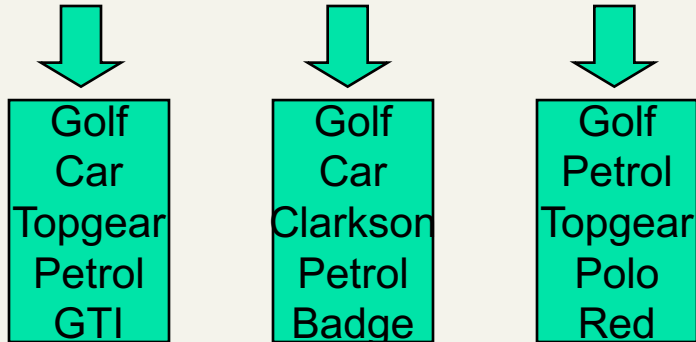
Car
 $2 \cdot (20/3) = 13$
Topgear
 $2 \cdot (20/3) = 13$
Petrol
 $3 \cdot (20/16) = 4$

20 docs



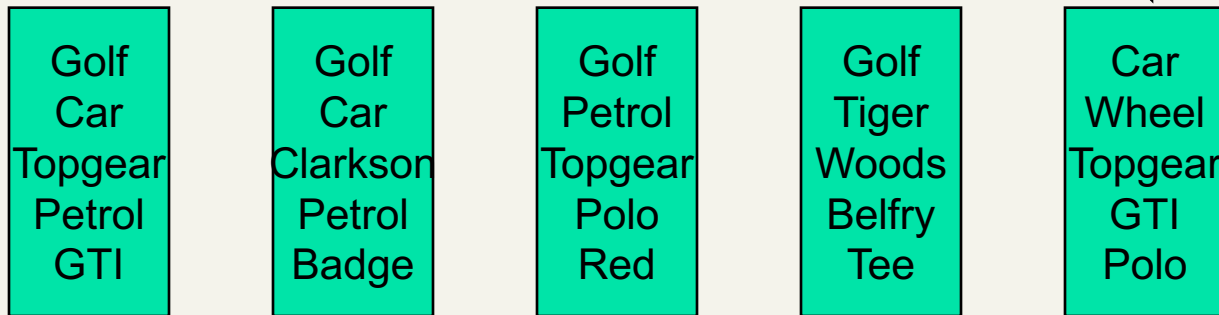
We now search with all the words in the "semantic domain" of Golf. The list of retrieved docs now is based on Golf and the other related words

Selection based on 'Golf'



Tiger
Woods
Belfry
Tee

Selection based on the semantic domain of Golf



Car
 $2 * (20/3) = 13$
Topgear
 $2 * (20/3) = 13$
Petrol
 $3 * (20/16) = 4$

20 docs

Golf
Car
Topgear
Petrol
GTI

Motor
Bike
Oil
Petrol
Tourer

Lupin
Petrol
Seed
May
April

Oil
Petrol
Work
Time
Lost

The co-occurrence based ranking improves the performance.
Note that one of the most relevant doc does NOT include the word Golf, and a doc with a "spurious" sense disappears

Select



Golf
Car
Topgear
Petrol
GTI

Petrol
Badge

Polo
Red

Selection based on semantic domain

Golf
Car
Topgear
Petrol
GTI

Golf
Car
Clarkson
Petrol
Badge

Golf
Petrol
Topgear
Polo
Red

Golf
Tiger
Woods
Belfry
Tee

Car
Wheel
Topgear
GTI
Polo

rank
of
selected
docs

Car
 $2 * (20/3) = 13$
Topgear
 $2 * (20/3) = 13$
Petrol
 $3 * (20/16) = 4$

Rank

30

17

17

0

26

Ranking with Latent Semantic Indexing

- Previous example just gives the intuition
- Latent Semantic Indexing is an **algebraic** method to identify clusters of co-occurring terms, called “latent topics”, and to compute query-doc similarity in a **latent space**, in which every coordinate is a latent topic.
- A “latent” quantity is one which **cannot directly observed**, what is observed is a measurement which may include some amount of random errors (topics are “latent” in this sense: we observe them, but they are an approximation of “true” semantic topics)
- Since it is an algebraic method, needs some linear algebra background

The LSI method: how to detect “topics”

Linear Algebra Background

Eigenvalues & Eigenvectors

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

$$\underline{\mathbf{A}}\underline{\mathbf{v}} = \lambda\underline{\mathbf{v}}$$

(right) eigenvector eigenvalue
 $\underline{\mathbf{v}} \in \mathbb{R}^m \neq \mathbf{0}$ $\lambda \in \mathbb{R}$

Example

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

$$\underline{\mathbf{A}}\underline{\mathbf{v}} = \lambda\underline{\mathbf{v}}$$

- Def: A vector $\underline{\mathbf{v}} \in \mathbb{R}^n$, $\underline{\mathbf{v}} \neq \underline{\mathbf{0}}$, is an **eigenvector** of a matrix $m \times m$ $\underline{\mathbf{A}}$ with corresponding **eigenvalue** λ , if:

$$\underline{\mathbf{A}}\underline{\mathbf{v}} = \lambda\underline{\mathbf{v}}$$

Algebraic method

- How many eigenvalues are there at most?

$$\underline{\mathbf{A}}\underline{\mathbf{v}} = \lambda\underline{\mathbf{v}}$$

equation has a non-zero solution if $|\underline{\mathbf{A}} - \lambda\underline{\mathbf{I}}| = 0$

$$\underline{\mathbf{A}}\underline{\mathbf{v}} = \lambda\underline{\mathbf{v}} \iff (\underline{\mathbf{A}} - \lambda\underline{\mathbf{I}})\underline{\mathbf{v}} = \mathbf{0}$$

Where \mathbf{I} is the identity matrix, and \mathbf{A} is $m \times m$
this is a m -th order equation in λ which can have **at most m distinct solutions** (roots of the characteristic polynomial) - can be complex even though \mathbf{A} is real.

Example of eigenvector/eigenvalues

$$A = \begin{pmatrix} 1 & -1 \\ 3 & 5 \end{pmatrix}, \quad v = \begin{pmatrix} 1 \\ -3 \end{pmatrix}, \quad \lambda = 4$$

$$A\underline{v} = \lambda\underline{v}$$

$$\begin{pmatrix} 1 & -1 \\ 3 & 5 \end{pmatrix} \begin{pmatrix} 1 \\ -3 \end{pmatrix} = 4 \begin{pmatrix} 1 \\ -3 \end{pmatrix}$$

$$\begin{pmatrix} 1 + (-3)(-1) \\ 3 + 5(-3) \end{pmatrix} = \begin{pmatrix} 4 \\ -12 \end{pmatrix}$$

$$\begin{pmatrix} 4 \\ -12 \end{pmatrix} = \begin{pmatrix} 4 \\ -12 \end{pmatrix}$$

We show that \underline{v} is an eigenvector for A

Example of computation

remember

$$\det M = |M| = \begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$$

$$A = \begin{pmatrix} 1 & -1 \\ 3 & 5 \end{pmatrix}$$

$$\det(A - \lambda I) = 0$$

$$\begin{vmatrix} 1 & -1 \\ 3 & 5 \end{vmatrix} - \lambda \begin{vmatrix} 1 & 0 \\ 0 & 1 \end{vmatrix} = 0$$

2 and 4 are the eigenvalues of S

$$(1 - \lambda)(\lambda - 5) + 3 = 0$$

$$-6\lambda + 5 + 3 = 0$$

$$\lambda^2 - 6\lambda + 8 = 0$$

$$(\lambda - 4)(\lambda - 2) = 0$$

Characteristic polynomial

$$\begin{vmatrix} 1 & -1 \\ 3 & 5 \end{vmatrix} - \begin{vmatrix} \lambda & 0 \\ 0 & \lambda \end{vmatrix} = 0$$

$$\begin{vmatrix} 1 - \lambda & -1 \\ 3 & 5 - \lambda \end{vmatrix} = 0$$

$$\begin{pmatrix} 1-4 & -1 \\ 3 & 5-4 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$\begin{pmatrix} -3 & -1 \\ 3 & 1 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$\begin{cases} -3\alpha - \beta = 0 \\ 3\alpha + \beta = 0 \end{cases} \quad \beta = -3\alpha$$

$$\begin{pmatrix} 1-2 & -1 \\ 3 & 3-2 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

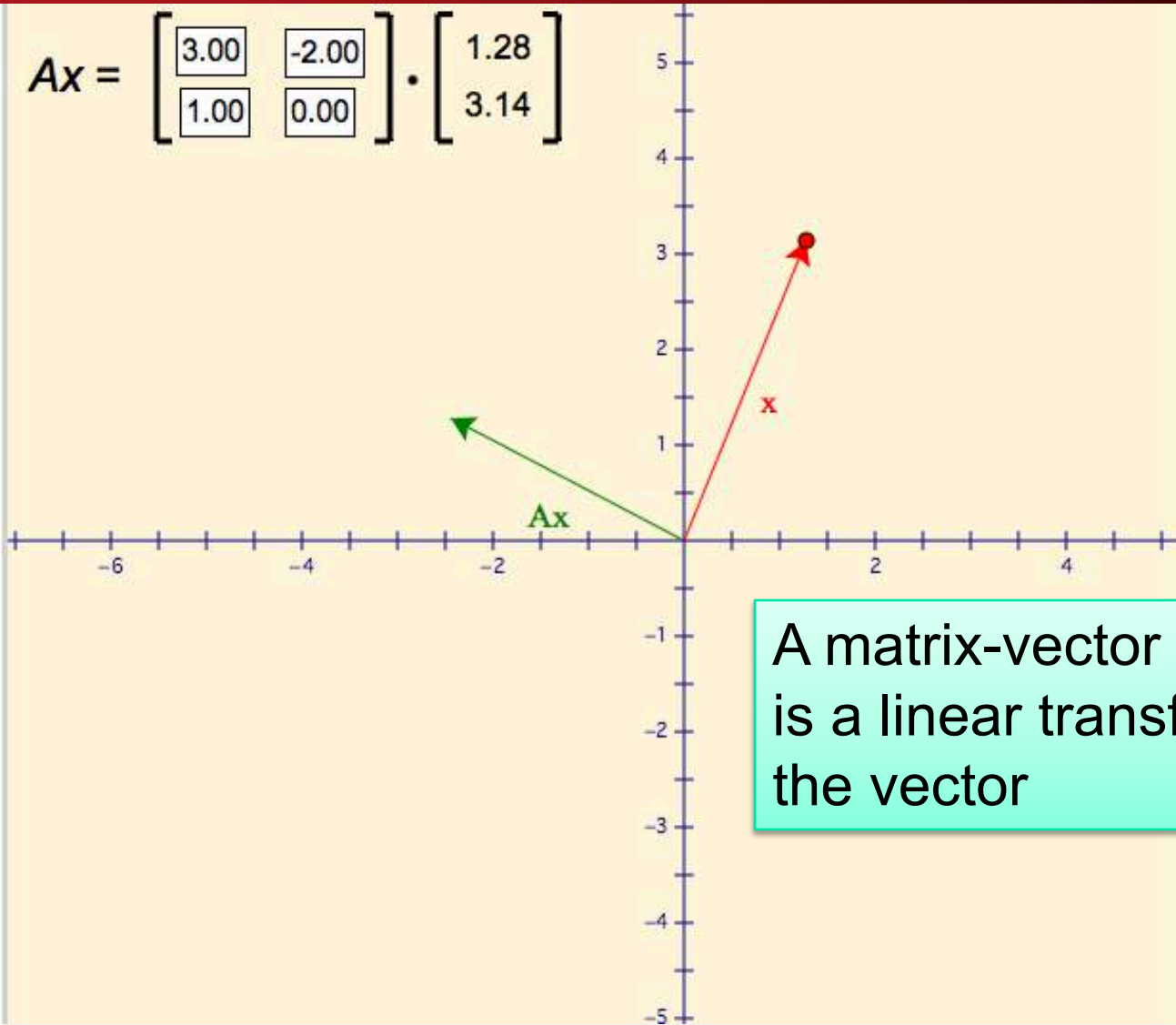
$$\begin{pmatrix} -1 & -1 \\ 3 & 3 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$\begin{cases} -\alpha - \beta = 0 \\ 3\alpha + 3\beta = 0 \end{cases} \quad \alpha = -\beta$$

Note that we compute only the **DIRECTION** of eigenvectors

Geometric interpretation

$$Ax = \begin{bmatrix} \boxed{3.00} & \boxed{-2.00} \\ \boxed{1.00} & \boxed{0.00} \end{bmatrix} \cdot \begin{bmatrix} 1.28 \\ 3.14 \end{bmatrix}$$



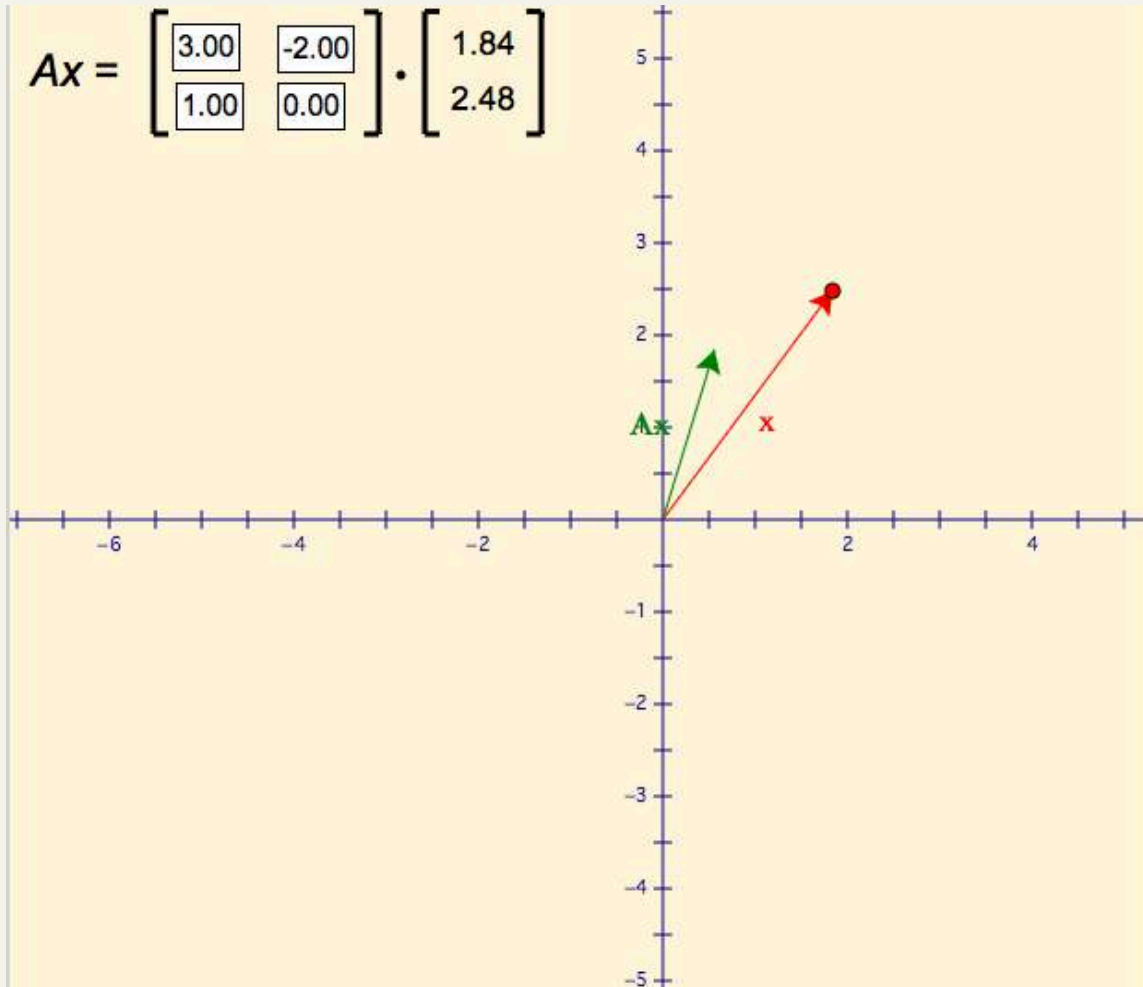
A matrix-vector multiplication Ax is a linear transformation over the vector

Matrix vector multiplication

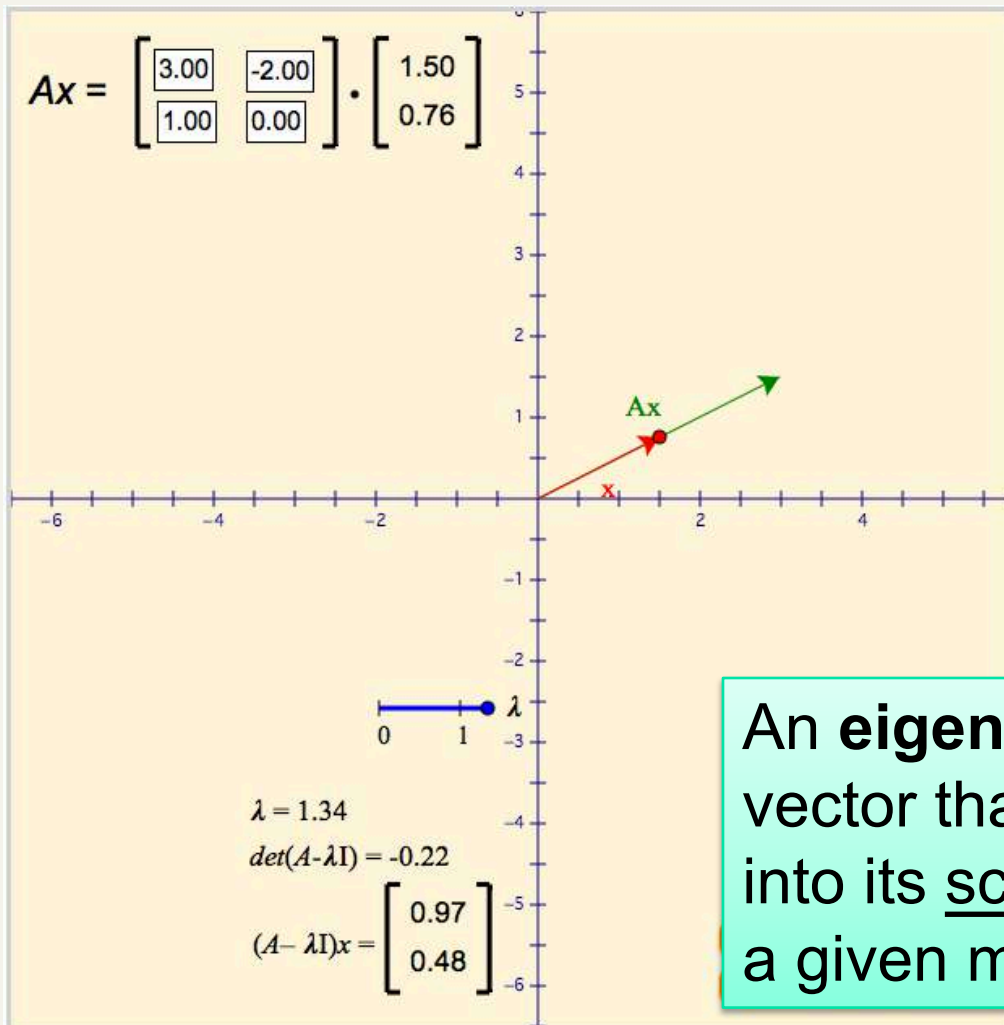
$$A\mathbf{x} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n \end{bmatrix}.$$

Matrix multiplication by a vector = a linear transformation of the initial vector, that implies **rotation** and **translation** of the original vector

Geometric interpretation



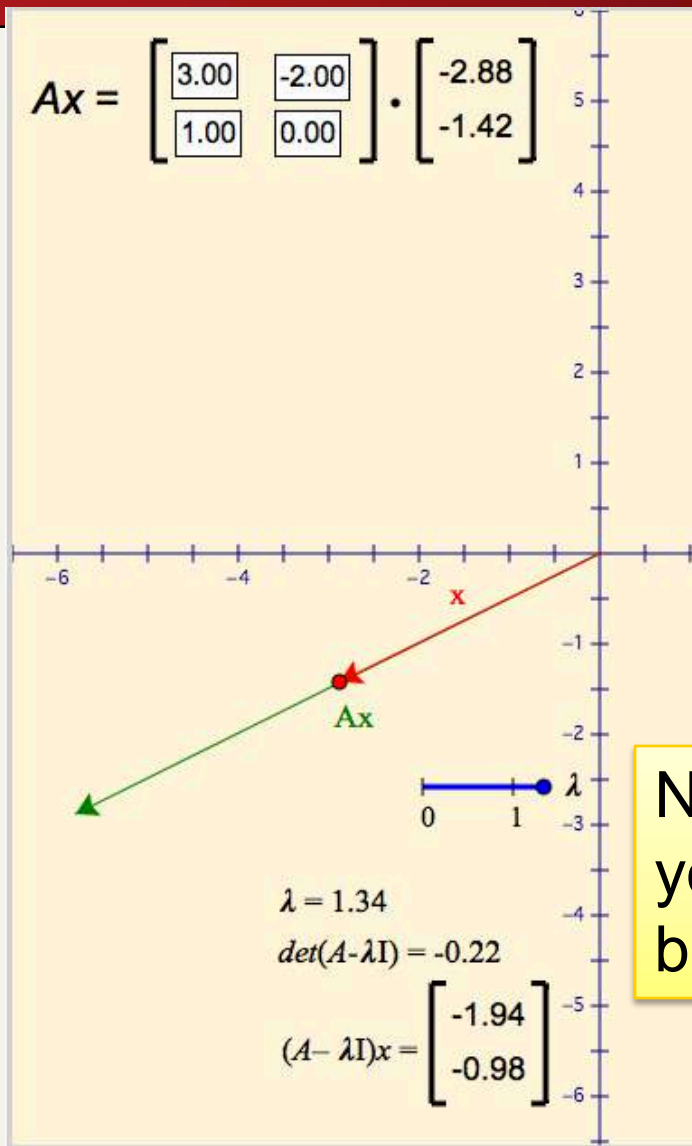
Geometric interpretation



$$Ax = 1.34x = \lambda x$$

An **eigenvector** is a special vector that is transformed into its scalar multiple under a given matrix (no rotation!)

Geometric interpretation



Here we found another eigenvector for the matrix A

Note that for **any single eigenvalue** you have **infinite** eigenvectors, but they have **the same direction**

Matrix-vector multiplication

- Eigenvectors of different eigenvalues are **linearly independent** (i.e. $\forall \alpha_1 \dots \alpha_n \rightarrow \alpha_1 v_1 + \dots \alpha_n v_n \neq 0$)
- For **square normal** matrixes eigenvectors of different eigenvalues define an **orthonormal space** and they are orthogonal.
- A square matrix is NORMAL iff it commutes with its transpose, i.e. $AA^T = A^T A$
- Example:

$$A = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix} \rightarrow AA^T = \begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{pmatrix} = A^T A$$

Difference between orthonormal and orthogonal?

- Orthogonal mean that the dot product is null (the cosin of the angle is zero).

Orthonormal mean that the dot product is null **and** the norm of the vectors is equal to 1.

What we are actually saying is that eigenvectors define a set of DIRECTIONS wich are orthogonal (=an othonormal space).

- If two or more vectors are orthonormal they are also orthogonal but the inverse is not true.

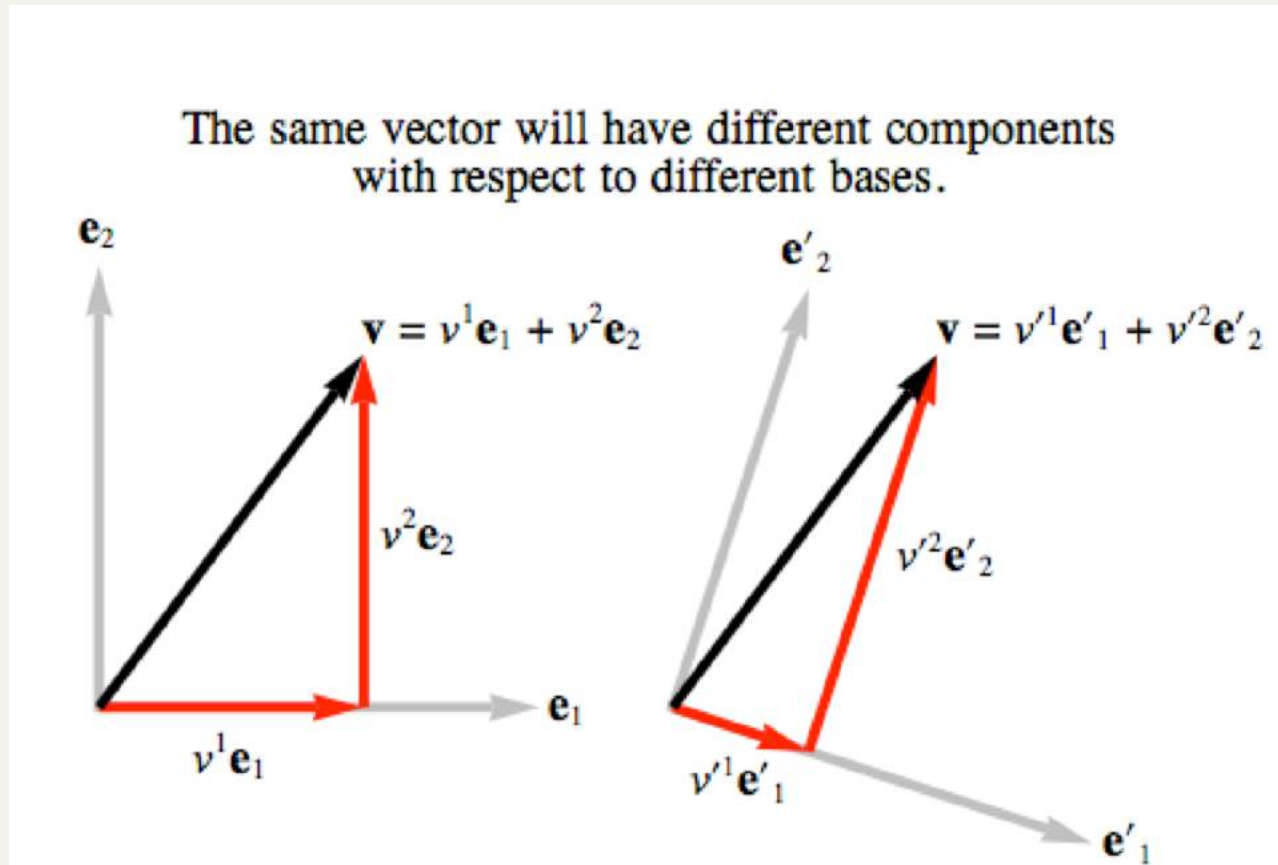
Why eigenvectors are orthonormal (if A is symmetric square matrix)

Let $\mathbf{v}_1, \mathbf{v}_2$ be two eigenvectors, and let λ_1 be the eigenvalue of \mathbf{v}_1 , then we have:

$$\begin{aligned}\lambda_1(\mathbf{v}_2 \cdot \mathbf{v}_1) &= \lambda_1(\mathbf{v}_2^T \mathbf{v}_1) = (\mathbf{v}_2^T \lambda_1 \mathbf{v}_1) = (\mathbf{v}_2^T A) \mathbf{v}_1 = (A^T \mathbf{v}_2)^T \mathbf{v}_1 \\ &= (A \mathbf{v}_2)^T \mathbf{v}_1 = \lambda_2 \mathbf{v}_2^T \mathbf{v}_1 = \lambda_2(\mathbf{v}_2 \cdot \mathbf{v}_1) \\ &\Rightarrow (\mathbf{v}_2 \cdot \mathbf{v}_1) = 0 \text{ and } \mathbf{v}_2 \perp \mathbf{v}_1\end{aligned}$$

Either $\lambda_1 = \lambda_2$ or $(\mathbf{v}_1 \cdot \mathbf{v}_2) = 0$!

Example: projecting a vector on 2 orthonormal spaces (or “bases”)



e_1, e'_1, e_2, e'_2 are **unary** vectors and v_1, v'_1, v_2, v'_2 are the coordinates of v along the directions of e_1, e'_1, e_2, e'_2

The effect of a matrix-vector multiplication is governed by eigenvectors and eigenvalues

Let \vec{x} be a generic vector and A a normal matrix

- $A \cdot \vec{x} = A(x_1 \vec{e}_1 + x_2 \vec{e}_2 + x_3 \vec{e}_3)$ where x_i are the vector coordinates in the base defined by unary vectors \vec{e}_i

- Let's now project the very same vector \vec{x} on the base defined by 3 eigenvectors of matrix A:

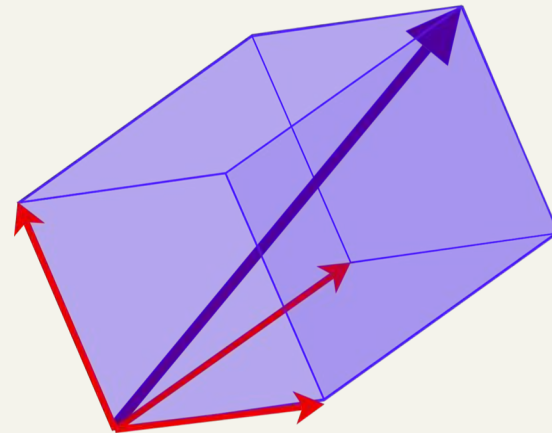
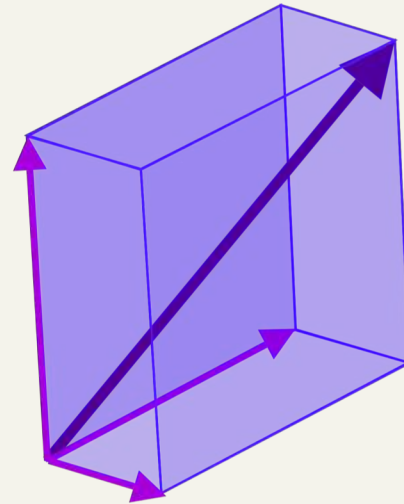
$$\vec{x} = x'_1 \vec{e}'_1 + x'_2 \vec{e}'_2 + x'_3 \vec{e}'_3 = \frac{x'_1}{|\vec{v}_1|} \vec{v}_1 + \frac{x'_2}{|\vec{v}_2|} \vec{v}_2 + \frac{x'_3}{|\vec{v}_3|} \vec{v}_3$$

- We then have: $x_1 = \frac{x'_1}{|\vec{v}_1|}$, $x_2 = \frac{x'_2}{|\vec{v}_2|}$, $x_3 = \frac{x'_3}{|\vec{v}_3|}$
- $A(x_1 \vec{e}_1 + x_2 \vec{e}_2 + x_3 \vec{e}_3) = A(x_1 \vec{v}_1 + x_2 \vec{v}_2 + x_3 \vec{v}_3)$
- $A(x_1 \vec{v}_1 + x_2 \vec{v}_2 + x_3 \vec{v}_3) = x_1 \lambda_1 \vec{v}_1 + x_2 \lambda_2 \vec{v}_2 + x_3 \lambda_3 \vec{v}_3$

The effect of a matrix-vector multiplication is governed by eigenvectors and eigenvalues (2)

$$A \cdot \vec{x} = x_1 \lambda_1 \vec{v}_1 + x_2 \lambda_2 \vec{v}_2 + x_3 \lambda_3 \vec{v}_3$$

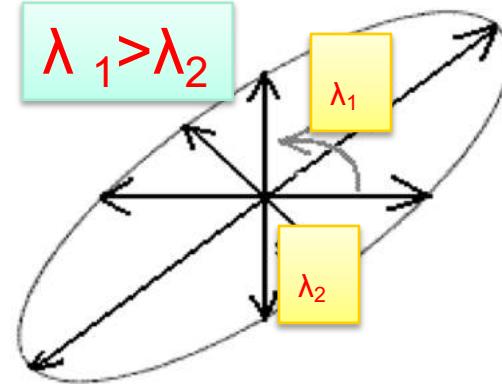
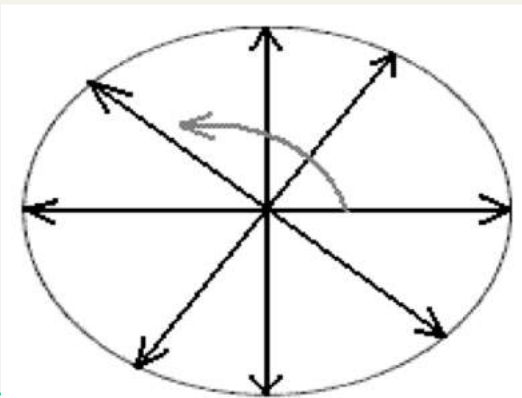
Even though x is an arbitrary vector, the action of A on x (transformation) is determined by the eigenvalues/vectors.



Geometric explanation: largest eigenvalues play the largest role in the “distortion” of the original vector

\mathbf{x} is a generic vector with coordinates x_i ; λ_i, \mathbf{v}_i are the eigenvalues and eigenvectors of A

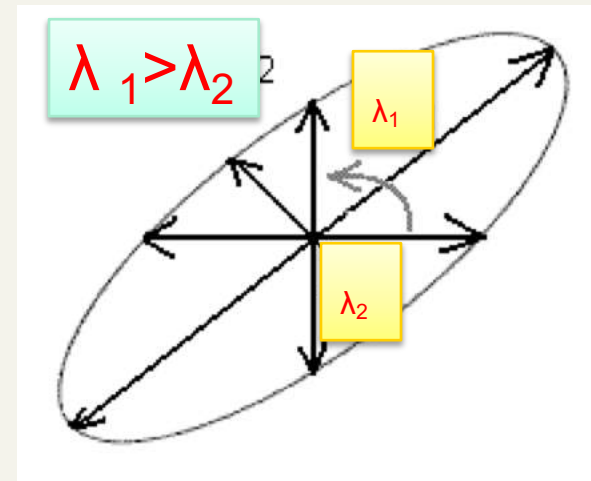
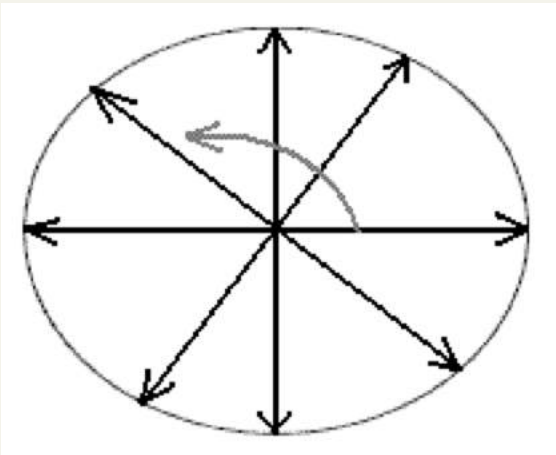
$$A\mathbf{x} = x_1\lambda_1\mathbf{v}_1 + x_2\lambda_2\mathbf{v}_2 + x_3\lambda_3\mathbf{v}_3$$



Multiplying a matrix and a vector has two effects over the vector: **rotation** (the coordinates of the vector change) and **scaling** (the length changes). **The max compression and rotation depends on the largest matrix's eigenvalues λ_i**

Geometric explanation

$$Ax = x_1 \lambda_1 v_1 + x_2 \lambda_2 v_2 + x_3 \lambda_3 v_3$$



In the distortion, the max effect **is played by the biggest eigenvalues** (s1 and s2 in the picture)

The eigenvalues describe the distortion operated by the matrix on the original vector

Summary so far

- A matrix A has **eigenvectors** \mathbf{v} and **eigenvalues** λ , defined by $A\mathbf{v}=\lambda\mathbf{v}$
- Eigenvalues can be computed as:

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

- We can compute only the the **direction** of eigenvectors, since for any eigenvalue there are infinite eigenvectors lying on the same direction
- If A is normal (i.e. if $AA^T=A^T A$) then the eigenvector form an **orthonormal basis**
- The product of A by **ANY vector** \mathbf{x} is a linear transformation of \mathbf{x} where the **rotation** is determined by eigenvectors and the **translation** is determined by the eigenvalues. The biggest role in this transformation is played by the **biggest (principal) eigenvalues**.

Bad news..



More algebra..

Eigen/diagonal Decomposition

- Let A be a **square** matrix with m **orthogonal eigenvectors** (hence, A is normal)

- **Theorem:** Exists an **eigen decomposition**

- $A = U\Lambda U^{-1}$

- Λ is a diagonal matrix (all zero except the diagonal cells)

$$\Lambda = \text{diag}(\lambda_1, \dots, \lambda_m), \quad \lambda_i \geq \lambda_{i+1}$$

- Columns of U are **eigenvectors** of A
- Diagonal elements of Λ are **eigenvalues** of A

Diagonal decomposition: why/how

Let \mathbf{U} have the eigenvectors as columns: $\mathbf{U} = \begin{bmatrix} \mathbf{v}_1 & \dots & \mathbf{v}_n \end{bmatrix}$

Then, \mathbf{AU} can be written

$$\mathbf{AU} = \mathbf{A} \begin{bmatrix} \mathbf{v}_1 & \dots & \mathbf{v}_n \end{bmatrix} = \begin{bmatrix} \lambda_1 \mathbf{v}_1 & \dots & \lambda_n \mathbf{v}_n \end{bmatrix} = \begin{bmatrix} \mathbf{v}_1 & \dots & \mathbf{v}_n \end{bmatrix} \begin{bmatrix} \lambda_1 & & \\ & \dots & \\ & & \lambda_n \end{bmatrix}$$

Thus $\mathbf{AU} = \mathbf{U}\mathbf{\Lambda}$, or $\mathbf{U}^{-1}\mathbf{AU} = \mathbf{\Lambda}$

And $\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{-1}$.

Example

$$A = \begin{bmatrix} 1 & 0 \\ 1 & 3 \end{bmatrix}$$

$$AU = U\Lambda$$


$$A[\mathbf{v}_1 \mathbf{v}_2] = \begin{bmatrix} 1 & 0 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} v_{11} & v_{21} \\ v_{12} & v_{22} \end{bmatrix} = \begin{bmatrix} v_{11} & v_{21} \\ v_{12} & v_{22} \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} = [\mathbf{v}_1 \mathbf{v}_2] \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

From this we compute $\lambda_1=1$, $\lambda_2=3$

$$\begin{bmatrix} 1 & 0 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} v_{11} \\ v_{12} \end{bmatrix} = 1 \begin{bmatrix} v_{11} \\ v_{12} \end{bmatrix}$$

from which we get $v_{11} = -2v_{12}$

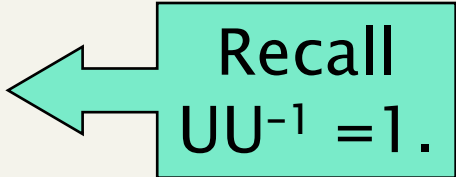
$$\begin{bmatrix} 1 & 0 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} v_{21} \\ v_{22} \end{bmatrix} = 3 \begin{bmatrix} v_{21} \\ v_{22} \end{bmatrix}$$

From which we get $v_{21}=0$ and v_{22} any real

Diagonal decomposition – example 2

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

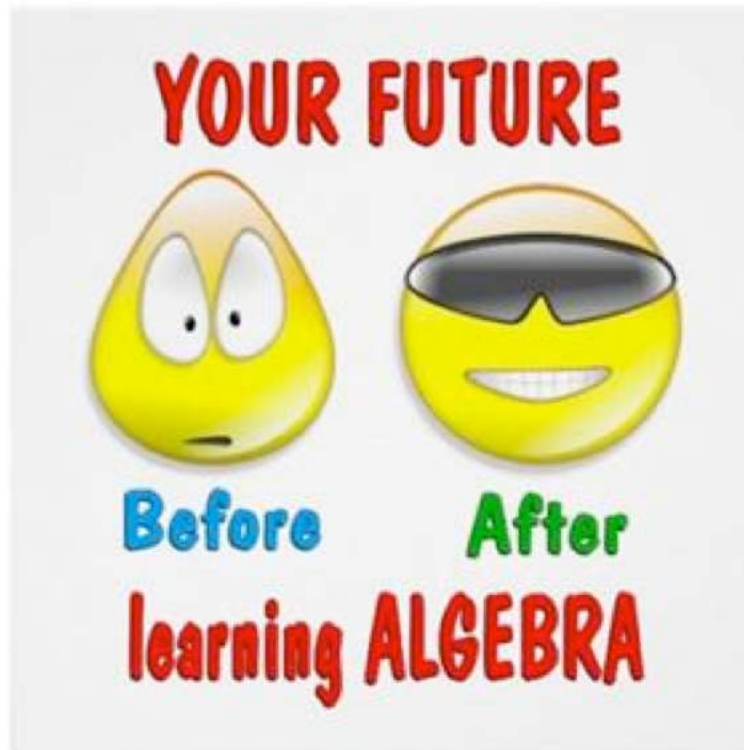
The eigenvectors $\begin{pmatrix} 1 \\ -1 \end{pmatrix}$ and $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$ form $U = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}$

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

Then, $A = U\Lambda U^{-1} = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} 1/2 & -1/2 \\ 1/2 & 1/2 \end{bmatrix}$ 37

So what?

- What do these matrices have to do with Information Retrieval and document ranking?
- Recall $M \times N$ term-document matrices ...
- But everything is **normal** – one last normal and learn



Singular Value Decomposition for non-square matrixes

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

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

$M \times M$ $M \times N$ V is $N \times N$

The columns of \mathbf{U} are the orthogonal eigenvectors of $\mathbf{A}\mathbf{A}^T$ (called **left singular vectors**).

The columns of \mathbf{V} (rows of \mathbf{V}^T) are the orthogonal eigenvectors of $\mathbf{A}^T\mathbf{A}$ (called right singular eigenvector). NOTE THAT $\mathbf{A}\mathbf{A}^T$ and $\mathbf{A}^T\mathbf{A}$ are square symmetric (and hence **NORMAL**)

Eigenvalues $\lambda_1 \dots \lambda_r$ of $\mathbf{A}\mathbf{A}^T =$ eigenvalues of $\mathbf{A}^T\mathbf{A}$ and: $\sigma_i = \sqrt{\lambda_i}$

$$\Sigma = \text{diag}(\sigma_1 \dots \sigma_r) \leftarrow \text{Singular values of } \mathbf{A}$$

An example

Find the SVD of A , $U\Sigma V^T$, where $A = \begin{pmatrix} 3 & 2 & 2 \\ 2 & 3 & -2 \end{pmatrix}$

$$A = U\Sigma V^T = U \begin{pmatrix} 5 & 0 & 0 \\ 0 & 3 & 0 \end{pmatrix} \begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 1/\sqrt{18} & -1/\sqrt{18} & 4/\sqrt{18} \\ 2/3 & -2/3 & -1/3 \end{pmatrix}$$

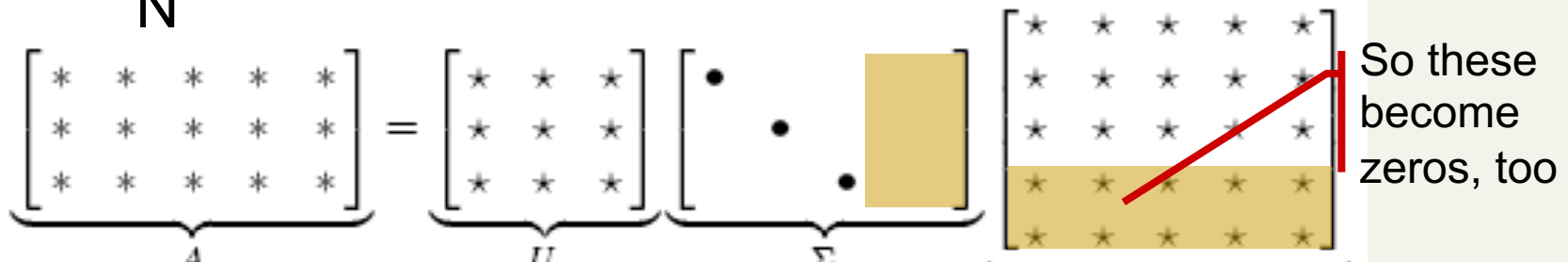
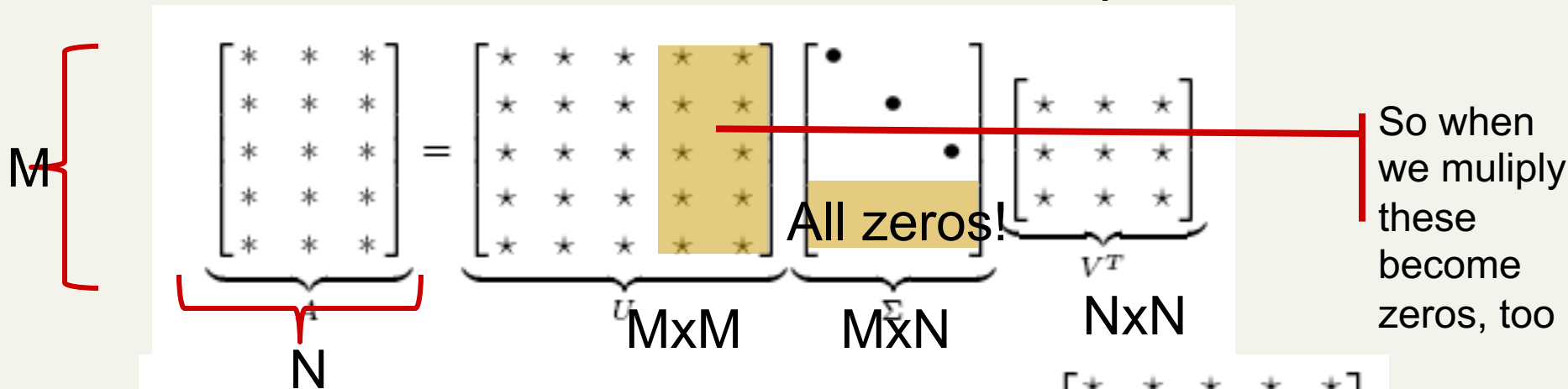
$$A = U\Sigma V^T = \begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{pmatrix} \begin{pmatrix} 5 & 0 & 0 \\ 0 & 3 & 0 \end{pmatrix} \begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 1/\sqrt{18} & -1/\sqrt{18} & 4/\sqrt{18} \\ 2/3 & -2/3 & -1/3 \end{pmatrix}$$

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad \begin{pmatrix} 4/\sqrt{18} \\ 0 \end{pmatrix}$$

$$v_3 = \begin{pmatrix} 2/3 \\ -2/3 \\ -1/3 \end{pmatrix}$$

Singular Value Decomposition

- Illustration of SVD dimensions and sparseness



$$\begin{pmatrix} 3 & 2 & 2 \\ 2 & 3 & -2 \end{pmatrix} = \begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{pmatrix} \begin{pmatrix} 5 & 0 & 0 \\ 0 & 3 & 0 \end{pmatrix} \begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 1/\sqrt{18} & -1/\sqrt{18} & 4/\sqrt{18} \\ 2/3 & -2/3 & -1/3 \end{pmatrix}$$

Back to matrix-vector multiplication

- Remember what we said? In a matrix vector multiplication the biggest role is played by the biggest eigenvalues
- The diagonal matrix Σ has the eigenvalues of $A^T A$ (called the *singular values* σ of A) in decreasing order along the diagonal
- We can therefore apply an **approximation** by setting $\sigma_i=0$ if $\sigma_i \leq \theta$ and only consider only the first k singular values

Reduced SVD

- If we retain only k highest singular values, and set the rest to 0, then **we don't need the matrix parts in red**
- 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, or rank k approximation**

Now all the red and yellow parts are zeros!!

$$\begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{bmatrix} = \begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \end{bmatrix} \begin{bmatrix} \bullet & & \\ & k & \\ & & \bullet \end{bmatrix} \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{bmatrix}$$

The diagram shows the decomposition of matrix A into U , Σ , and V^T . Matrix A is a 3x5 grid of stars. Matrix U is a 3x3 grid of stars with a red vertical bar on the right. Matrix Σ is a 3x3 grid with two dots on the diagonal, a red 'k' above the first dot, a red horizontal bar on the bottom row, and a yellow vertical bar on the right. Matrix V^T is a 3x5 grid of stars with a red horizontal bar on the top row and a yellow vertical bar on the right. Brackets below each matrix label them as A , U , Σ , and V^T respectively.

Let's recap

$M < N$

$$\underbrace{\begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{bmatrix}}_A = \underbrace{\begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \end{bmatrix}}_U \underbrace{\begin{bmatrix} \bullet & & & & \\ & \bullet & & & \\ & & \bullet & & \\ & & & \text{yellow} & \\ & & & & \text{yellow} \end{bmatrix}}_\Sigma \underbrace{\begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{bmatrix}}_{V^T}$$

$M > N$

$$\underbrace{\begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \\ * & * & * \\ * & * & * \end{bmatrix}}_A = \underbrace{\begin{bmatrix} * & * & * & \text{yellow} & \text{yellow} \\ * & * & * & \text{yellow} & \text{yellow} \\ * & * & * & \text{yellow} & \text{yellow} \\ * & * & * & \text{yellow} & \text{yellow} \\ * & * & * & \text{yellow} & \text{yellow} \end{bmatrix}}_U \underbrace{\begin{bmatrix} \bullet & & \\ & \bullet & \\ & & \bullet \\ & & & \text{yellow} \\ & & & & \text{yellow} \end{bmatrix}}_\Sigma \underbrace{\begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \end{bmatrix}}_{V^T}$$

Since the yellow part is zero, an **exact** representation of A is:

$$A = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \dots + \sigma_r u_r v_r^T$$

$$r = \min(M, N)$$

But “for some” $k < r$, a **good approximation** is:

$$A_k = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \dots + \sigma_k u_k v_k^T$$

Example of rank approximation

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 2 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 4 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} 4 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & \sqrt{5} & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ \sqrt{0.2} & 0 & 0 & 0 & \sqrt{0.8} \\ 0 & 0 & 0 & 1 & 0 \\ -\sqrt{0.8} & 0 & 0 & 0 & \sqrt{0.2} \end{bmatrix}$$

A

$$[A]^* = \begin{bmatrix} 001 \\ 010 \\ 000 \\ 100 \end{bmatrix} \times \begin{bmatrix} 400 \\ 030 \\ 00\sqrt{5} \end{bmatrix} \times \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ \sqrt{0.2} & 0 & 0 & 0 & \sqrt{0.8} \end{bmatrix}$$

$$A^* = \begin{bmatrix} 0.981 & 0.000 & 0.000 & 0.000 & 1.985 \\ 0.000 & 0.000 & 3.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 4.000 & 0.000 & 0.000 & 0.000 \end{bmatrix} \cong A$$

Approximation error

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

$$\min_{X:\text{rank}(X)=k} \|A - X\|_F = \|A - A_k\|_F = \sigma_{k+1} \quad \sigma_i = \sqrt{\lambda_i}$$

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

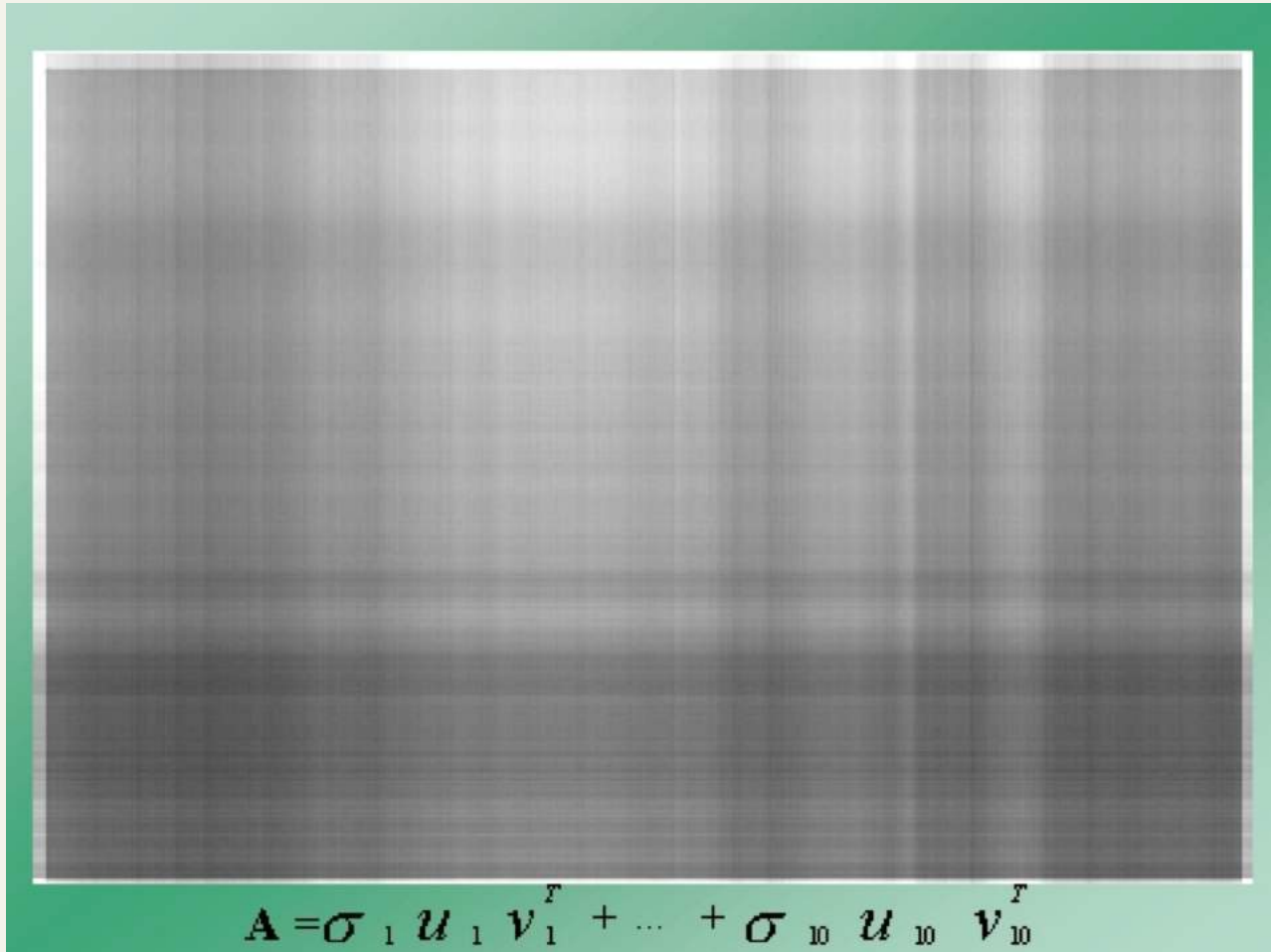
- Suggests why Frobenius error drops as k increases.

Images gives a better intuition (image = matrix of pixels)



The original image

K=10



K=20



$$\mathbf{A} = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T + \dots + \sigma_{20} \mathbf{u}_{20} \mathbf{v}_{20}^T$$

K=30



$$\mathbf{A} = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T + \dots + \sigma_{30} \mathbf{u}_{30} \mathbf{v}_{30}^T$$

K=100



$$\mathbf{A} = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T + \dots + \sigma_{100} \mathbf{u}_{100} \mathbf{v}_{100}^T$$

K=322 (the original image)



The original image

We obtained our approximation by summing up only the first 100 terms of the singular value decomposition.

This approximation reduced the amount of information in our image by nearly **70% !!!**

We save space!! But this is only one issue

So, finally, back to IR!!!

- Our initial problem was:
 - the term-document ($M \times N$) matrix A is highly sparse (has many zeros)
 - However, since groups of terms tend to co-occur together, can we identify the **semantic space** of these clusters of terms, and apply the vector space model in the semantic space defined by such clusters (rather than the space of terms)?
- What we learned so far:
 - Matrix A can be decomposed, and rank- k approximated using SVD
 - Does this help solving our initial problems?

A is our term document matrix

- **Latent Semantic Indexing via the SVD**

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

The diagram shows the equation $A = U \Sigma V^T$ at the top. Below it are three boxes: the first contains $M \times M$, the second contains $M \times N$, and the third contains $V \text{ is } N \times N$. Arrows point from each box to the corresponding matrix in the equation above.

The columns of \mathbf{U} are orthogonal eigenvectors of \mathbf{AA}^T .

The columns of \mathbf{V} are orthogonal eigenvectors of $\mathbf{A}^T\mathbf{A}$.

Eigenvalues $\lambda_1 \dots \lambda_r$ of \mathbf{AA}^T are the eigenvalues of $\mathbf{A}^T\mathbf{A}$.

- If A is a term/document matrix, then AA^T and $A^T A$ are the (square) matrixes of **term** and **document** co-occurrences, repectively

Meaning of $A^T A$ and $A A^T$

$$L = A A^T = \begin{pmatrix} \vdots & L_{ij} & \vdots \\ \vdots & \vdots & \vdots \end{pmatrix} = \begin{pmatrix} \vdots & A_{ik} & \vdots \\ \vdots & \vdots & \vdots \end{pmatrix} \begin{pmatrix} \vdots & A_{kj} & \vdots \\ \vdots & \vdots & \vdots \end{pmatrix}$$

Word i in doc k

Word j in doc k

$$L_{ij} = \sum_{k=1}^N A_{ik} A_{kj}^T = \sum_{k=1}^N A_{ik} A_{jk}$$

A A^T

L_{ij} depends on the number of documents d_k in which w_i and w_j co-occur (the non-zero products $A_{ik} A_{kj}^T$ of the sum)
Similarly, L^T_{ij} depends on the number of common documents for two word pairs (or vice-versa if A is a document-term matrix rather than term-document)

Example

Example of text data: Titles of Some Technical Memos

- c1: *Human machine interface for ABC computer applications*
- c2: *A survey of user opinion of computer system response time*
- c3: *The EPS user interface management system*
- c4: *System and human system engineering testing of EPS*
- c5: *Relation of user perceived response time to error measurement*

- m1: *The generation of random, binary, ordered trees*
- m2: *The intersection graph of paths in trees*
- m3: *Graph minors IV: Widths of trees and well-quasi-ordering*
- m4: *Graph minors: A survey*

Term co-occurrences example

	c1	c2	c3	c4	c5	m1	m2	m3	m4
human	1	0	0	1	0	0	0	0	0
interface	1	0	1	0	0	0	0	0	0
computer	1	1	0	0	0	0	0	0	0
user	0	1	1	0	1	0	0	0	0
system	0	1	1	2	0	0	0	0	0
response	0	1	0	0	1	0	0	0	0
time	0	1	0	0	1	0	0	0	0
EPS	0	0	1	1	0	0	0	0	0
survey	0	1	0	0	0	0	0	0	1
trees	0	0	0	0	0	1	1	1	0
graph	0	0	0	0	0	0	1	1	1
minors	0	0	0	0	0	0	0	1	1

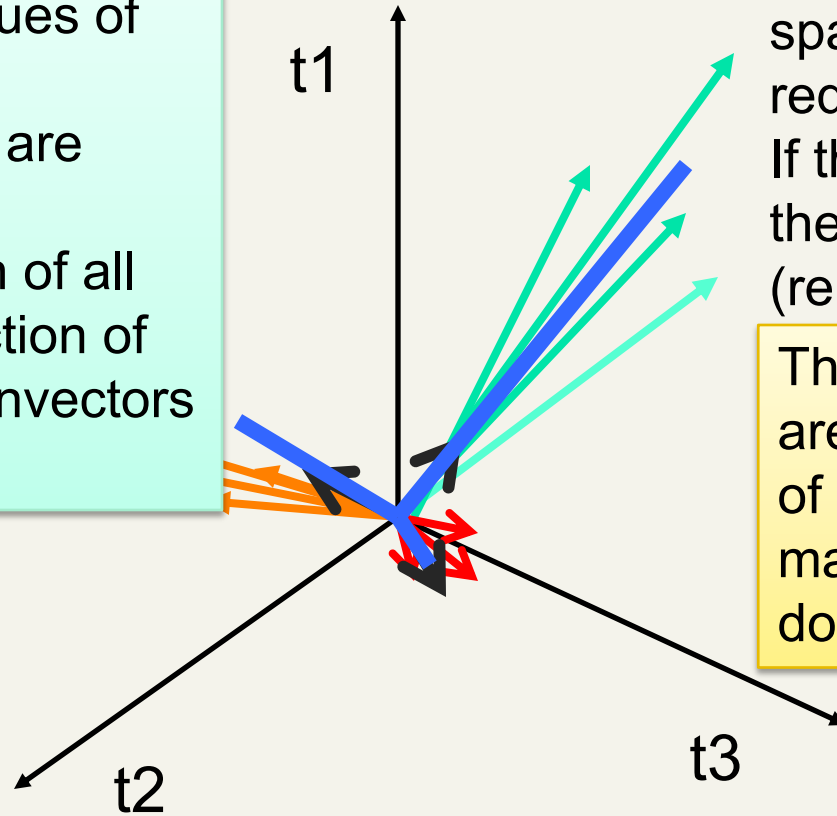
$$L_{\text{trees,graph}} = (000001110) \bullet (000000111)^T = 2$$

So the matrix $L=AA^T$ is the matrix of term co-occurrences in docs

- **Remember:** eigenvectors of a matrix define an orthonormal space
- **Remember:** bigger eigenvalues define the “main” directions of this space
- **But:** Matrixes L and L^T are SIMILARITY (co-occurrence) matrixes (respectively, of terms and of documents). They define a SIMILARITY SPACE (the orthonormal space of their eigenvectors)
- **If the matrix elements are word co-occurrences, bigger eigenvalues are associated to bigger groups of similar words**
- **Similarly, bigger eigenvalues of $L^T=A^T A$ are associated with bigger groups of similar documents (those in which co-occur the same terms)**

LSI: the intuition

The blue segments give the intuition of eigenvalues of $L^T = A^T A$. Bigger eigenvalues are those for which the projection of all vectors on the direction of correspondent eigenvectors is higher.



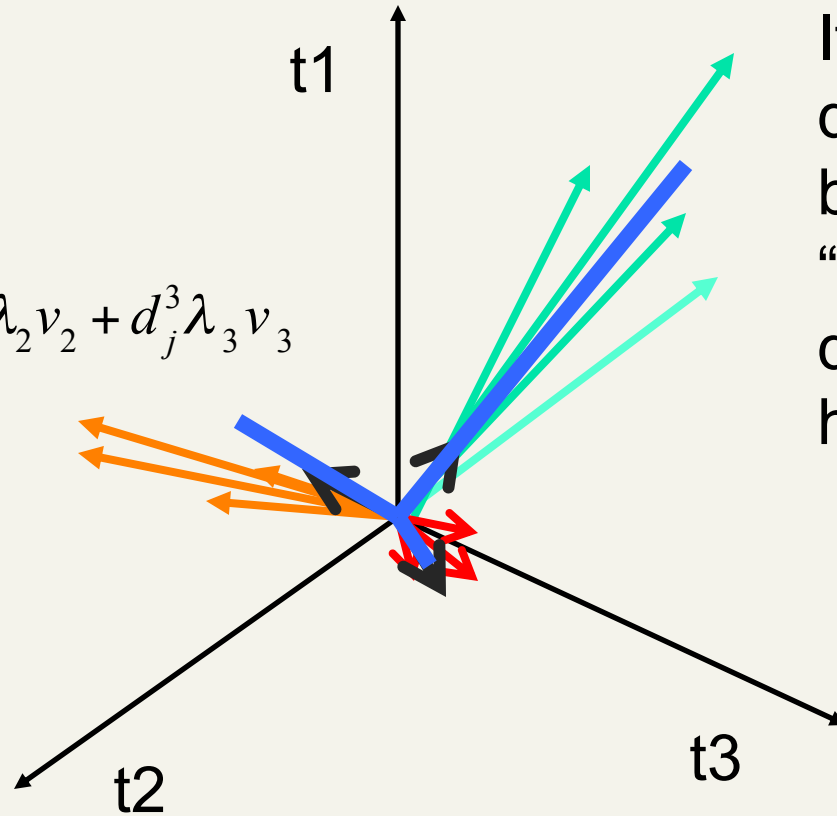
Projecting A in the term space: green, yellow and red vectors are documents. If they form small angles, they have common words (remember cosin-sim)

The black vector are the unary eigenvector of L^T : they represent the main "directions" of the document vectors

LSI intuition

$$\vec{d}_j = (d_j^1, d_j^2, d_j^3)$$

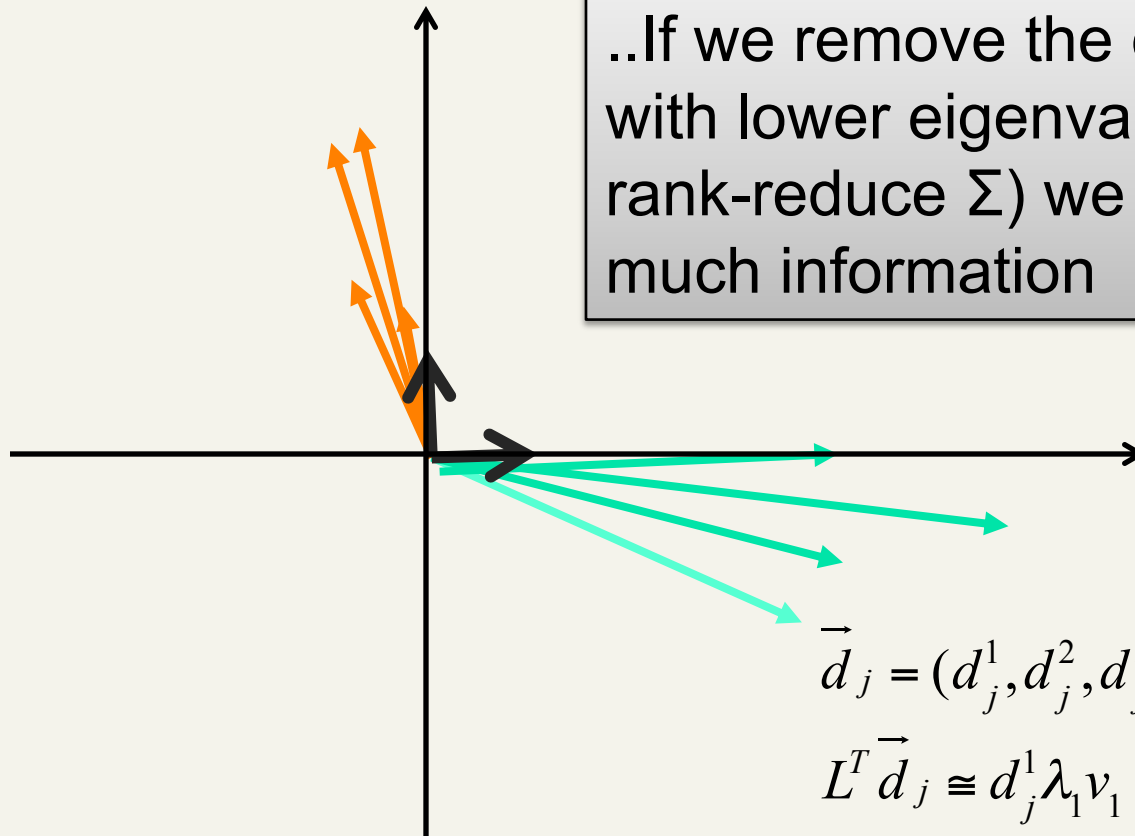
$$L^T \vec{d}_j = d_j^1 \lambda_1 v_1 + d_j^2 \lambda_2 v_2 + d_j^3 \lambda_3 v_3$$



If we multiply all document vectors by $L^T = A^T A$, their “distorsion” is mostly determined by the highest eigenvalues

We now project our document vectors on the reference orthonormal system represented by the 3 black vectors

LSI intuition



$$\vec{d}_j = (d_j^1, d_j^2, d_j^3)$$

$$L^T \vec{d}_j \cong d_j^1 \lambda_1 v_1 + d_j^2 \lambda_2 v_2$$

Remember that the two “new” axis represent a combination of co-occurring words e.g. a **latent semantic space**

Example

	d1	d2	d3	d4
t1	1.000	1.000	0.000	0.000
t2	1.000	0.000	0.000	0.000
t3	0.000	0.000	1.000	1.000
t4	0.000	0.000	1.000	1.000

We project terms and docs on two dimensions, **v1** and **v2** (the principal eigenvectors)

0.000	-0.851	-0.526	0.000	2.000	0.000
0.000	-0.526	0.851	0.000	0.000	1.618
-0.707	0.000	0.000	-0.707	X	
-0.707	0.000	0.000	0.707		

0.000	0.000	-0.707	-0.707	
-0.851	-0.526	0.000	0.000	
X	0.526	-0.851	0.000	0.000
	0.000	0.000	-0.707	0.707

Note that the direction of each eigenvector is determined by the direction of just two terms: (t1,t2) or (t3,t4)

matrix

0.000	0.000	1.000	1.000
0.000	0.000	1.000	1.000

coordinates. s1:(t1,t2) and s2:(t3,t4)

Even if t2 does not occur in d2, now if we query with t2 the system will return also d2!!

Co-occurrence space

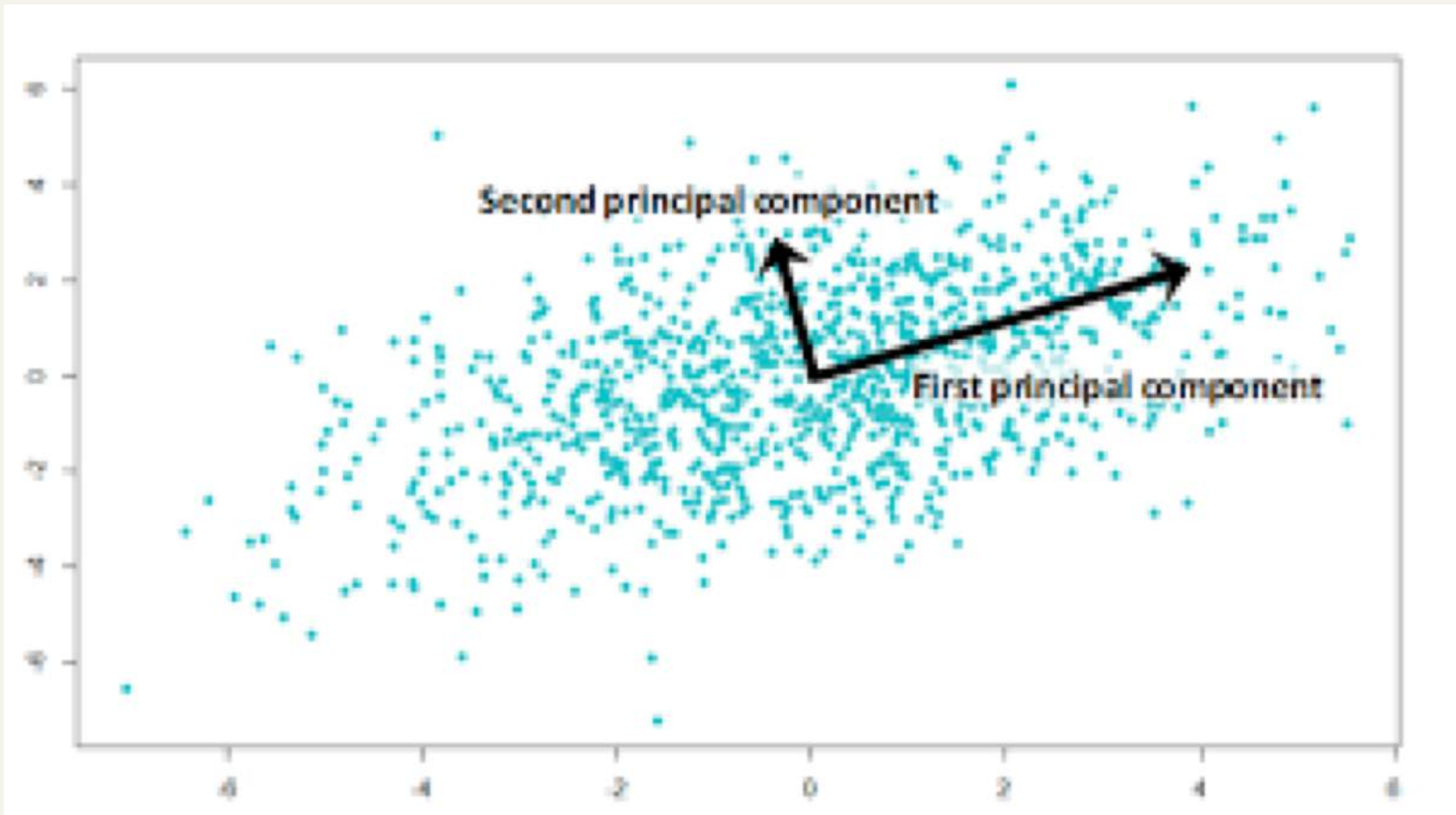
- In principle, the space of document or term co-occurrences is much (much!) higher than the original space of terms!!
- But with SVD we consider only the most relevant ones, through **rank reduction**

$$A = U \Sigma V^T \cong U_k \Sigma_k V_k^T = A_k$$

Summary so-far

- We compute the SVD rank-k approximation for the term-document matrix A
- This approximation is based on considering only the *principal eigenvalues* of the term co-occurrence and document similarity matrixes ($L=AA^T$ and $L^T=A^T A$)
- The eigenvectors of the eigenvalues of $L=AA^T$ and $L^T=A^T A$ represent the main “*directions*”(principal components) identified by term vectors and document vectors, respectively.

Principal components



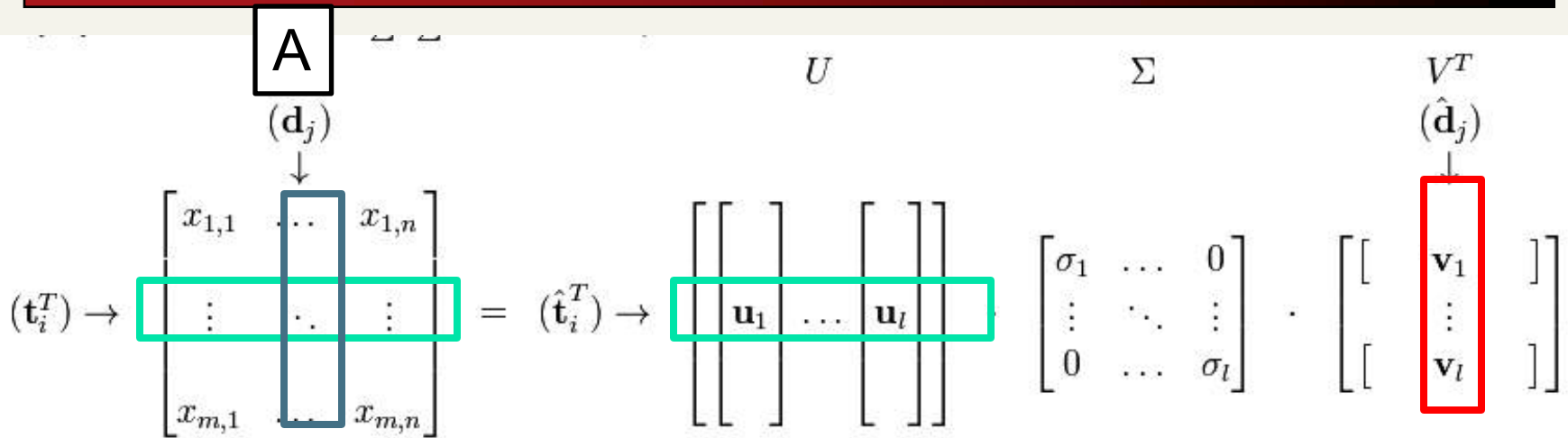
LSI: what are the steps

- From term-doc matrix A , compute the rank- k approximation A_k with SVD
- **Project docs and queries** in a reduced space of $k \ll r$ dimensions (the k “survived” eigenvectors) and compute cos-similarity as usual
- These dimensions are **not** the original axes (terms), but those defined by the orthonormal space of the reduced matrix A_k

$$A\vec{q} \cong A_k\vec{q} = \sigma_1 q_1 \vec{v}_1 + \sigma_2 q_2 \vec{v}_2 + \dots \sigma_k q_k \vec{v}_k$$

Where $\sigma_i q_i$ ($i=1,2..k \ll r$) are the new coordinates of q in the orthonormal space of A_k

Projecting terms documents and queries in the LS space



If $A=U\Sigma V^T$ we also have that:

$$\begin{aligned} V &= A^T U \Sigma^{-1} \\ t &= t'^T \Sigma V^T \\ d &= d'^T U \Sigma^{-1} \\ q &= q'^T U \Sigma^{-1} \end{aligned}$$

After rank k-approximation :

$$\begin{aligned} A &\approx A_k = U_k \Sigma_k V_k^T \\ d_k &\approx d^T U_k \Sigma_k^{-1} \\ q_k &\approx q^T U_k \Sigma_k^{-1} \\ \text{sim}(q, d) &= \\ &\text{sim}(q^T U_k \Sigma_k^{-1}, \\ &d^T U_k \Sigma_k^{-1}) \end{aligned}$$

Consider a term-doc matrix $M \times N$
($M=11$, $N=3$) and a query q

Terms	d1	d2	d3	query
a	1	1	1	0
arrived	0	1	1	0
damaged	1	0	0	0
delivery	0	1	0	0
fire	1	0	0	0
gold	1	0	1	1
in	1	1	1	0
of	1	1	1	0
shipment	1	0	1	0
silver	0	2	0	1
truck	0	1	1	1

1. Compute SVD: $A = U\Sigma V^T$

$$U = \begin{bmatrix} -0.4201 & 0.0748 & -0.0460 \\ -0.2995 & -0.2001 & 0.4078 \\ -0.1206 & 0.2749 & -0.4538 \\ -0.1576 & -0.3046 & -0.2006 \\ -0.1206 & 0.2749 & -0.4538 \\ -0.2626 & 0.3794 & 0.1547 \\ -0.4201 & 0.0748 & -0.0460 \\ -0.4201 & 0.0748 & -0.0460 \\ -0.2626 & 0.3794 & 0.1547 \\ -0.3151 & -0.6093 & -0.4013 \\ -0.2995 & -0.2001 & 0.4078 \end{bmatrix}$$

$$S = \begin{bmatrix} 4.0989 & 0.0000 & 0.0000 \\ 0.0000 & 2.3616 & 0.0000 \\ 0.0000 & 0.0000 & 1.2737 \end{bmatrix}$$

$$V = \begin{bmatrix} -0.4945 & 0.6492 & -0.5780 \\ -0.6458 & -0.7194 & -0.2556 \\ -0.5817 & 0.2469 & 0.7750 \end{bmatrix}$$

$$V^T = \begin{bmatrix} -0.4945 & -0.6458 & -0.5817 \\ 0.6492 & -0.7194 & 0.2469 \\ -0.5780 & -0.2556 & 0.7750 \end{bmatrix}$$

2. Obtain a low rank approximation

$$(k=2) A_k = U_k \Sigma_k V_k^T$$

$$U = \begin{bmatrix} -0.4201 & 0.0748 \\ -0.2995 & -0.2001 \\ -0.1206 & 0.2749 \\ -0.1576 & -0.3046 \\ -0.1206 & 0.2749 \\ -0.2626 & 0.3794 \\ -0.4201 & 0.0748 \\ -0.4201 & 0.0748 \\ -0.2626 & 0.3794 \\ -0.3151 & -0.6093 \\ -0.2995 & -0.2001 \end{bmatrix}$$

$$S = \begin{bmatrix} 4.0989 & 0.0000 \\ 0.0000 & 2.3616 \end{bmatrix}$$

“latent” 2-dimensional
term- similarity space

$$V = \begin{bmatrix} -0.4945 & 0.6492 \\ -0.6458 & -0.7194 \\ -0.5817 & 0.2469 \end{bmatrix}$$

$$V^T = \begin{bmatrix} -0.4945 & -0.6458 & -0.5817 \\ 0.6492 & -0.7194 & 0.2469 \end{bmatrix}$$

“latent” document-
similarity space

3a. Compute doc/query similarity

- For N documents, A_k has N columns, each representing the coordinates of a document d_i projected in the k LSI dimensions
- A query is considered like a document, and is projected in the LSI space

3c. Compute the query vector

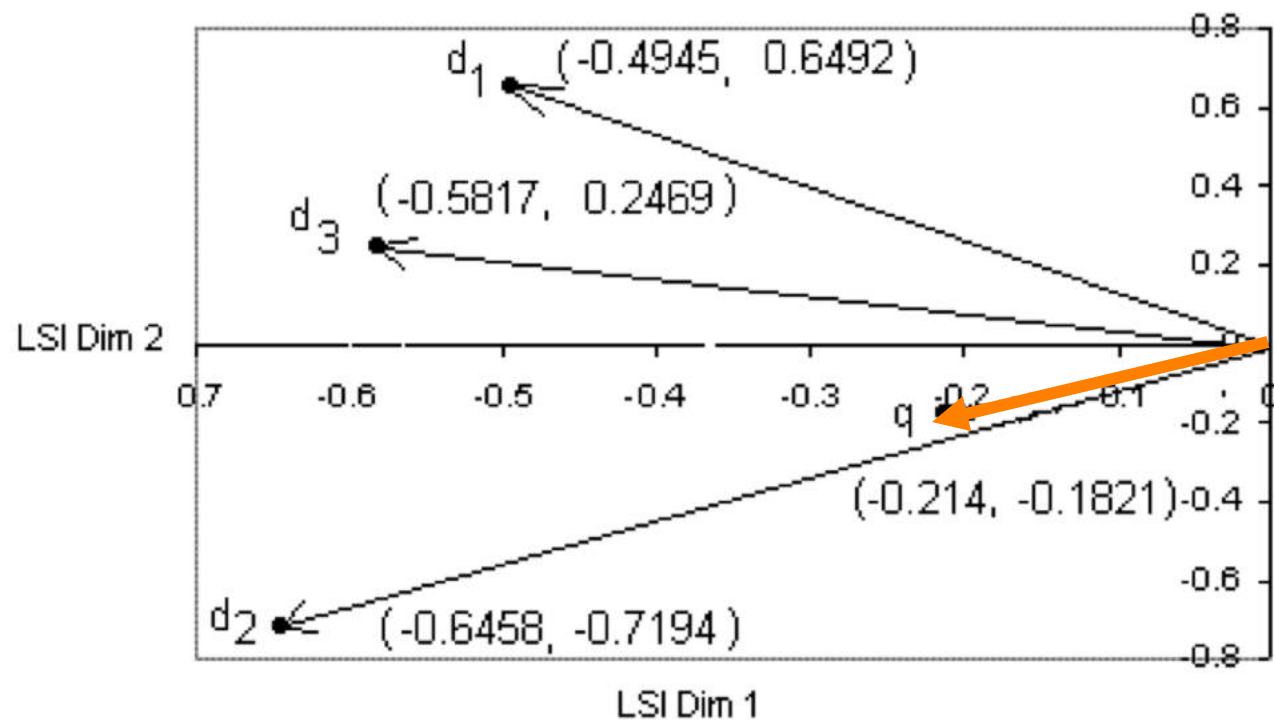
$$q_k = q^T U_k \Sigma_k^{-1}$$

$$q = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} -0.4201 & 0.0748 \\ -0.2995 & -0.2001 \\ -0.1206 & 0.2749 \\ -0.1576 & -0.3046 \\ -0.1206 & 0.2749 \\ -0.2626 & 0.3794 \\ -0.4201 & 0.0748 \\ -0.4201 & 0.0748 \\ -0.2626 & 0.3794 \\ -0.3151 & -0.6093 \\ -0.2995 & -0.2001 \end{bmatrix} \begin{bmatrix} \frac{1}{4.0989} & 0.0000 \\ 0.0000 & \frac{1}{2.3616} \end{bmatrix}$$

$$q = \begin{bmatrix} -0.2140 & -0.1821 \end{bmatrix}$$

q is projected in the 2-dimension LSI space!

Documents and queries projected in the LSI space



q/d similarity

$$\text{sim}(q, d) = \frac{q \bullet d}{\|q\| \|d\|}$$

$$\text{sim}(q, d_1) = \frac{(-0.2140)(-0.4945) + (-0.1821)(0.6492)}{\sqrt{(-0.2140)^2 + (-0.1821)^2} \sqrt{(-0.4945)^2 + (0.6492)^2}} = -0.0541$$

$$\text{sim}(q, d_2) = \frac{(-0.2140)(-0.6458) + (-0.1821)(-0.7194)}{\sqrt{(-0.2140)^2 + (-0.1821)^2} \sqrt{(-0.6458)^2 + (-0.7194)^2}} = 0.9910$$

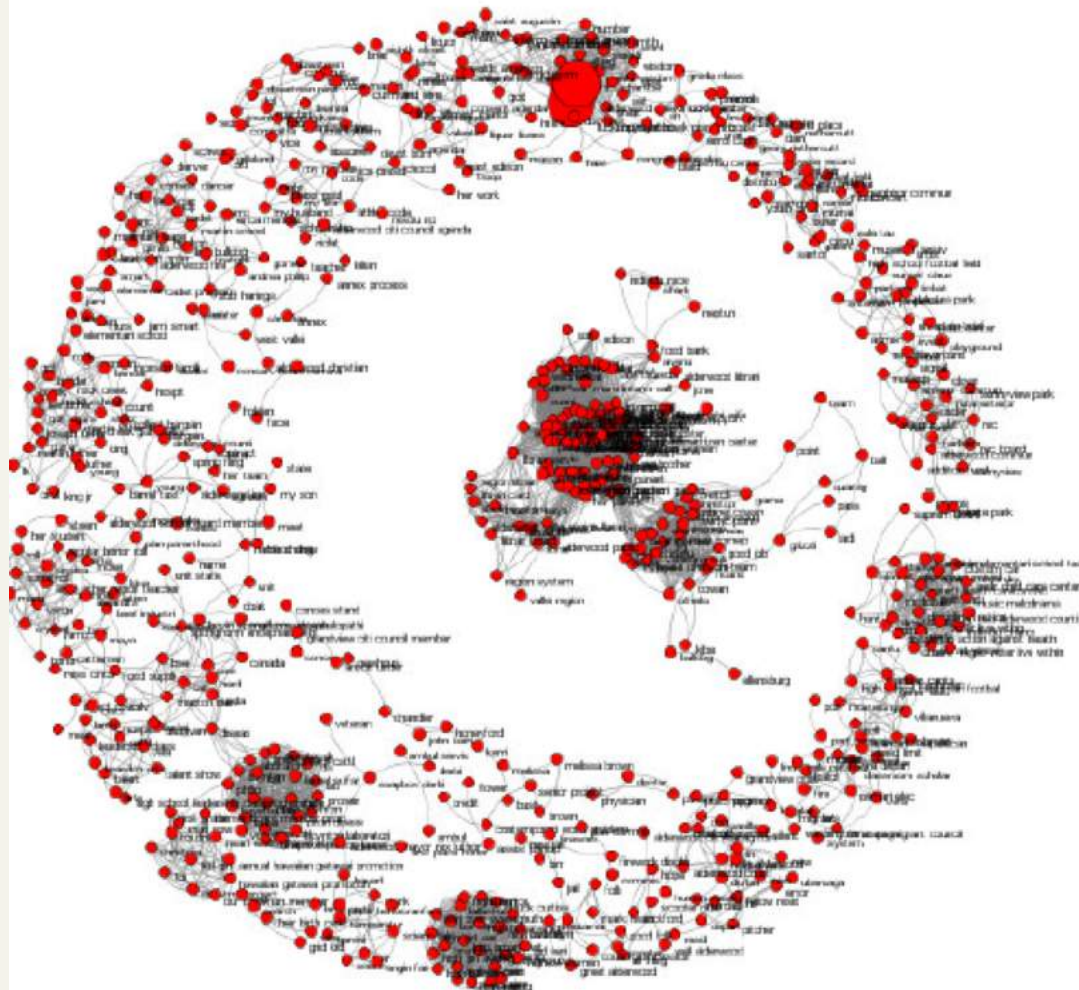
$$\text{sim}(q, d_3) = \frac{(-0.2140)(-0.5817) + (-0.1821)(0.2469)}{\sqrt{(-0.2140)^2 + (-0.1821)^2} \sqrt{(-0.5817)^2 + (0.2469)^2}} = 0.4478$$

Ranking documents in descending order

$$d_2 > d_3 > d_1$$

An overview of a semantic network of terms based on the top 100 most significant latent semantic dimensions (Zhu&Chen)

Semantic Network View of Themes in Latent Concept Dimensions



Conclusion

- LSI performs 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 between words** (latent semantic space).
 - Compute document similarity based on the **cos-sim** in this **latent semantic space**

Another LSI Example

Input matrix A:

	d1	d2	d3
t1	1.000	1.000	0.000
t2	1.000	1.000	0.000
t3	0.000	1.000	0.000
t4	0.000	0.000	1.000

Input matrix B:

1.000	1.000	0.000	0.000
1.000	1.000	1.000	0.000
0.000	0.000	0.000	1.000

AA^T

Matrix product A*B

t1	2.000	2.000	1.000	0.000
t2	2.000	2.000	1.000	0.000
t3	1.000	1.000	1.000	0.000
t4	0.000	0.000	0.000	1.000

Term co-occurrences

Input matrix:

```
1.000 1.000 0.000
1.000 1.000 0.000
0.000 1.000 0.000
0.000 0.000 1.000
```

Singular Value Decomposition:

U:

```
-0.657 0.000 0.261
-0.657 0.000 0.261
-0.369 0.000 -0.929
0.000 -1.000 0.000
```

S:

```
2.136 0.000 0.000
0.000 1.000 0.000
0.000 0.000 0.662
```

V^T

```
-0.615 -0.788 0.000
0.000 0.000 -1.000
0.788 -0.615 0.000
```

$$A_k = U_k \Sigma_k V_k^T \approx A$$

```
0.863 1.106 0.000
0.863 1.106 0.000
0.485 0.621 0.000
0.000 0.000 1.000
```

Now it is like if t3 belongs to d1!

Problems with SVD

- Computational cost **scales quadratically for $n \times m$** matrix: $O(mn^2)$ flops (when $n < m$)
- Hard to incorporate new words or documents
- Does not consider order of words
- Anything better?
- (note that there are a variety of methods similar to SVD, see “principal component analysis”, based on same principles - finding the “main directions” of a set of vectors in a multi-dimensional space)

Is there anything more advanced than co-occurrences to learn correlations?

- Traditional IR uses Term matching, → # of times the doc says “Albuquerque” – not fully appropriate
- We can use a **different approach**: compare all-pairs of query-document **terms**, → # of terms in the doc that relate to Albuquerque
- To detect these similarities (next lessons):
 - Latent Semantic Indexing
 - **Word embeddings (a.k.o. deep method)**

Albuquerque is the most populous city in the U.S. state of New Mexico. The high-altitude city serves as the county seat of Bernalillo County, and it is situated in the central part of the state, straddling the Rio Grande. The city population is 557,169 as of the July 1, 2014, population estimate from the United States Census Bureau, and ranks as the 32nd-largest city in the U.S. The Metropolitan Statistical Area (or MSA) has a population of 902,797 according to the United States Census Bureau's most recently available estimate for July 1, 2013.

Passage *about* Albuquerque

Allen suggested that they could program a BASIC interpreter for the device; after a call from Gates claiming to have a working interpreter, MITS requested a demonstration. Since they didn't actually have one, Allen worked on a simulator for the Altair while Gates developed the interpreter. Although they developed the interpreter on a simulator and not the actual device, the interpreter worked flawlessly when they demonstrated the interpreter to MITS in Albuquerque, New Mexico in March 1975; MITS agreed to distribute it, marketing it as Altair BASIC.

Passage not about Albuquerque