Data Mining and Matrices

01 – Introduction

Prof. Dr. Rainer Gemulla

Universität Mannheim

FSS 2018

Outline

1. What is data mining?
2. What is a matrix?
3. Why data mining and matrices?
4. Summary

Outline

What is data mining?

"Data mining is the process of discovering knowledge or patterns from massive amounts of data." (Encyclopedia of Database Systems)

Rock	Gold	Tools	Miners
			
Data	Knowledge	Software	Analysts

Estimated $100 billion industry around managing and analyzing data.

Data, Data everywhere. The Economist, 2010.

What is data mining?

"Data mining is the process of discovering knowledge or patterns from massive amounts of data." (Encyclopedia of Database Systems)

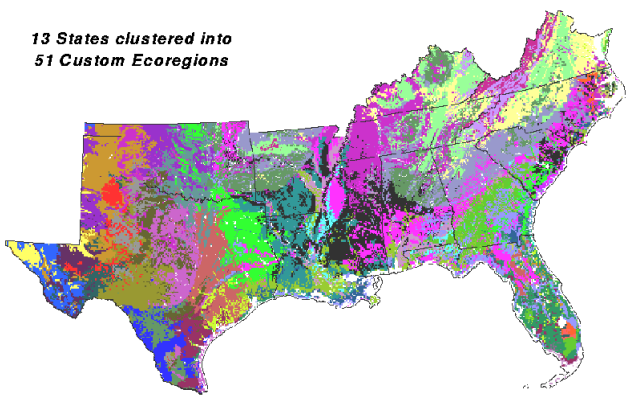
- Science
  - The Sloan Digital Sky Survey gathered 140TB of information
  - NASA Center for Climate Simulation stores 32PB of data
  - 3B base pairs exist in the human genome
  - LHC registers 600M particle collisions per second, 25PB/year
- Social/business data
  - 1M customer transactions are performed at Walmart per hour
  - 4.6B mobile-phone subscriptions worldwide
  - 300M active Twitter users write 500M tweets per day
  - 100M Netflix customers view and rate hundreds of thousands of movies
  - 300h of videos uploaded every minute on YouTube
- Government, health care, news, stocks, books, web search, ...
- Often: less data than in these extreme cases, but still "massive"

Data, Data everywhere. The Economist, 2010.

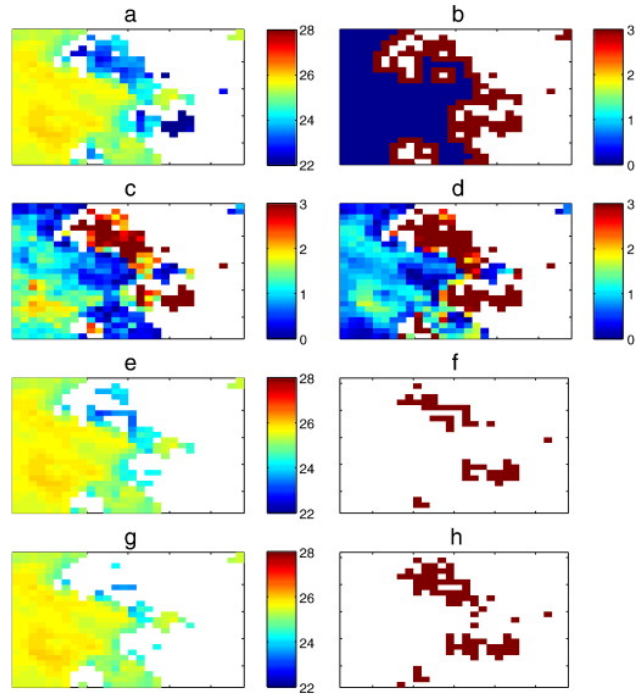
What is data mining?

"Data mining is the process of discovering knowledge or patterns from massive amounts of data." (Encyclopedia of Database Systems)

Prediction



Clustering



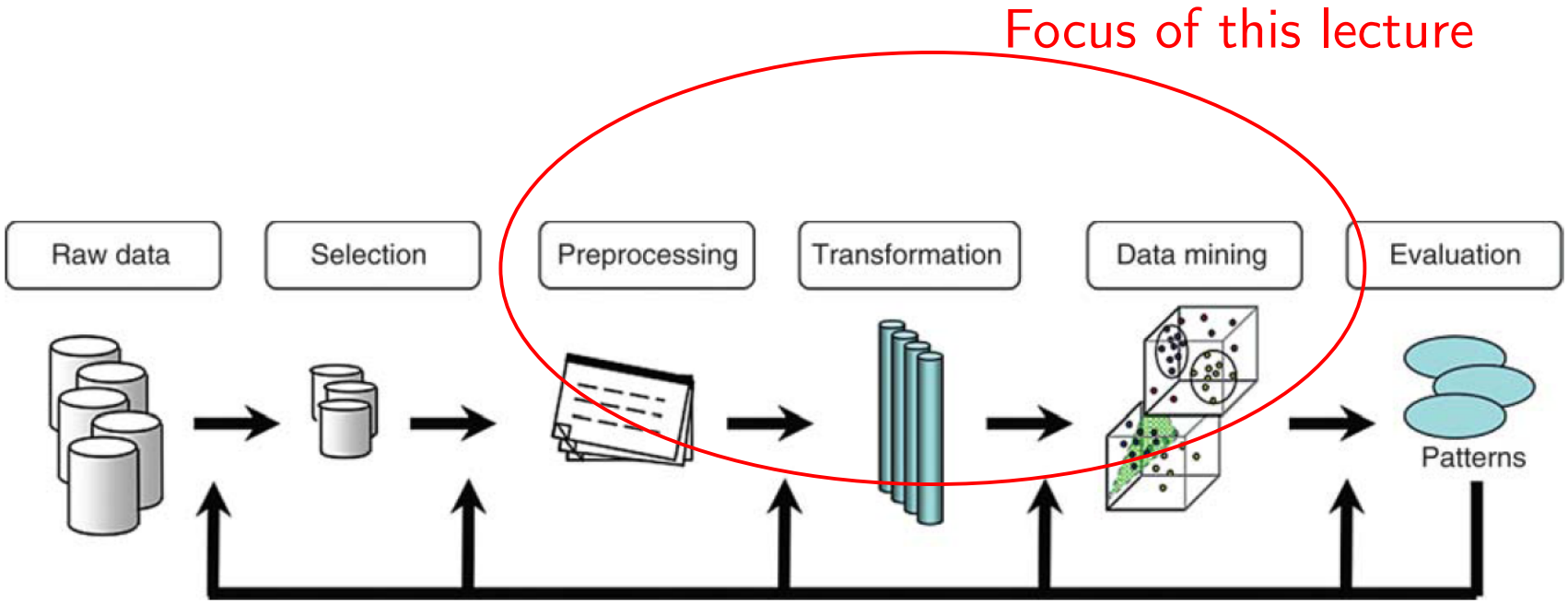
Outlier detection

"Regnet es am Siebenschläfertag, der Regen sieben Wochen nicht weichen mag." (German folklore)

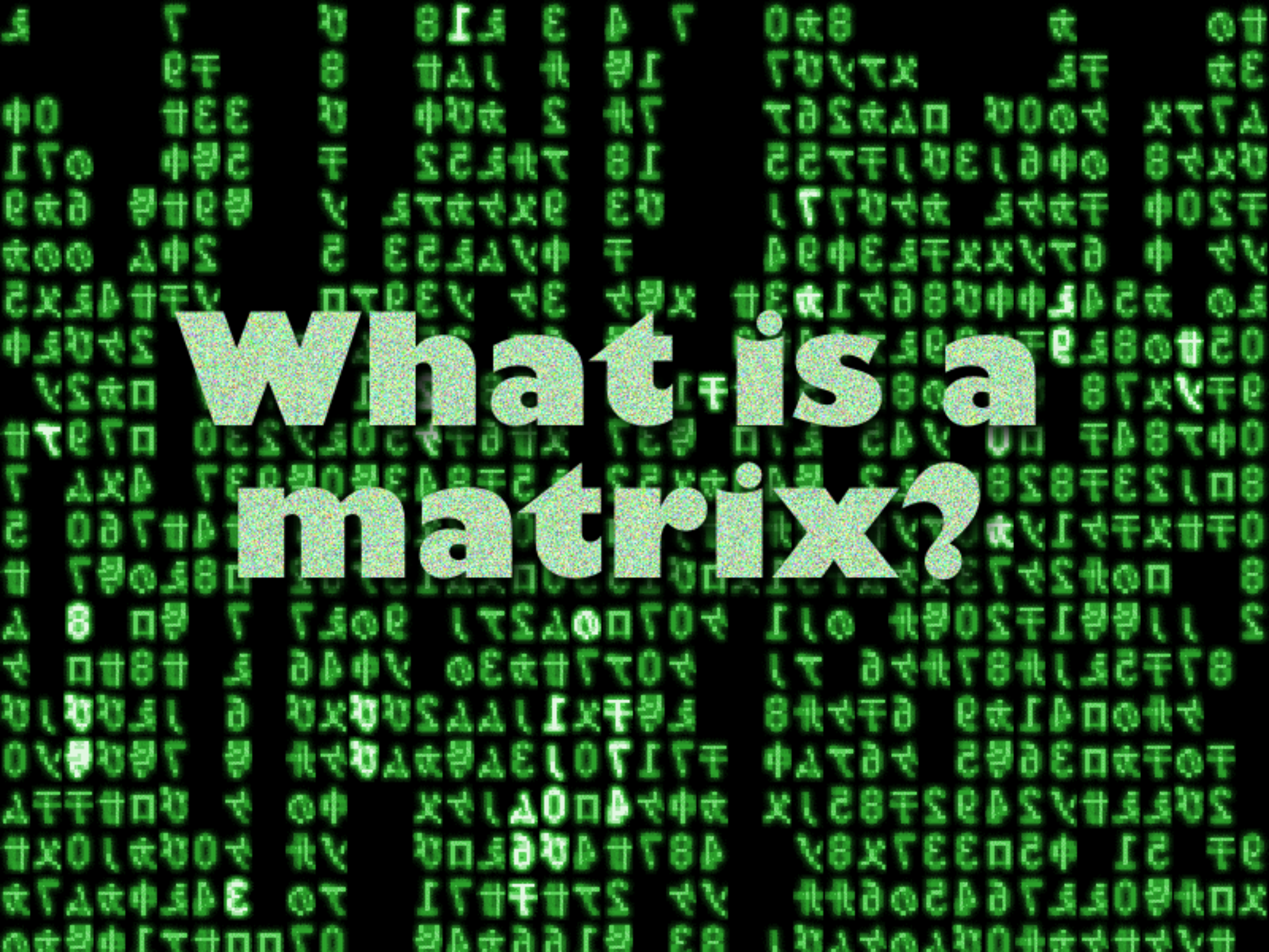
Pattern mining

What is data mining?

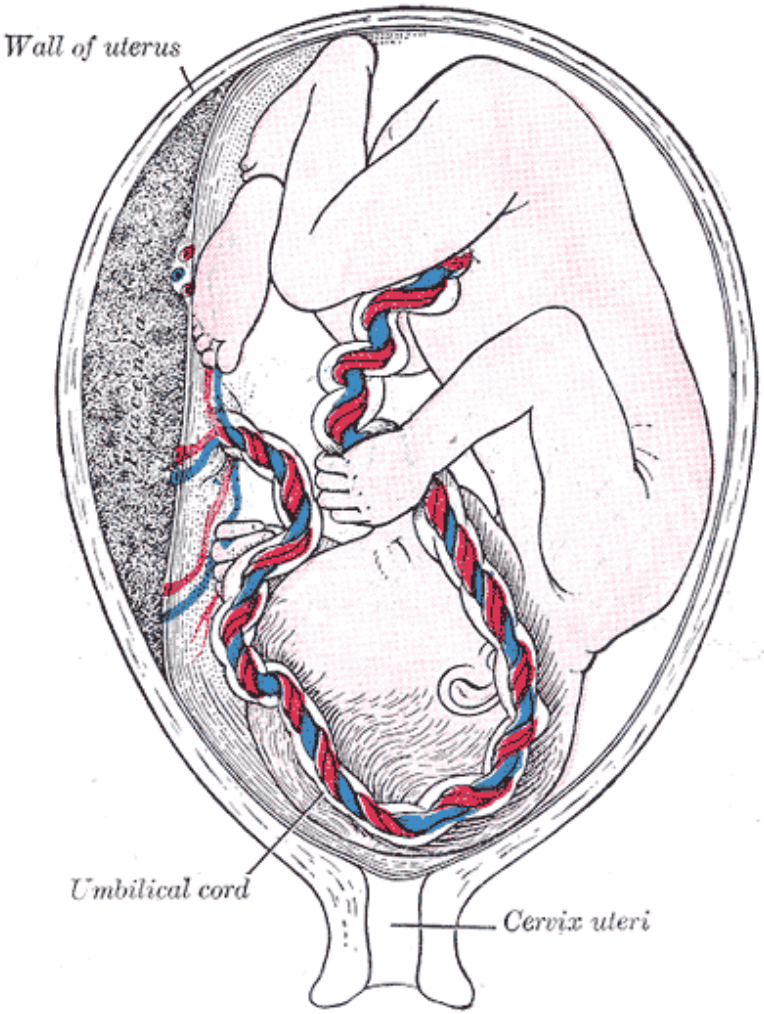
"Data mining is the process of discovering knowledge or patterns from massive amounts of data." (Encyclopedia of Database Systems)



Knowledge discovery process


What is a matrix?

Womb



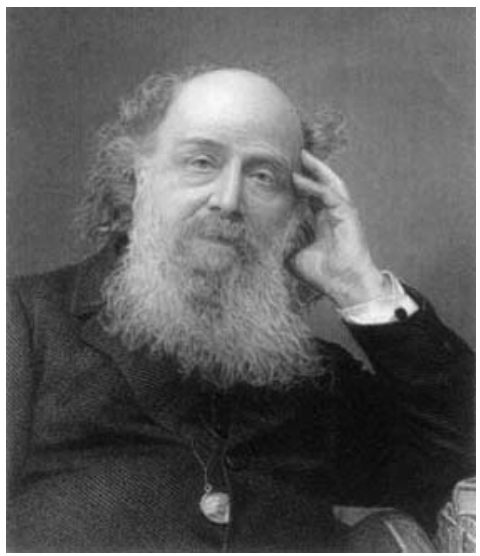
- *mater* (Latin) = *mother*
- *matrix* (Latin) = *pregnant animal*
- *matrix* (Late Latin) = *womb*
  also *source*, *origin*
- Since 1550s: *place or medium where something is developed*
- Since 1640s: *embedding or enclosing mass*

Rectangular arrays of numbers

- "Rectangular arrays" known in ancient China (rod calculus, estimated as early as 300BC)

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

- Term "matrix" coined by J.J. Sylvester in 1850



System of linear equations

Three sheafs of a good crop, two sheafs of a mediocre crop, and one sheaf of a bad crop are sold for 39 dou. Two sheafs of good, three mediocre, and one bad are sold for 34 dou; and one good, two mediocre, and three bad are sold for 26 dou. What is the price received for each sheaf of a good crop, each sheaf of a mediocre crop, and each sheaf of a bad crop?
Chiu-chang Suan-shu (Nine Chapters on Arithmetic), $\approx$ 200BC

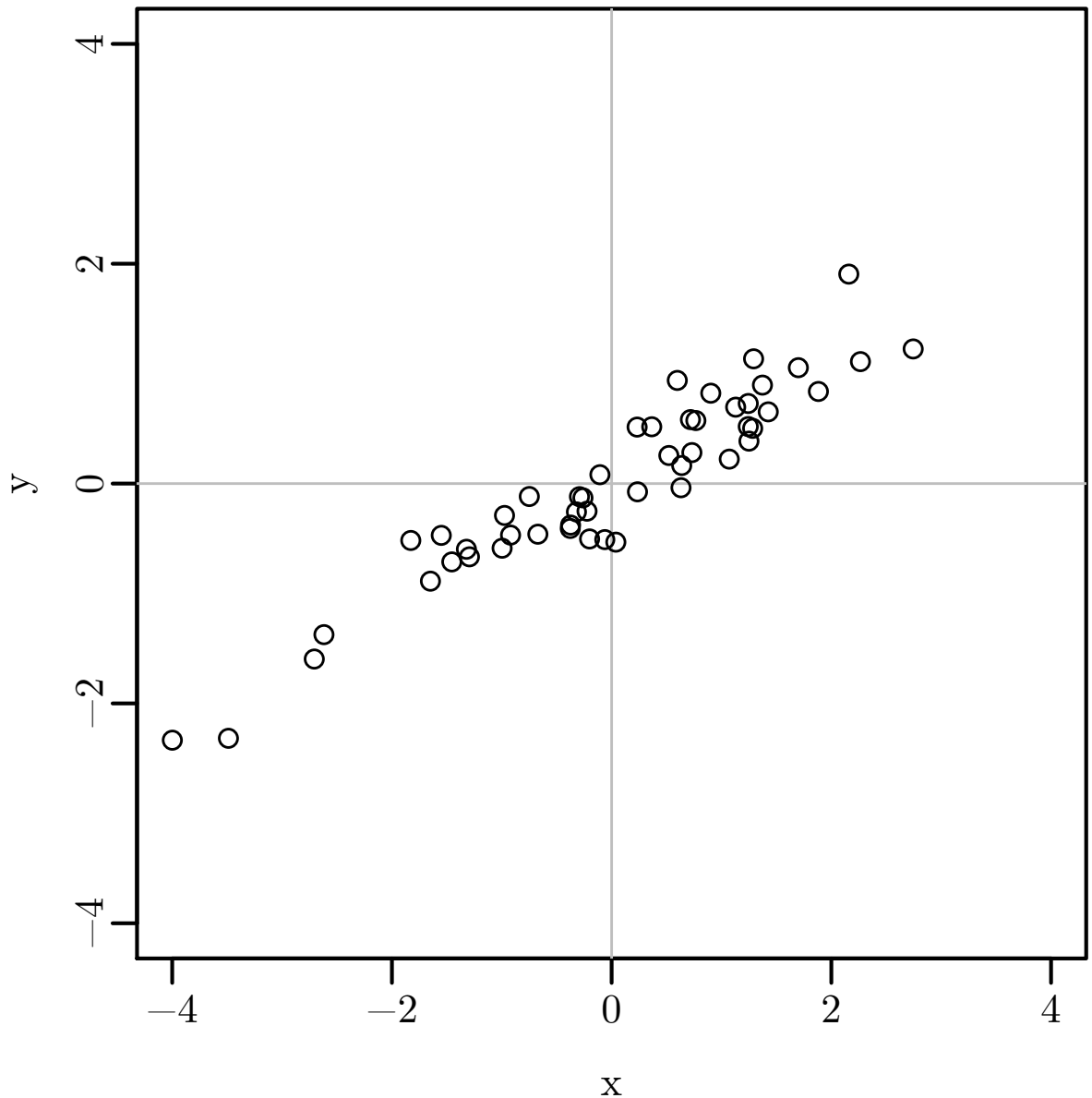
- Systems of linear equations can be written as matrices

$$\begin{aligned} 3x + 2y + z &= 39 \\ 2x + 3y + z &= 34 \\ x + 2y + 3z &= 26 \end{aligned} \quad \rightarrow \quad \left(\begin{array}{ccc|c} 3 & 2 & 1 & 39 \\ 2 & 3 & 1 & 34 \\ 1 & 2 & 3 & 26 \end{array}\right)$$

- and then be solved using linear algebra methods

$$\left(\begin{array}{ccc|c} 3 & 2 & 1 & 39 \\ & 5 & 1 & 24 \\ & & 12 & 33 \end{array}\right) \implies \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 9.25 \\ 4.25 \\ 2.75 \end{pmatrix}$$

Set of data points



$$\begin{pmatrix} x & y \\ -3.84 & -2.21 \\ -3.33 & -2.19 \\ -2.55 & -1.47 \\ -2.46 & -1.25 \\ -1.49 & -0.76 \\ -1.67 & -0.39 \\ -1.3 & -0.59 \\ \vdots & \vdots \\ 1.59 & 0.78 \\ 1.53 & 1.02 \\ 1.45 & 1.26 \\ 1.86 & 1.18 \\ 2.04 & 0.96 \\ 2.42 & 1.24 \\ 2.32 & 2.03 \\ 2.9 & 1.35 \end{pmatrix}$$

Linear maps

- Linear maps from $\mathbb{R}^3$ to $\mathbb{R}$

$$f_1(x, y, z) = 3x + 2y + z$$
$$f_2(x, y, z) = 2x + 3y + z$$
$$f_3(x, y, z) = x + 2y + 3z$$
$$f_4(x, y, z) = x$$

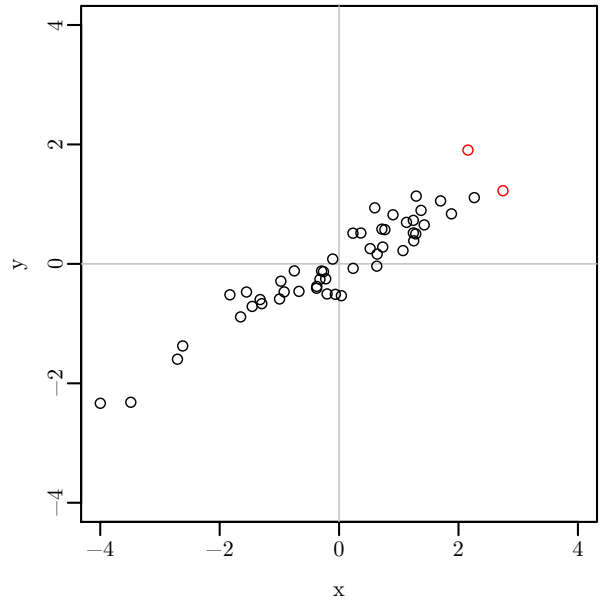
- Linear map f_1 written as a matrix

$$\begin{pmatrix} 3 & 2 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = f_1(x, y, z)$$

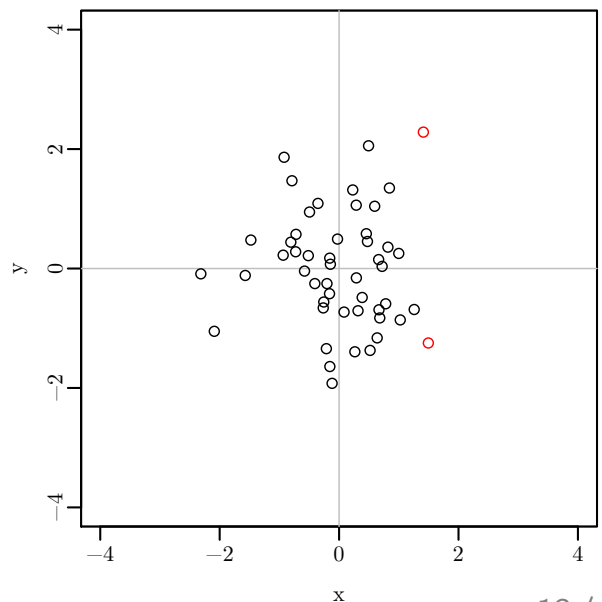
- Linear map from $\mathbb{R}^3$ to $\mathbb{R}^4$

$$\begin{pmatrix} 3 & 2 & 1 \\ 2 & 3 & 1 \\ 1 & 2 & 3 \\ 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} f_1(x, y, z) \\ f_2(x, y, z) \\ f_3(x, y, z) \\ f_4(x, y, z) \end{pmatrix}$$

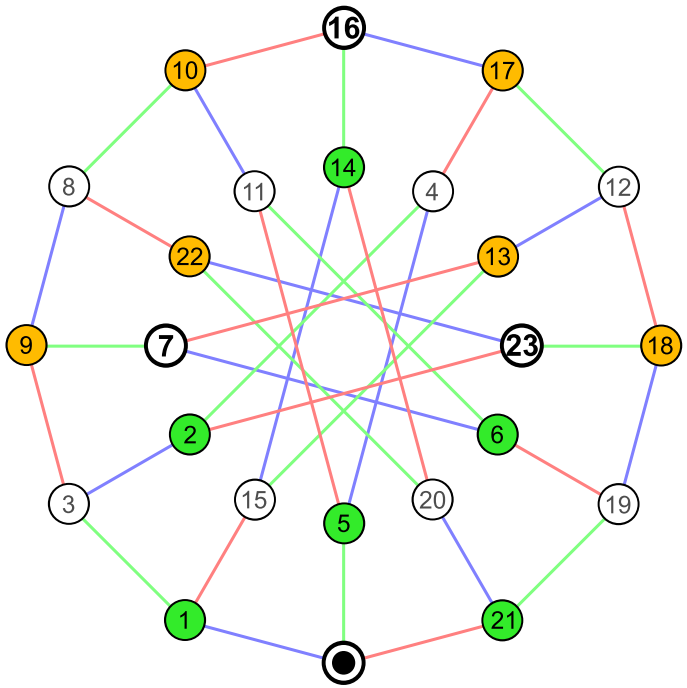
Original data

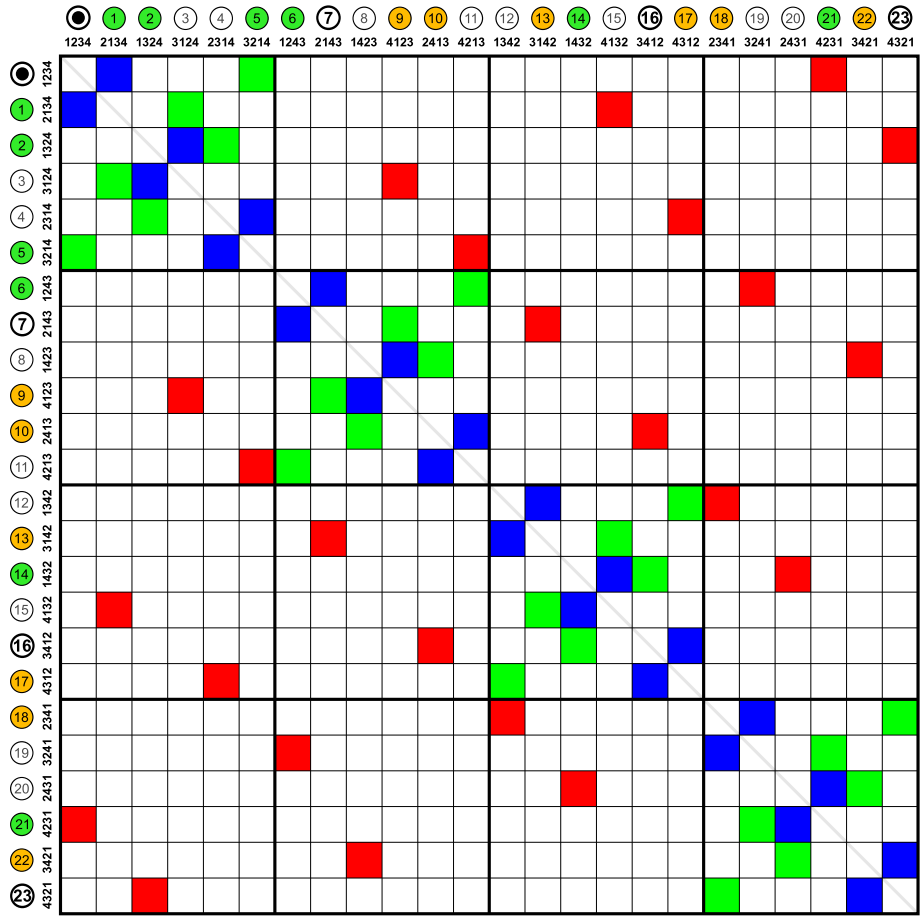


Rotated and stretched



Graphs





Adjacency matrix

Objects and attributes

Anna, Bob, and Charlie went shopping

- Anna bought butter and bread
- Bob bought butter, bread, and beer
- Charlie bought bread and beer

	Bread	Butter	Beer
Anna	1	1	0
Bob	1	1	1
Charlie	0	1	1

Customer transactions

	Data	Matrix	Mining
Book 1	5	0	3
Book 2	0	0	7
Book 3	4	6	5

Document-term matrix

	Avatar	The Matrix	Up
Alice		4	2
Bob	3	2	
Charlie	5		3

Incomplete rating matrix

	Jan	Jun	Sep
Saarbrücken	−1	11	10
Helsinki	−6.5	10.9	8.7
Cape Town	15.7	7.8	8.7

Cities and monthly temperatures

Many different kinds of data fit this object-attribute viewpoint.

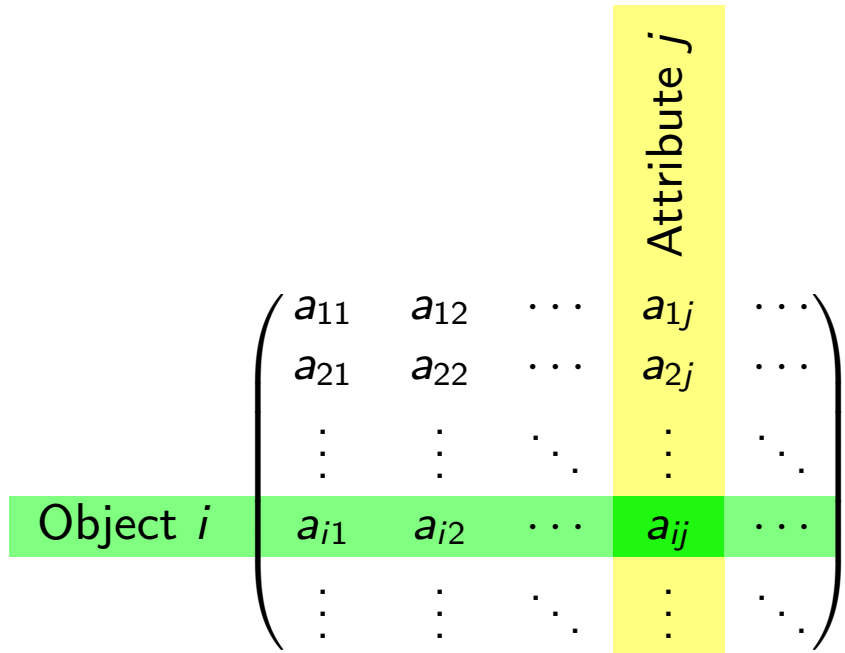
What is a matrix?

- A means to describe *computation*
  - Rotation
  - Rescaling
  - Permutation
  - Projection
  - ...

  } Linear operators

- A means to describe *data*

Rows	Columns	Entries
Objects	*Attributes*	*Values*
Equations	Variables	Coefficients
Data points	Axes	Coordinates
Vertices	Vertices	Edges
⋮	⋮	⋮



In data mining, we make use of both viewpoints simultaneously.

Outline

Key tool: Matrix decompositions

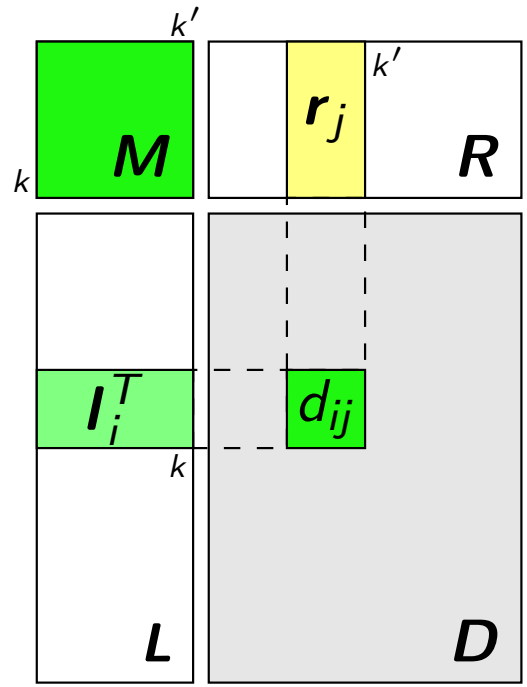
A **matrix decomposition** of a data matrix $\boldsymbol{D}$ is given by three matrices $\boldsymbol{L}$, $\boldsymbol{M}$, $\boldsymbol{R}$ such that

$$\boldsymbol{D} = \boldsymbol{LMR},$$

where

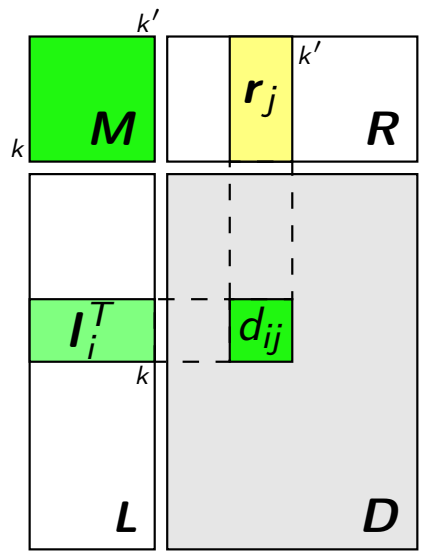
- $\boldsymbol{D}$ is an $m \times n$ data matrix,
- $\boldsymbol{L}$ is an $m \times r_1$ matrix,
- $\boldsymbol{M}$ is an $r_1 \times r_2$ matrix,
- $\boldsymbol{R}$ is an $r_2 \times n$ matrix, and
- r_1 and r_2 are integers ≥ 1.
- Often $r_1 = r_2 = r \geq 1$

$d_{ij} = \sum_{k,k'} l_{ik} m_{kk'} r_{k'j}$



Why matrix decompositions?

- Decompositions as just defined are not really helpful
  - Suppose we set $r = r_1 = r_2 = n$, $\boldsymbol{L} = \boldsymbol{D}$, $\boldsymbol{M} = \boldsymbol{R} = \boldsymbol{I}_n$ (the $n \times n$ identity matrix)
  - Then $\boldsymbol{D} = \boldsymbol{LMR} = \boldsymbol{DI}_n\boldsymbol{I}_n = \boldsymbol{D}$
  - But this does not provide insight
- To make decompositions useful, we want the decomposition to satisfy certain (carefully chosen) properties or contraints
- For example: we may want r to be small
  - Each object is represented by n numbers in $\boldsymbol{D}$
  - Each object is represented by r numbers in $\boldsymbol{L}$ (called **embedding** or **distributed representation**)
  - If $r < n$, we performed some form of compression
- Another example: we may want factors to have certain properties
  - Compare: integer factorization
  - $391 = 17 \cdot 13$



Approximate matrix decompositions

A **approximate matrix decomposition** of data matrix $\boldsymbol{D}$ is given by three matrices $\boldsymbol{L}$, $\boldsymbol{M}$, $\boldsymbol{R}$ such that

$$\boldsymbol{D} \approx \boldsymbol{LMR} = \hat{\boldsymbol{D}},$$

where each matrix has conforming dimensions (as before).

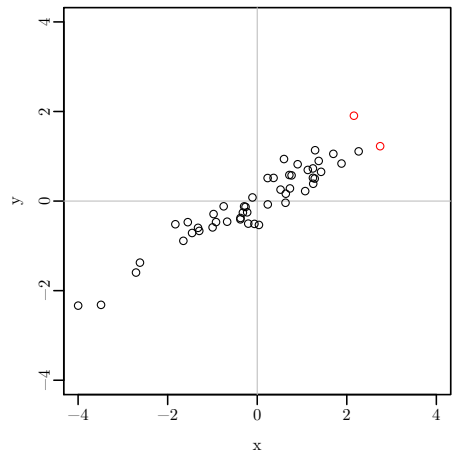
- We often look at approximate decompositions
  - Data is noisy anyway
  - Approximation may remove noise
  - Allows to focus on global (small r) or local (large r) patterns
  - Often more insightful
  - More efficient to compute
- $\approx$ defined by some **loss function** $L(\boldsymbol{D}, \hat{\boldsymbol{D}})$
  - E.g., squared error
  - Low means good approximation, high means bad
  - Finding the best approximation (smallest loss) can be hard
- We often say "matrix decomposition" when we actually mean "approximate matrix decomposition"

Some matrix decompositions

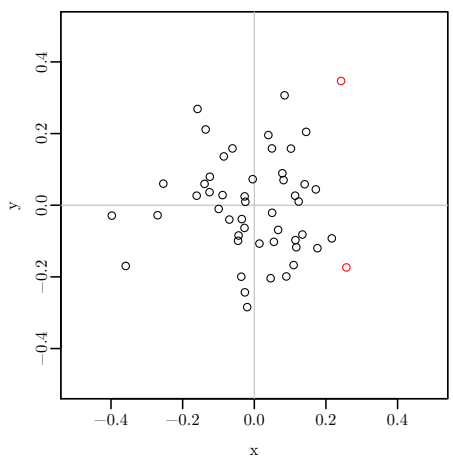
- There are many different decompositions, each enforcing different constraints and serving a different purpose
  - Singular value decomposition (SVD)
  - k-means (crisp or fuzzy)
  - Non-negative matrix factorization (NMF)
  - Semi-discrete decomposition (SDD)
  - Boolean matrix decomposition (BMF)
  - Independent component analysis (ICA)
  - Matrix completion
  - Probabilistic matrix factorization
  - Tensor decompositions
  - ...
- Picking the right one for the problem (if any) at hand is hard, experience helps
- Decompositions are not always easy (and often hard) to compute

Example: Singular value decomposition

$$\boldsymbol{D}_{50\times 2}$$



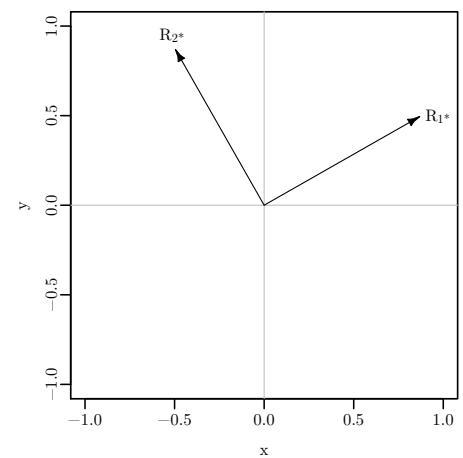
$$\boldsymbol{L}_{50\times 2}$$



$$\boldsymbol{M}_{2\times 2}$$

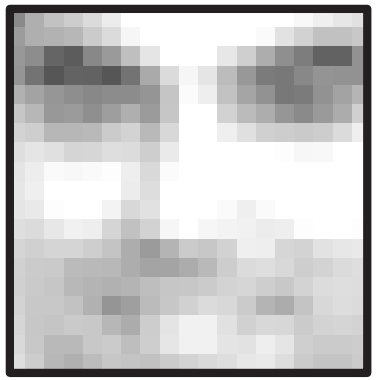
$$\begin{pmatrix} 11.73 & 0 \\ 0 & 1.71 \end{pmatrix}$$

$$\boldsymbol{R}_{2\times 2}$$

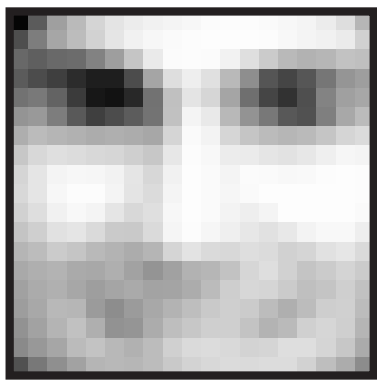


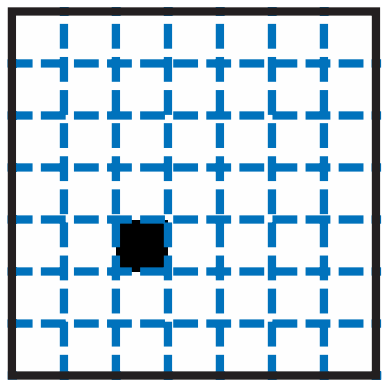
Example: k-Means

Original

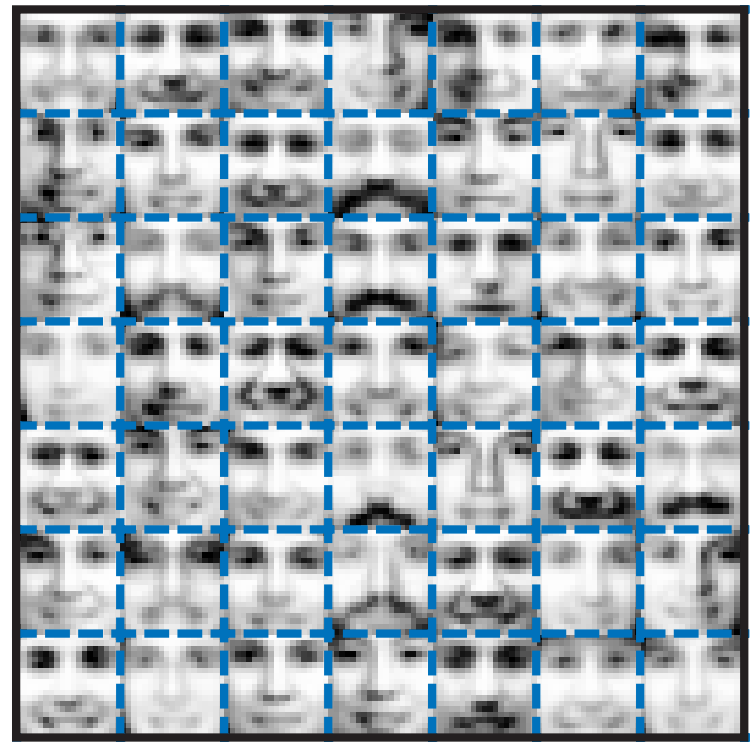


$\boldsymbol{d}_i^T$ (original)





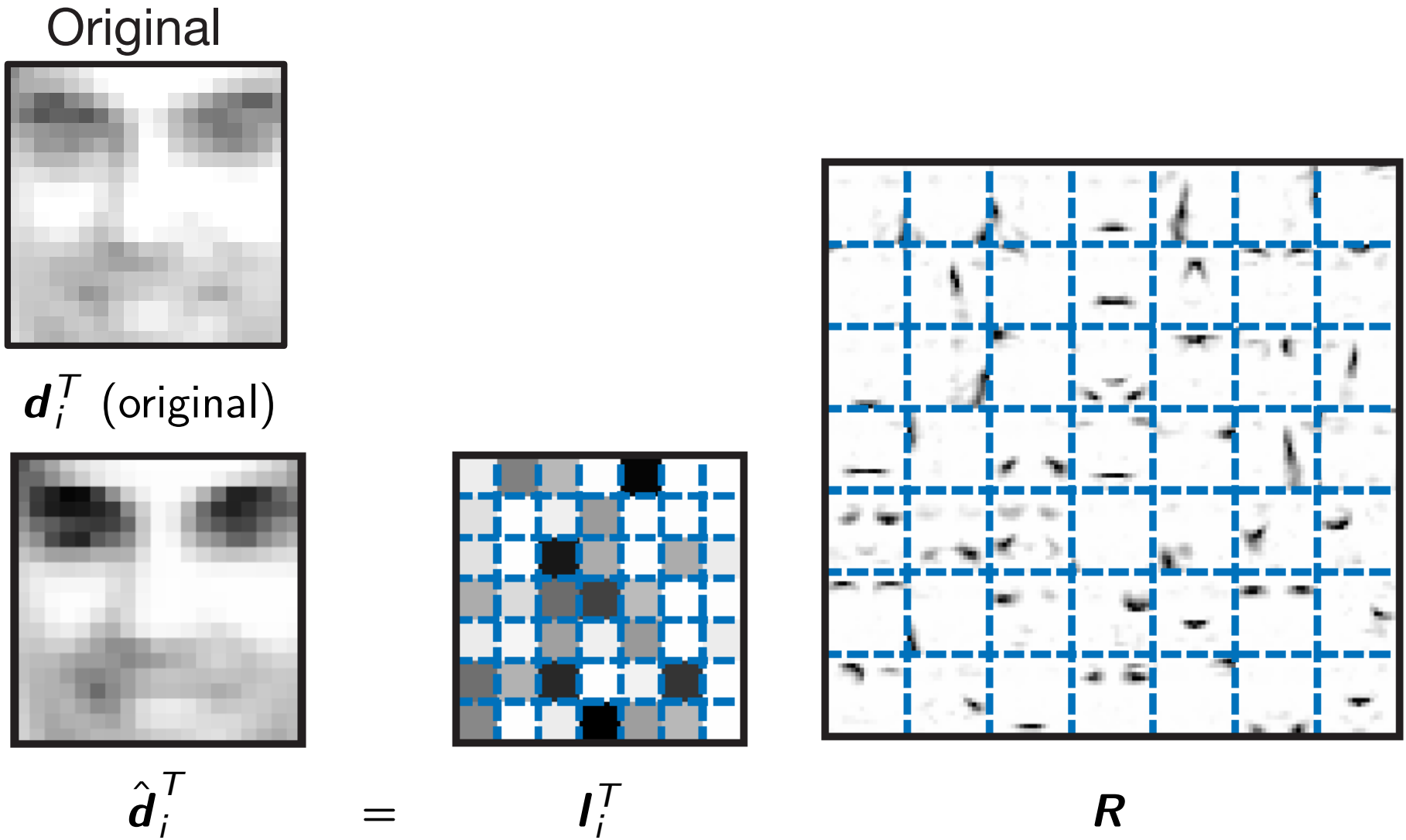
VQ



$\hat{\boldsymbol{d}}_i^T \quad = \quad \boldsymbol{l}_i^T \quad \boldsymbol{R}$

k-means factors correspond to prototypical faces.

Example: Non-negative matrix factorization



NMF factors correspond to parts of faces.

Lee and Seung, 1999.

Example: Latent Dirichlet allocation

"Arts"	"Budgets"	"Children"	"Education"
NEW	MILLION	CHILDREN	SCHOOL
FILM	TAX	WOMEN	STUDENTS
SHOW	PROGRAM	PEOPLE	SCHOOLS
MUSIC	BUDGET	CHILD	EDUCATION
MOVIE	BILLION	YEARS	TEACHERS
PLAY	FEDERAL	FAMILIES	HIGH
MUSICAL	YEAR	WORK	PUBLIC
BEST	SPENDING	PARENTS	TEACHER
ACTOR	NEW	SAYS	BENNETT
FIRST	STATE	FAMILY	MANIGAT
YORK	PLAN	WELFARE	NAMPHY
OPERA	MONEY	MEN	STATE
THEATER	PROGRAMS	PERCENT	PRESIDENT
ACTRESS	GOVERNMENT	CARE	ELEMENTARY
LOVE	CONGRESS	LIFE	HAITI

R
(topic×word)

The William Randolph Hearst Foundation will give $1.25 million to Lincoln Center, Metropolitan Opera Co., New York Philharmonic and Juilliard School. "Our board felt that we had a real opportunity to make a mark on the future of the performing arts with these grants an act every bit as important as our traditional areas of support in health, medical research, education and the social services," Hearst Foundation President Randolph A. Hearst said Monday in announcing the grants. Lincoln Center's share will be $200,000 for its new building, which will house young artists and provide new public facilities. The Metropolitan Opera Co. and New York Philharmonic will receive $400,000 each. The Juilliard School, where music and the performing arts are taught, will get $250,000. The Hearst Foundation, a leading supporter of the Lincoln Center Consolidated Corporate Fund, will make its usual annual $100,000 donation, too.

L
(doc×topic)

What can we do with matrix decompositions?

- Separate data from multiple processes
- Remove noise from the data
- Remove redundancy from the data
- Reveal latent structure and similarities in the data
- Cluster the data
- Fill in missing entries
- Find local patterns
- Reduce space consumption
- Reduce computational cost
- Aid visualization
- ...

Matrix decompositions can make data mining algorithms more effective. They may also provide insight into the data by themselves.

Factor interpretion of matrix decompositions

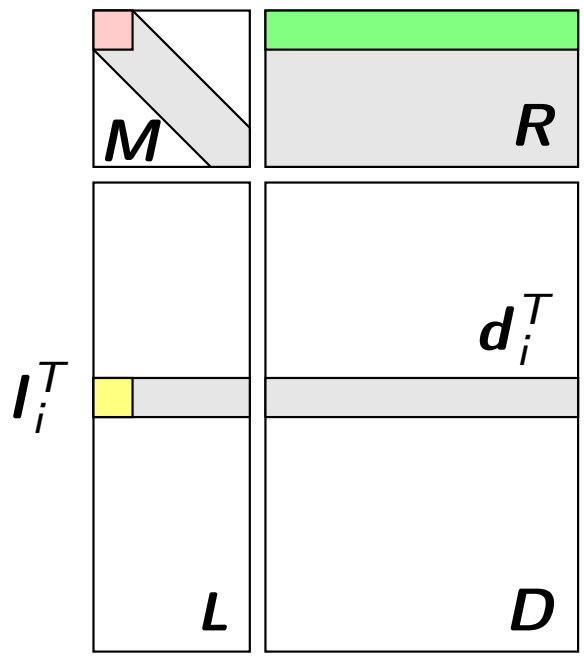
Assume that $\boldsymbol{M}$ is $r \times r$ and diagonal. Consider object i.

- Row of $\boldsymbol{R}$ = part (or piece), called **latent factor** ("latent object")
- Entry of $\boldsymbol{M}$ = weight of corresponding part
- Row of $\boldsymbol{MR}$ = weighted part
- Row of $\boldsymbol{L}$ = representation of object via weighted parts, called **embedding** or **distr. representation** (r pieces of information)
- r forces "compactness" (often $r < n$)

Each object can be viewed as a combination of r (weighted) "latent objects" (or "prototypical objects"). Similarly, each attribute can be viewed as a combination of r (weighted) "latent attributes."

(e.g., latent attribute = "body size"; latent object relates body size to real attributes such as "height", "weight", "shoe size")

$$\boldsymbol{d}_i^T = \sum_k l_{ik} m_{kk} \boldsymbol{r}_k^T$$



Example: Weather data ($r = 1$)

1.00			9.05	16.55	26.73	18.75	17.81

		Jan	Apr	Jul	Oct	Year
0.62	Stockholm	-0.70	8.60	21.90	9.90	10.00
0.69	Minsk	-2.10	12.20	23.60	10.20	10.60
0.83	London	7.90	13.30	22.80	15.20	14.80
0.90	Budapest	1.20	16.30	26.50	16.10	15.00
0.88	Paris	6.90	14.70	24.40	15.80	15.50
0.98	Bucharests	1.50	18.00	28.80	18.00	16.50
1.09	Barcelona	12.40	17.60	27.50	21.50	20.00
1.14	Rome	11.90	17.70	30.30	21.40	20.40
1.16	Lisbon	14.80	19.80	27.90	22.50	21.50
1.24	Athens	12.90	20.30	32.60	23.10	22.30
1.21	Valencia	16.10	20.20	29.10	23.60	22.30
1.27	Malta	16.10	20.00	31.50	25.20	23.20
						$\boldsymbol{D}$

Example: Weather data ($r = 1$), reconstruction

1.00			9.05	16.55	26.73	18.75	17.81

		Jan	Apr	Jul	Oct	Year
0.62	Stockholm	5.65	10.33	16.68	11.70	11.11
0.69	Minsk	6.21	11.36	18.35	12.87	12.23
0.83	London	7.55	13.80	22.28	15.63	14.85
0.90	Budapest	8.11	14.83	23.94	16.80	15.96
0.88	Paris	7.96	14.56	23.52	16.50	15.67
0.98	Bucharests	8.91	16.30	26.32	18.47	17.54
1.09	Barcelona	9.88	18.06	29.17	20.46	19.44
1.14	Rome	10.28	18.80	30.35	21.30	20.23
1.16	Lisbon	10.47	19.15	30.92	21.70	20.61
1.24	Athens	11.21	20.50	33.11	23.23	22.07
1.21	Valencia	10.92	19.96	32.24	22.62	21.48
1.27	Malta	11.47	20.98	33.88	23.77	22.58
						$\hat{\boldsymbol{D}}$

(RMSE: 2.66)

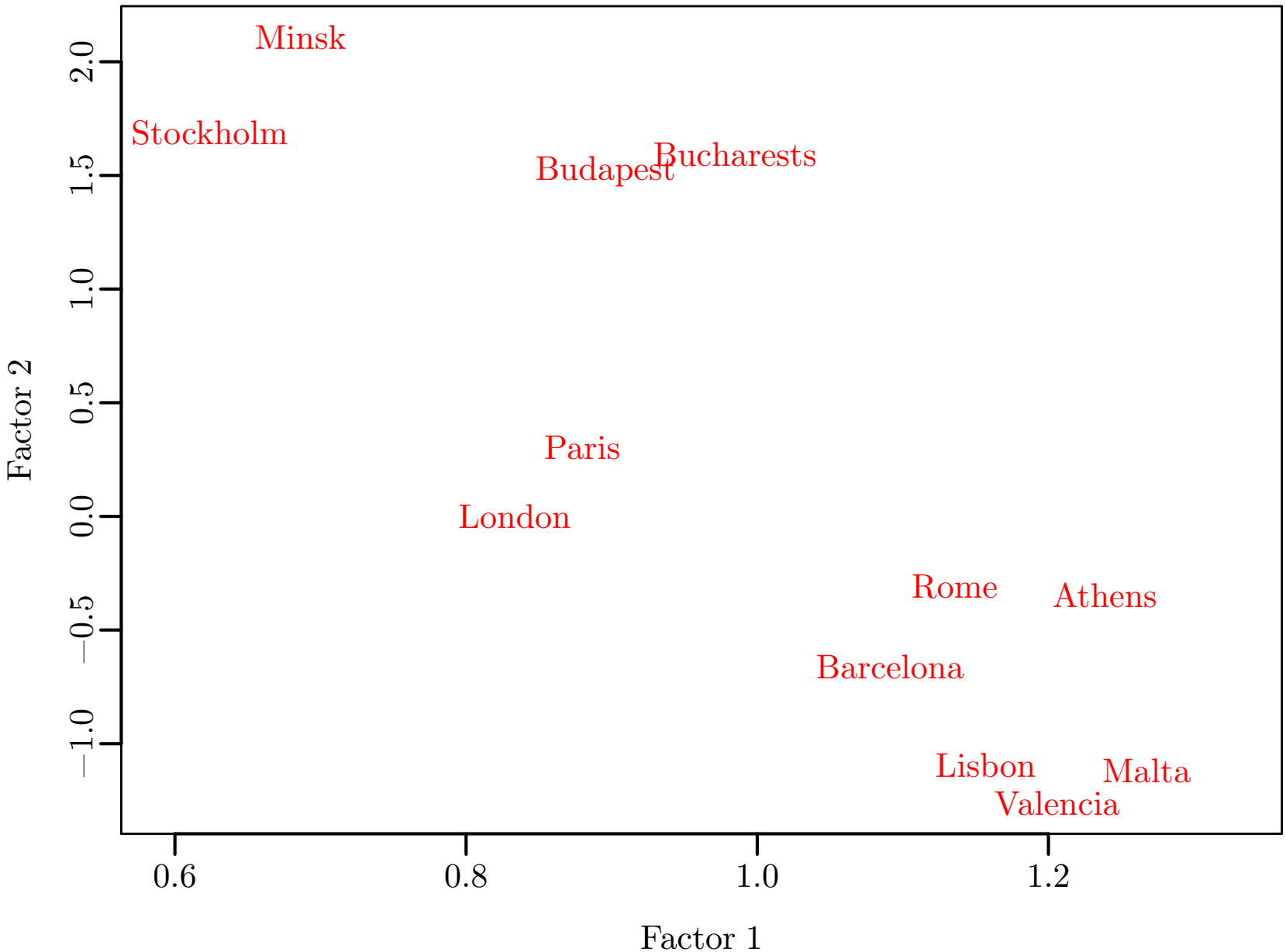
Example: Weather data ($r = 2$), reconstruction

1.00			9.05	16.55	26.73	18.75	17.81
	1.00		-4.14	0.27	2.32	-0.89	-0.69

			Jan	Apr	Jul	Oct	Year
0.62	1.69	Stockholm	-1.34	10.79	20.59	10.20	9.95
0.69	2.11	Minsk	-2.52	11.94	23.23	10.99	10.77
0.83	0.00	London	7.54	13.80	22.28	15.63	14.85
0.90	1.52	Budapest	1.82	15.24	27.46	15.45	14.91
0.88	0.30	Paris	6.71	14.65	24.22	16.23	15.46
0.98	1.59	Bucharests	2.31	16.74	30.02	17.05	16.44
1.09	-0.66	Barcelona	12.61	17.88	27.64	21.05	19.90
1.14	-0.31	Rome	11.55	18.71	29.64	21.57	20.44
1.16	-1.09	Lisbon	15.00	18.85	28.39	22.67	21.36
1.24	-0.35	Athens	12.65	20.41	32.31	23.54	22.31
1.21	-1.26	Valencia	16.14	19.62	29.31	23.74	22.36
1.27	-1.12	Malta	16.10	20.67	31.29	24.76	23.35
							$\hat{D}$

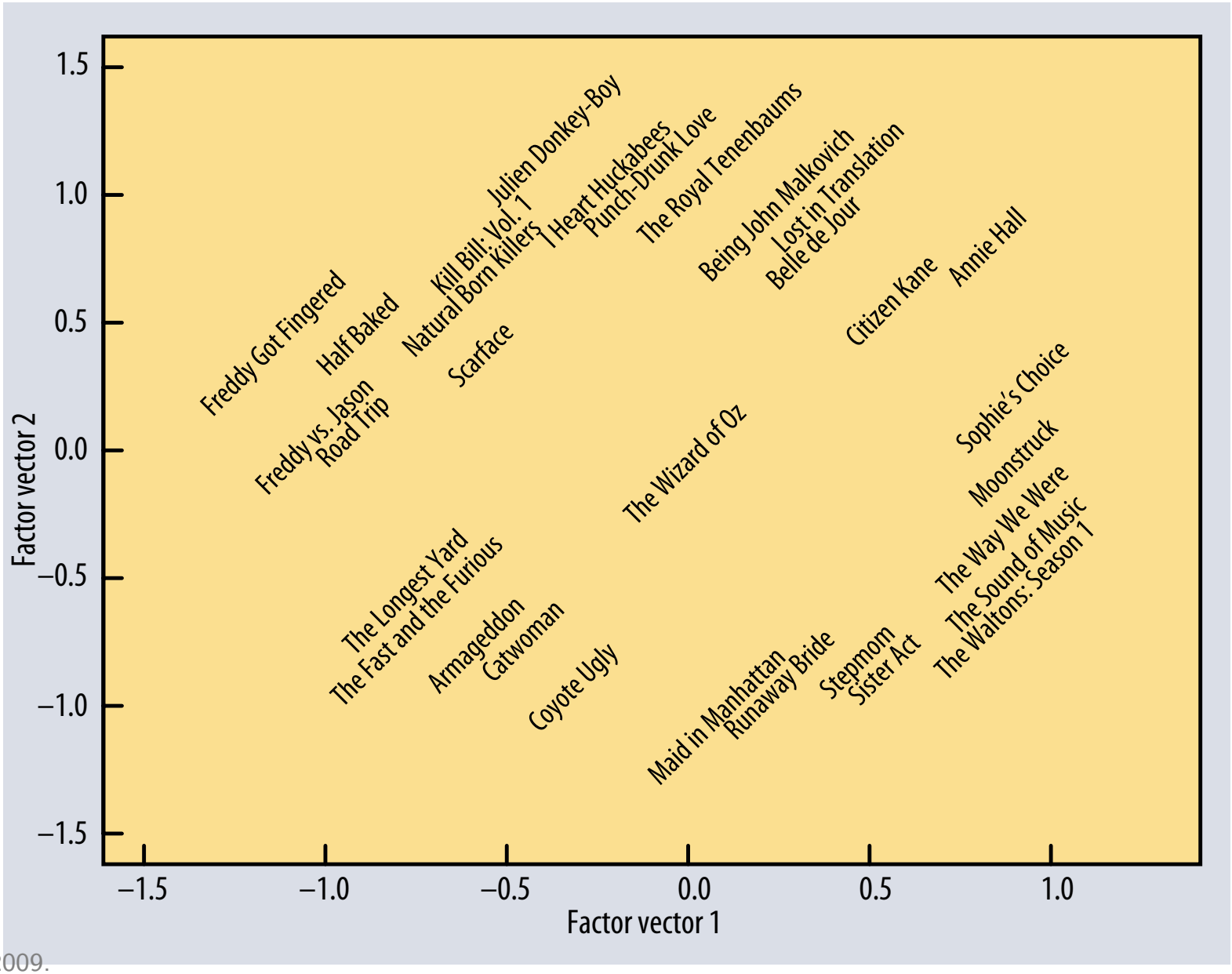
(RMSE: 0.60)

Example: Weather data ($r = 2$), plot



Example: Netflix prize data

($\approx$ 500k users, $\approx$ 17k movies, $\approx$ 100M ratings)



Koren et al., 2009.

Other interpretions

- Geometric interpretation
  - Transformation of n-dimensional space in r-dimensional space
  - Row of $\boldsymbol{R}$ = new axes
  - Row of $\boldsymbol{L}$ = new coordinates (*embeddings*)

- Component interpretation
  - $\boldsymbol{D}$ is viewed as consisting of r layers (of same shape as $\boldsymbol{D}$)
  - k-th layer described by $\boldsymbol{l}_k m_{kk} \boldsymbol{r}_k^T$
  - $\boldsymbol{D} = \sum_k \boldsymbol{l}_k m_{kk} \boldsymbol{r}_k^T$

- Graph interpretation
  - $\boldsymbol{D}$ is thought of as a bipartite graph with object and attribute vertexes
  - Edge weights measure association b/w objects and attributes
  - Decomposition thought of as a tripartite graph with row, waypoint, and column vertexes

All interpretations are useful (more later).

Outline

Lessons learned

- Data mining = from data to knowledge
  $\rightarrow$ Prediction, clustering, outlier detection, patterns
- Matrices are common representation of datasets
  $\rightarrow$ Linear maps, data points, sets, graphs, relational data, ...
- Often: rows = objects, columns = attributes
- Matrix decompositions reveal structure in the data
  $\rightarrow \boldsymbol{D \approx LMR}$
- Many different decompositions with different applications
  $\rightarrow$ SVD, k-means, NMF, SDD, BMF, ICA, completion, ...
- Factor interpretation: objects described by "prototypical objects"

Suggested reading

- Skillicorn, Ch. 1: *Data Mining*
- Skillicorn, Ch. 2: *Matrix Decompositions*

Data Mining and Matrices

02 – Vectors & Matrices

Prof. Dr. Rainer Gemulla

Universität Mannheim

FSS 2018

Outline

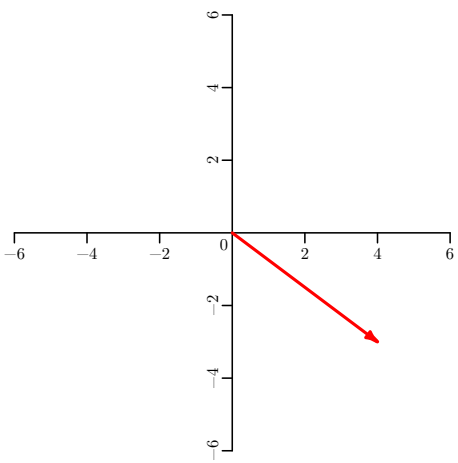
Outline

Vector

A **vector** is

- A 1D array of numbers
- A geometric entity with magnitude and direction
- A matrix with exactly one row or column
  - Called row vector and column vector, resp.
  - **Transpose** $\boldsymbol{v}^T$ transposes a row vector into a column vector and vice versa
- A (latent) object or attribute

	Year
Stockholm	9.95
Minsk	10.77
London	14.85
Budapest	14.91
Paris	15.46
Bucharests	16.44
Barcelona	19.90
Rome	20.44
Lisbon	21.36
Athens	22.31
Valencia	22.36
Malta	23.35



$$\begin{pmatrix} 4 \\ -3 \end{pmatrix}$$

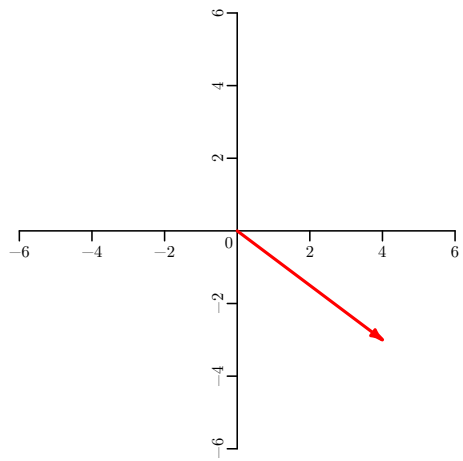
	Jan	*Apr*	*Jul*	*Oct*	*Year*
Stockholm	−0.70	8.60	21.90	9.90	10.00

Vector norm

The **norm** of vector defines its magnitude. Let $\boldsymbol{v} = \begin{pmatrix} v_1 & v_2 & \cdots & v_n \end{pmatrix}^T$.

- **Euclidean norm**: $\|\boldsymbol{v}\| = \sqrt{\sum_{i=1}^{n} v_i^2}$
  - Corresponds to intuitive notion of length in Euclidean space
- L_p **norm** for $1 \leq p \leq \infty$: $\|\boldsymbol{v}\|_p = \left(\sum_{i=1}^{n} |v_i|^p\right)^{1/p}$
  - L_1 norm = sum of absolute values (Manhattan distance from origin)
  - L_2 norm = Euclidean norm (bird-fly distance from origin)
  - L_∞ norm = maximum absolute value
  - The L_p norms never increase as p increases, i.e.,

$$\|\boldsymbol{v}\|_{p+a} \leq \|\boldsymbol{v}\|_p \qquad \text{for } a \geq 0$$



$$v = \begin{pmatrix} 4 \\ -3 \end{pmatrix}$$

$$\|\boldsymbol{v}\|_1 = 7$$

$$\|\boldsymbol{v}\| = 5$$

$$\|\boldsymbol{v}\|_\infty = 4$$

- Properties of vector norms
  - $\|\boldsymbol{v}\| > 0$ when $\boldsymbol{v} \neq 0$ and $\|\boldsymbol{v}\| = 0$ iff $\boldsymbol{v} = \boldsymbol{0}$
  - $\|a\boldsymbol{v}\| = |a| \, \|\boldsymbol{v}\|$ (absolute scalability)
  - $\|\boldsymbol{v}_1 + \boldsymbol{v}_2\| \leq \|\boldsymbol{v}_1\| + \|\boldsymbol{v}_2\|$ (triangle inequality)

Norms and distances

The **distance** between two vectors $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^n$ can be quantified with norm $\|\boldsymbol{u} - \boldsymbol{v}\|$.

	Jan	*Apr*	*Jul*	*Oct*	*Year*
• Stockholm, $\boldsymbol{s} =$	$(-0.70$	8.60	21.90	9.90	$10.00)$
• Minsk, $\boldsymbol{m} =$	$(-2.10$	12.20	23.60	10.20	$10.60)$
• Athens, $\boldsymbol{a} =$	$(12.90$	20.30	32.60	23.10	$22.30)$

L_1	$\boldsymbol{s}$	$\boldsymbol{m}$	$\boldsymbol{a}$
$\boldsymbol{s}$	0.00	7.60	61.50
$\boldsymbol{m}$	7.60	0.00	56.70
$\boldsymbol{a}$	61.50	56.70	0.00

L_2	$\boldsymbol{s}$	$\boldsymbol{m}$	$\boldsymbol{a}$
$\boldsymbol{s}$	0.00	4.27	27.60
$\boldsymbol{m}$	4.27	0.00	25.98
$\boldsymbol{a}$	27.60	25.98	0.00

L_∞	$\boldsymbol{s}$	$\boldsymbol{m}$	$\boldsymbol{a}$
$\boldsymbol{s}$	0.00	3.60	13.60
$\boldsymbol{m}$	3.60	0.00	15.00
$\boldsymbol{a}$	13.60	15.00	0.00

Dot product (algebraic definition)

The **dot product** of two vectors $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^n$ is given by

$$\boldsymbol{u} \cdot \boldsymbol{v} = \sum_{i=1}^{n} u_i v_i.$$

- Also known as **scalar product** or **inner product**
- We'll often use matrix product notation and write $\boldsymbol{u}^T \boldsymbol{v}$
- Properties (with $a, b \in \mathbb{R}$)
  - $\boldsymbol{u} \cdot \boldsymbol{v} = \boldsymbol{v} \cdot \boldsymbol{u}$
  - $(a\boldsymbol{u}) \cdot \boldsymbol{v} = a(\boldsymbol{u} \cdot \boldsymbol{v})$
  - $(a\boldsymbol{u} + b\boldsymbol{v}) \cdot \boldsymbol{w} = (a\boldsymbol{u}) \cdot \boldsymbol{w} + (b\boldsymbol{v}) \cdot \boldsymbol{w}$
- Many uses, many interpretations

With dot products, we can ...

- Compute the (squared) Euclidean norm

$$\boldsymbol{v} \cdot \boldsymbol{v} = \sum_{i=1}^{n} v_i^2 = \|\boldsymbol{v}\|^2$$

- Normalize a vector to length 1 (then a **unit vector**)

$$\hat{\boldsymbol{v}} = \boldsymbol{v} / \|\boldsymbol{v}\|$$

- Determine the value of a coordinate

$$v_i = \boldsymbol{v} \cdot \boldsymbol{e}_i,$$

  where $\boldsymbol{e}_i$ denotes the i-th standard basis vector (i.e., $[\boldsymbol{e}_i]_j = 1$ if $i = j$ else 0)

- Compute the sum of the elements of a vector

$$\boldsymbol{v} \cdot \mathbf{1}_{\boldsymbol{n}} = \sum_{i=1}^{n} v_i,$$

  where $\mathbf{1}_n$ is the all-ones vector of dimensionality n

- ...

Dot product: Weighted sum

The elements of one vector are interpreted as weights for the elements of the other vector.

Example: Anna goes shopping

Item	Bread	Butter	Pizza
Price/piece	1 €	0.50 €	3 €
Quantity bought	1	2	5

- How much does Anna pay?
- Prices can be interpreted as "weights": $\boldsymbol{p} = \begin{pmatrix} 1 & 0.5 & 3 \end{pmatrix}^T$
- Quantities are $\boldsymbol{n} = \begin{pmatrix} 1 & 2 & 5 \end{pmatrix}^T$
- Total is $\boldsymbol{p} \cdot \boldsymbol{n} = 1 \cdot 1 + 0.5 \cdot 2 + 3 \cdot 5 = 17$
- Similarly: Can interpret quantities as weights for prices

Dot product: Expected value

One vector corresponds to probabilities, the other one to a random variable.

Example: Bob is gambling

Outcome	Jackpot	Win	Loss
Probability	0.1	0.2	0.7
Amount won	5 €	1 €	-2 €

- How much does Bob win in expectation? (Should he play?)
- Probabilities $\boldsymbol{p} = \begin{pmatrix} 0.1 & 0.2 & 0.7 \end{pmatrix}^T$
  - A non-negative vector that sums to one ($\|\boldsymbol{p}\|_1 = 1$) is called a **probability vector**
  - Corresponds to a probability distribution over a finite set of outcomes
- Amounts won $\boldsymbol{x} = \begin{pmatrix} 5 & 1 & -2 \end{pmatrix}^T$
  - Corresponds to a random variable; associates a real value with each outcome
- Expected value $\boldsymbol{p} \cdot \boldsymbol{x} = 0.1 \cdot 5 + 0.2 \cdot 1 + 0.7 \cdot (-2) = -0.7$

Dot product: Sample variance

Denote by $\bar{u} = \frac{1}{n}\sum_i u_i$ the mean of $\boldsymbol{u}$. If we treat the entries of $\boldsymbol{u}$ as samples from some distribution, then the unbiased **sample variance** is given by

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(u_i - \bar{u})^2 = \frac{\|\boldsymbol{u} - \bar{\boldsymbol{u}}\|^2}{n-1} = \frac{(\boldsymbol{u} - \bar{\boldsymbol{u}}) \cdot (\boldsymbol{u} - \bar{\boldsymbol{u}})}{n-1},$$

where $\bar{\boldsymbol{u}}$ denotes the **sample mean vector**, i.e., $[\bar{\boldsymbol{u}}]_i = \bar{u}$ for $1 \leq i \leq n$.

- Example
  - $\boldsymbol{u} = \begin{pmatrix} 10 & 11 & 12 \end{pmatrix}^T$
  - $\bar{u} = 11$, $\bar{\boldsymbol{u}} = \begin{pmatrix} 11 & 11 & 11 \end{pmatrix}^T$
  - $\boldsymbol{u} - \bar{\boldsymbol{u}} = \begin{pmatrix} -1 & 0 & 1 \end{pmatrix}^T$
  - $s^2 = 1$, $\|\boldsymbol{u}\|^2 = 365$
- Variances are thus closely related to norms; the key difference is centering and averaging
- When we center data before analyzing it, dot products are proportional to variances ($\boldsymbol{u} \cdot \boldsymbol{u}$) or covariances ($\boldsymbol{u} \cdot \boldsymbol{v}$)

Dot product: Sets and intersections

The **indicator vector** of a subset T of a set $S = \{ s_1, \ldots, s_n \}$ is the vector $\boldsymbol{x}$ such that $x_i = 1$ if $s_i \in T$ and $x_i = 0$ if $s_i \notin T$. If $\boldsymbol{u}$ and $\boldsymbol{v}$ are indicator vectors for subsets $U, V \subseteq S$, resp., then $\boldsymbol{u} \cdot \boldsymbol{v} = |U \cap V|$.

- $S = \{ \text{France}, \text{Germany}, \text{Denmark}, \text{Poland} \}$
- Anna visited France, Germany, and Poland: $\boldsymbol{u} = \begin{pmatrix} 1 & 1 & 0 & 1 \end{pmatrix}^T$
- Bob visited Germany, Denmark, and Poland: $\boldsymbol{v} = \begin{pmatrix} 0 & 1 & 1 & 1 \end{pmatrix}^T$
- Number of countries visited by both:

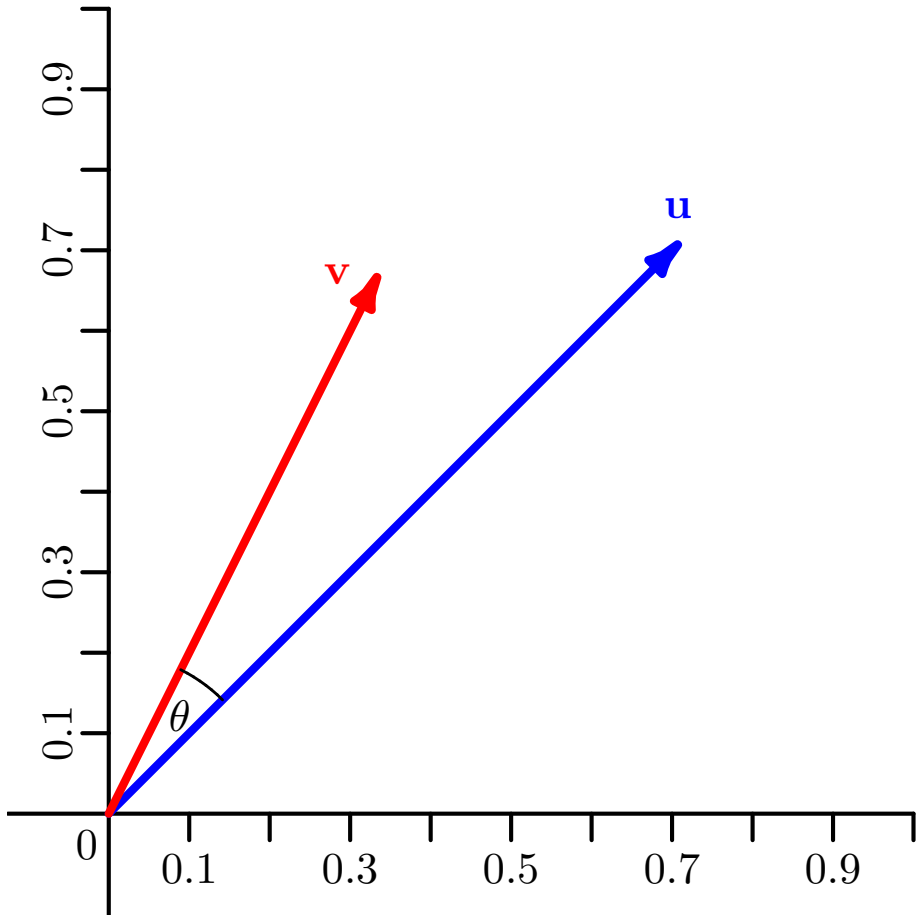
$$\boldsymbol{u} \cdot \boldsymbol{v} = 1 \cdot 0 + 1 \cdot 1 + 0 \cdot 1 + 1 \cdot 1 = 2 = |\{ \text{Germany}, \text{Poland} \}|$$

Dot product (geometric definition)

An alternative geometric definition of the dot product of two vectors $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^n$ is

$$\boldsymbol{u} \cdot \boldsymbol{v} = \|\boldsymbol{u}\| \, \|\boldsymbol{v}\| \cos\theta,$$

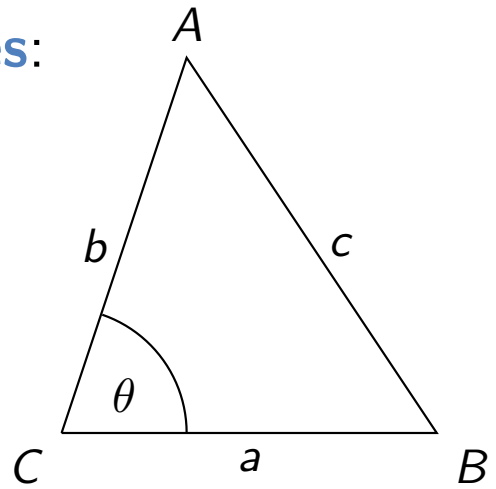
where $-\pi \leq \theta \leq \pi$ denotes the angle between $\boldsymbol{u}$ and $\boldsymbol{v}$.



Why is this?

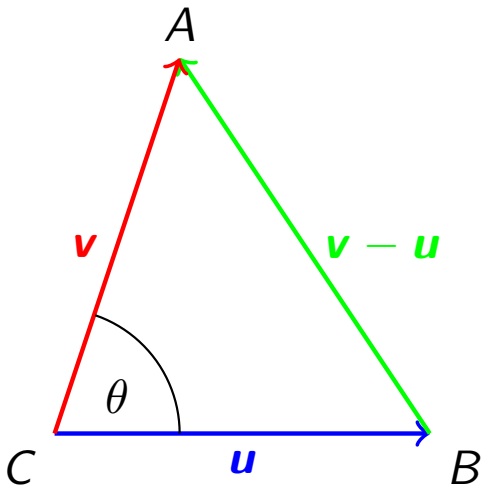
Let's focus on the 2D case. Recall the **law of cosines**:

$$c^2 = a^2 + b^2 - 2ab\cos\theta.$$



Now set $\boldsymbol{u} = B - C$ and $\boldsymbol{v} = A - C$ and observe that $\boldsymbol{v} - \boldsymbol{u} = A - B$.

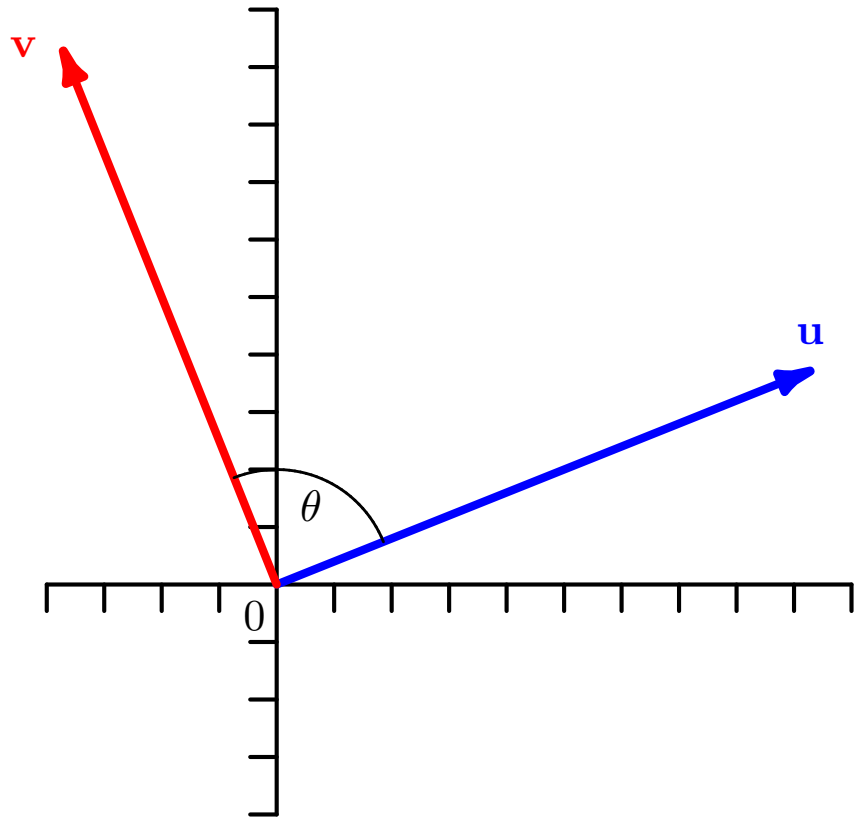
$$\begin{aligned}
\cos\theta &= \frac{a^2 + b^2 - c^2}{2ab} = \frac{\|\boldsymbol{u}\|^2 + \|\boldsymbol{v}\|^2 - \|\boldsymbol{v} - \boldsymbol{u}\|^2}{2\,\|\boldsymbol{u}\|\,\|\boldsymbol{v}\|} \\
&= \frac{\boldsymbol{u}\cdot\boldsymbol{u} + \boldsymbol{v}\cdot\boldsymbol{v} - (\boldsymbol{v} - \boldsymbol{u})\cdot(\boldsymbol{v} - \boldsymbol{u})}{2\,\|\boldsymbol{u}\|\,\|\boldsymbol{v}\|} \\
&= \frac{\boldsymbol{u}\cdot\boldsymbol{u} + \boldsymbol{v}\cdot\boldsymbol{v} - \boldsymbol{v}\cdot\boldsymbol{v} + 2\boldsymbol{u}\cdot\boldsymbol{v} - \boldsymbol{u}\cdot\boldsymbol{u}}{2\,\|\boldsymbol{u}\|\,\|\boldsymbol{v}\|} \\
&= \frac{\boldsymbol{u}\cdot\boldsymbol{v}}{\|\boldsymbol{u}\|\,\|\boldsymbol{v}\|}
\end{aligned}$$



Dot product: Test for orthogonality

Two nonzero vectors $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^n$ are **orthogonal** iff $\boldsymbol{u} \cdot \boldsymbol{v} = 0$.

- Since $0 = \boldsymbol{u} \cdot \boldsymbol{v} = \|\boldsymbol{u}\| \, \|\boldsymbol{v}\| \cos\theta$ and $\|\boldsymbol{u}\|, \|\boldsymbol{v}\| > 0$, we have $\cos\theta = 0$
- And this means that the angle is 90 degrees



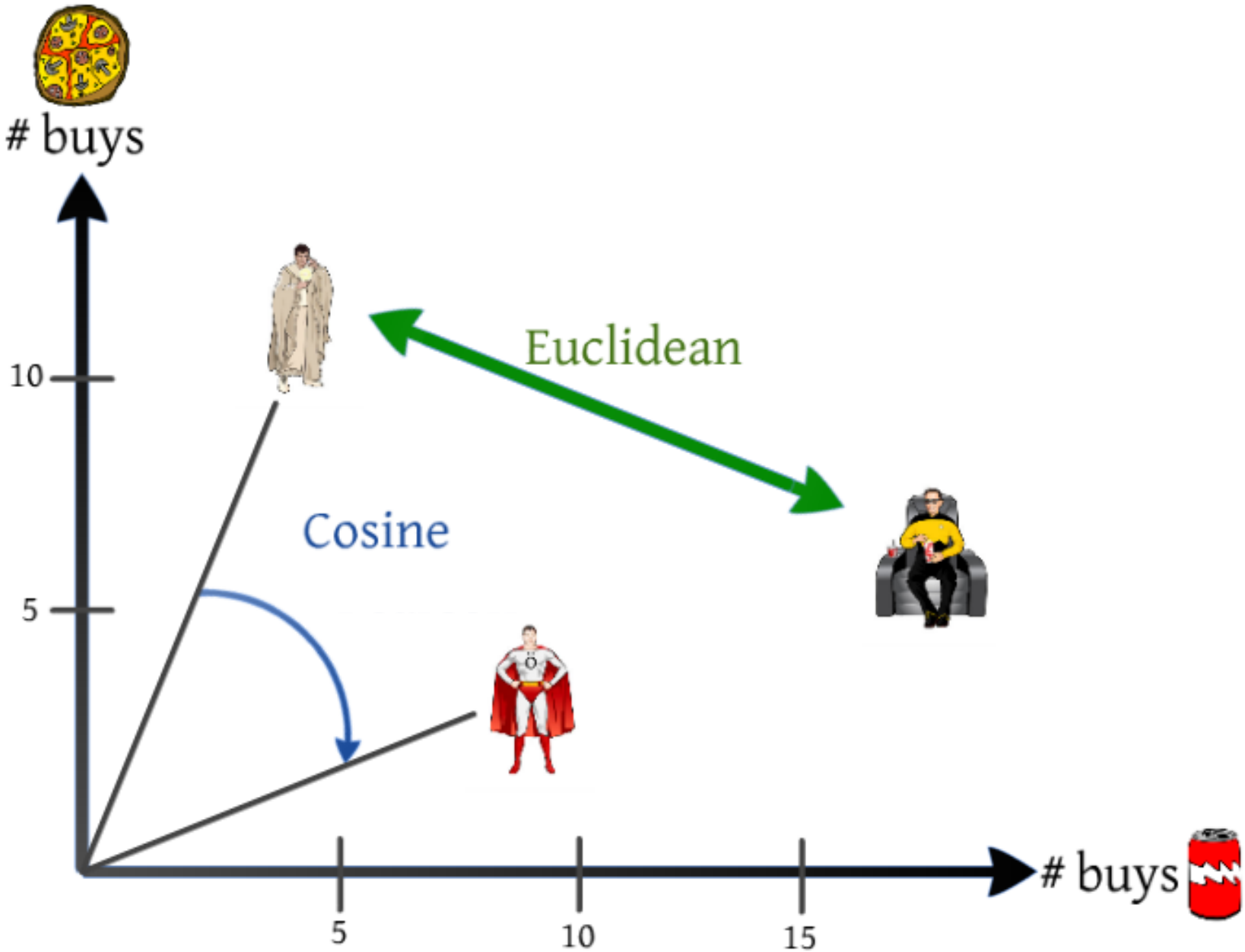
Dot product: Cosine similarity (1)

The angle between $\boldsymbol{u}$ and $\boldsymbol{v}$ is another way to measure the similarity between two vectors. The **cosine similarity** of $\boldsymbol{u}$ and $\boldsymbol{v}$ is given by

$$\cos(\boldsymbol{u}, \boldsymbol{v}) = \frac{\boldsymbol{u} \cdot \boldsymbol{v}}{\|\boldsymbol{u}\| \, \|\boldsymbol{v}\|}.$$

- $-1 \leq \cos(\boldsymbol{u}, \boldsymbol{v}) \leq 1$
- Vectors that point in roughly the same direction $\rightarrow$ small angle $\rightarrow$ cosine similarity ≈ 1
- Vectors that point in roughly opposite directions $\rightarrow$ large angle $\rightarrow$ cosine similarity ≈ -1
- Vectors that are roughly orthogonal $\rightarrow$ roughly right angle $\rightarrow$ cosine similarity ≈ 0
- Popular in IR to determine the similarity between a document and a query

Dot product: Cosine similarity (2)

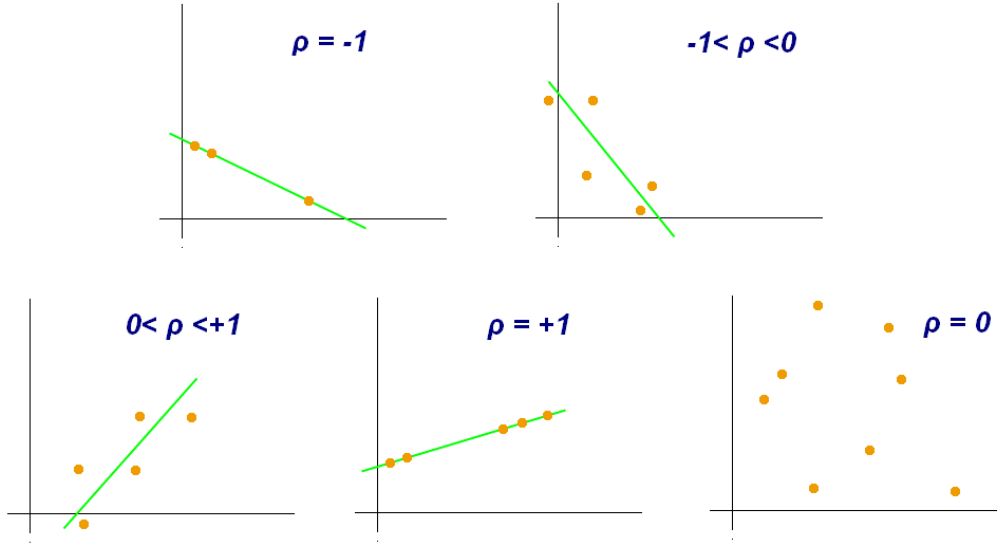


Dot product: Pearson correlation

The **sample Pearson correlation coefficient** is a measure of linear correlation. It is given by

$$r_{\boldsymbol{x},\boldsymbol{y}} = \frac{(\boldsymbol{x} - \bar{\boldsymbol{x}}) \cdot (\boldsymbol{y} - \bar{\boldsymbol{y}})}{\|\boldsymbol{x} - \bar{\boldsymbol{x}}\| \, \|\boldsymbol{y} - \bar{\boldsymbol{y}}\|}.$$

- Numerator proportional to the sample covariance
- Denominator proportional to sample standard deviations
- Closely related to cosine similarity but performs centering
  - This is sometimes desired
  - And sometimes a bad idea (e.g., example last slide)



Dot product: Similarity

The dot product itself can also be seen as a measure of similarity or compatibility. Recall

$$\boldsymbol{u} \cdot \boldsymbol{v} = \|\boldsymbol{u}\| \, \|\boldsymbol{v}\| \cos\theta.$$

Example: Shopping transactions

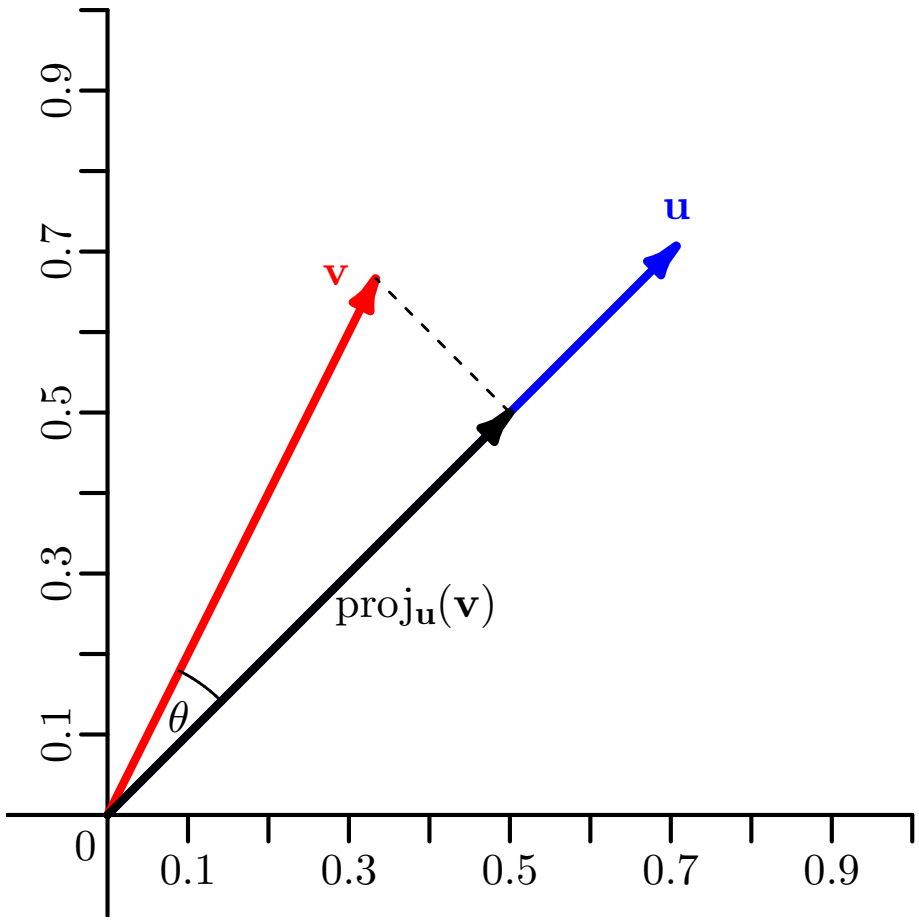
- Like in previous example, vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ correspond to persons
- Elements correponds to frequencies of buying each product
- We can think of the direction of a vector as "preference"
  - Which products are being bought?
  - $\cos\theta$ large when $\boldsymbol{u}$ and $\boldsymbol{v}$ have similar interest
- We can think of the magnitude of a vector as "strength"
  - How much is being bought?
  - $\|\boldsymbol{u}\| \, \|\boldsymbol{v}\|$ large when both persons buy a lot
- If $\boldsymbol{u} \cdot \boldsymbol{v}$ is large, $\boldsymbol{u}$ and $\boldsymbol{v}$ have similar shopping behaviour and buy a lot

Dot product: Projection

The **vector projection** of $\boldsymbol{v}$ onto $\boldsymbol{u}$ is given by

$$\text{proj}_{\boldsymbol{u}}(\boldsymbol{v}) = \underbrace{\frac{\boldsymbol{u}\cdot\boldsymbol{v}}{\|\boldsymbol{u}\|}}_{\substack{\textbf{scalar}\\ \textbf{projection}}} \frac{\boldsymbol{u}}{\|\boldsymbol{u}\|} = \frac{\boldsymbol{u}\cdot\boldsymbol{v}}{\|\boldsymbol{u}\|^2}\boldsymbol{u}$$

- The scalar projection describes how far $\boldsymbol{v}$ points in the direction $\boldsymbol{u}$
- The vector projection is a vector pointing this far in the direction of $\boldsymbol{u}$



Outline

Notation

Let $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ be a real $m \times n$ matrix. We write

- a_{ij} or $\boldsymbol{A}_{ij}$ (both scalars) for the value of entry (i, j)
- $\boldsymbol{a}_j$ or $\boldsymbol{A}_{*j}$ (both column vectors) for the j-th column of $\boldsymbol{A}$
- $\boldsymbol{a}_i$ (column vector) or $\boldsymbol{A}_{i*}$ (row vector) for the i-th row of $\boldsymbol{A}$

Thus

$$\boldsymbol{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$$

$$= \begin{pmatrix} \boldsymbol{A}_{*1} & \boldsymbol{A}_{*2} & \cdots & \boldsymbol{A}_{*n} \end{pmatrix} = \begin{pmatrix} \boldsymbol{A}_{1*} \\ \boldsymbol{A}_{2*} \\ \vdots \\ \boldsymbol{A}_{m*} \end{pmatrix}$$

Full matrix ring (addition)

The set of all matrices in $\mathbb{R}^{n \times n}$ form a ring, the **full matrix ring**.

- Addition and substraction are element-wise

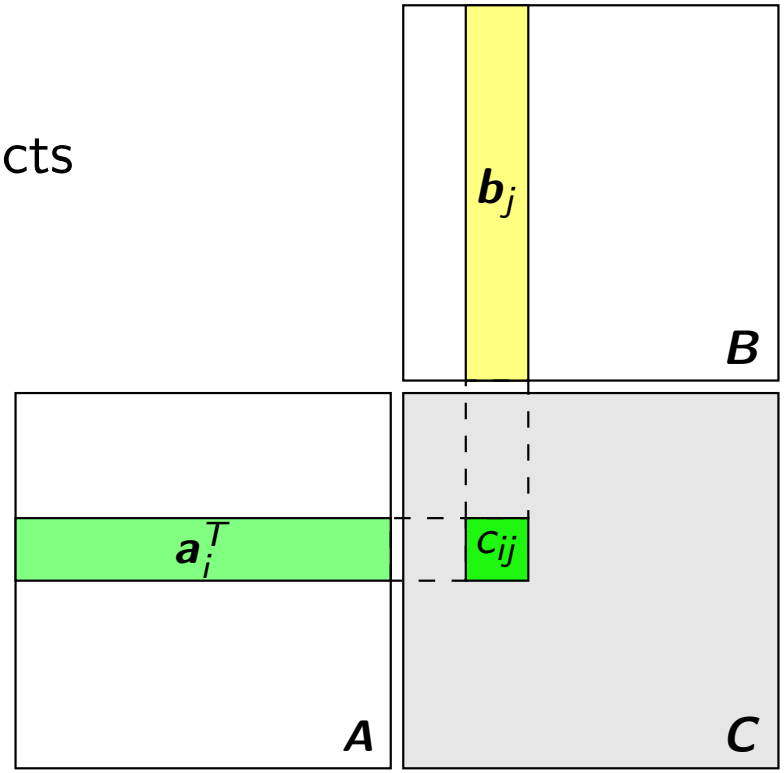
$$[\boldsymbol{A} + \boldsymbol{B}]_{ij} = a_{ij} + b_{ij}$$
$$[\boldsymbol{A} - \boldsymbol{B}]_{ij} = a_{ij} - b_{ij}$$

- The additive identity is the $n \times n$ zero matrix $\mathbf{0}_{n \times n}$
- The additive inverse is $-\boldsymbol{A}$ with $[-\boldsymbol{A}]_{ij} = -a_{ij}$
- In general $[c\boldsymbol{A}]_{ij} = ca_{ij}$ for $c \in \mathbb{R}$ (scalar multiplication)
- Addition is associative and commutative

Full matrix ring (multiplication)

- For multiplication, we take dot products

$$[\boldsymbol{AB}]_{ij} = \boldsymbol{a}_i \cdot \boldsymbol{b}_j = \sum_{k=1}^{n} a_{ik} b_{kj}$$



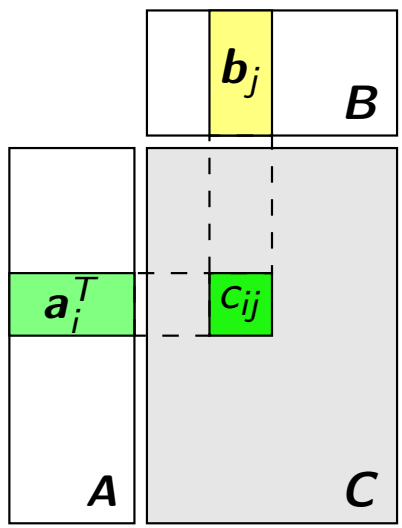
- The multiplicative identity is the $n \times n$ **identity matrix** $\boldsymbol{I}_n$

$$\boldsymbol{I}_n = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}$$

- Multiplication is associative, but not commutative ($\boldsymbol{AB} \neq \boldsymbol{BA}$ in general)
- Multiplication distributes over addition ($\boldsymbol{A}(\boldsymbol{B}+\boldsymbol{C}) = \boldsymbol{AB} + \boldsymbol{AC}$ and $(\boldsymbol{B}+\boldsymbol{C})\boldsymbol{A} = \boldsymbol{BA} + \boldsymbol{CA}$)
- Multiplication does not always have an inverse (division)

Rectangular matrices

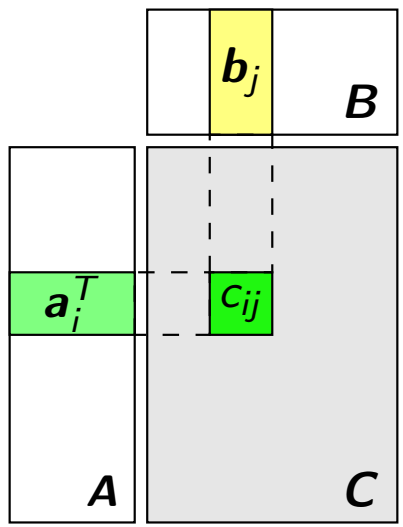
- We generally have rectangular matrices $\boldsymbol{A} \in \mathbb{R}^{m \times n}$
- We can only add and substract matrices of the same dimensions ($\boldsymbol{A}_{m \times n} + \boldsymbol{B}_{m \times n}$)
- We can only multiply matrices with a matching inner dimension
  - We can multiply $\boldsymbol{A} \in \mathbb{R}^{m \times r}$ with $\boldsymbol{B} \in \mathbb{R}^{r \times n}$ (inner dimension is r)
  - Gives an $m \times n$ matrix (outer dimensions)
  - $[\boldsymbol{AB}]_{ij} = \boldsymbol{a}_i \cdot \boldsymbol{b}_j = \sum_{k=1}^{r} a_{ik} b_{kj}$



Interpretation for matrix multiplication (1)

When we multiply $\boldsymbol{A}$ and $\boldsymbol{B}$, we compute all dot products between rows of $\boldsymbol{A}$ and columns of $\boldsymbol{B}$.

- We can apply any of the interpretations of the dot product
- E.g., weighted sum
  - m supermarkets, r products, n persons
  - a_{ik} = price of product k at supermarket i
  - b_{kj} = quantity of product k bought by person i
  - $[\boldsymbol{AB}]_{ij}$ = how much the j-th person would pay when buying at the i-th supermarket
- E.g., covariance
  - If all columns of $\boldsymbol{A}_{m\times n}$ are centered ($\sum_k a_{ik} = 0$), then $\frac{1}{m-1}\boldsymbol{A}^T\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is the **sample covariance matrix**
  - $[\frac{1}{m-1}\boldsymbol{A}^T\boldsymbol{A}]_{ii}$ holds the sample variance of column i
  - $[\frac{1}{m-1}\boldsymbol{A}^T\boldsymbol{A}]_{ij}$ holds the sample covariance between columns i and j



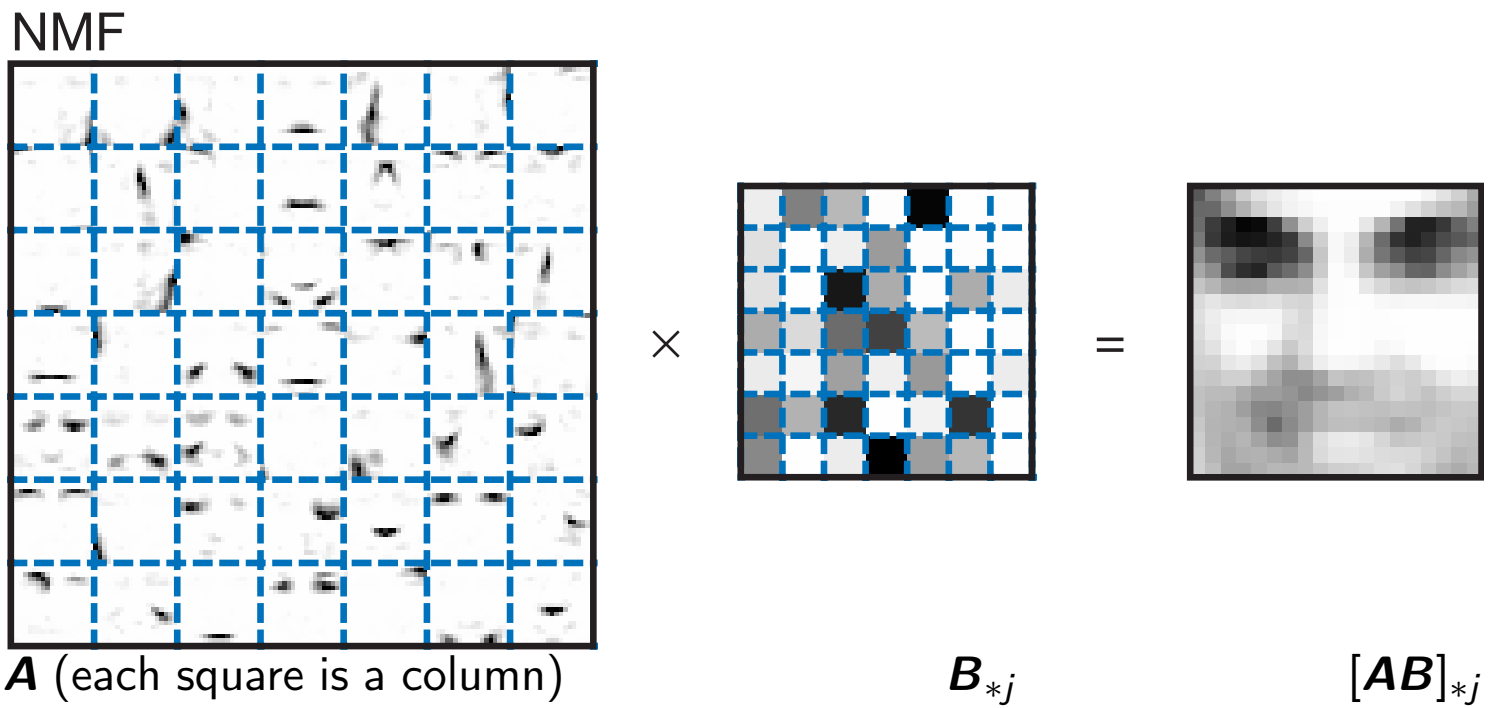
Interpretation for matrix multiplication (2)

We can also interpret rows i of $\boldsymbol{AB}$ as a linear combination of the rows of $\boldsymbol{B}$ with the coefficients coming from $\boldsymbol{A}$

$$[\boldsymbol{AB}]_{i*} = a_{i1}\boldsymbol{B}_{1*} + a_{i2}\boldsymbol{B}_{2*} + \cdots + a_{ir}\boldsymbol{B}_{r*},$$

and, similarly, the columns of $\boldsymbol{AB}$ as linear combinations of the columns of $\boldsymbol{A}$

$$[\boldsymbol{AB}]_{*j} = b_{1j}\boldsymbol{A}_{*1} + b_{2j}\boldsymbol{A}_{*2} + \cdots + b_{rj}\boldsymbol{A}_{*r}.$$

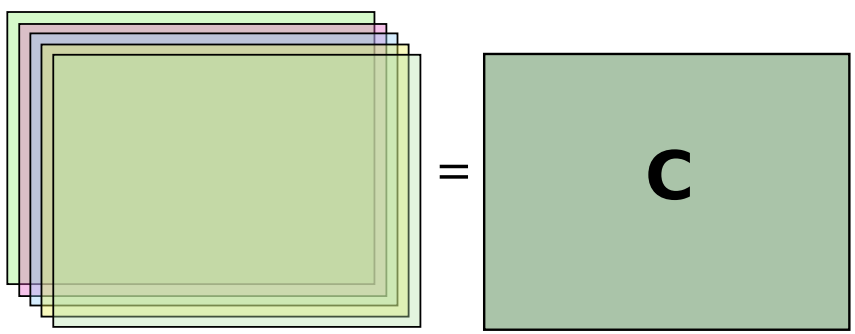


Interpretation for matrix multiplication (3)

We can view matrix $\boldsymbol{AB}$ as the sum of the r **component matrices** obtained by multiplying the k-th column of $\boldsymbol{A}$ and the k-th row of $\boldsymbol{B}$:

$$\boldsymbol{AB} = \boldsymbol{A}_{*1}\boldsymbol{B}_{1*} + \boldsymbol{A}_{*2}\boldsymbol{B}_{2*} + \cdots + \boldsymbol{A}_{*r}\boldsymbol{B}_{r*}$$

- Components $\boldsymbol{A}_{*k}\boldsymbol{B}_{k*}$ are outer products ($m \times n$ matrices)
- Note: when $\boldsymbol{u} \in \mathbb{R}^m$ and $\boldsymbol{v} \in \mathbb{R}^n$, the matrix product
  - $\boldsymbol{u}^T\boldsymbol{v}$ corresponds to a dot product (a scalar), $m = n$ required
  - $\boldsymbol{u}\boldsymbol{v}^T$ corresponds to an **outer product** (an $m \times n$ matrix)
- In our supermarket example
  - Components correspond to products
  - Entry (i, j) of k-th component indicates how much the j-th person would pay *for product k* when buying at the i-th supermarket



Transposes

The **matrix transpose** $\boldsymbol{A}^T$ switches rows and columns, i.e.,

$$[\boldsymbol{A}^T]_{ij} = a_{ji}.$$

The following properties hold

- $(\boldsymbol{A}^T)^T = \boldsymbol{A}$
- $(\boldsymbol{A} + \boldsymbol{B})^T = \boldsymbol{A}^T + \boldsymbol{B}^T$
- $(c\boldsymbol{A})^T = c\boldsymbol{A}^T$
- $(\boldsymbol{A}\boldsymbol{B})^T = \boldsymbol{B}^T\boldsymbol{A}^T$

Summing and scaling

Let $\boldsymbol{A} \in \mathbb{R}^{m \times n}$. Denote by $\mathbf{1}_n$ the all-ones vector of dimensionality n. For $s \in \mathbb{R}^n$, denote by $\operatorname{diag}(\boldsymbol{s})$ the $n \times n$ matrix with the entries of $\boldsymbol{s}$ on the main diagonal:

$$\operatorname{diag}(\boldsymbol{s}) = \begin{pmatrix} s_1 & 0 & \cdots & 0 \\ 0 & s_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & s_n \end{pmatrix}$$

- $\boldsymbol{A}\mathbf{1}_n$ computes the **row sums** of $\boldsymbol{A}$
- $\mathbf{1}_m^T \boldsymbol{A}$ computes the **column sums** of $\boldsymbol{A}$
- $\boldsymbol{A} \operatorname{diag}(\boldsymbol{c})$ **scales each column** j of $\boldsymbol{A}$ by c_j, $\boldsymbol{c} \in \mathbb{R}^n$
- $\operatorname{diag}(\boldsymbol{r}) \boldsymbol{A}$ **scales each row** i of $\boldsymbol{A}$ by r_i, $\boldsymbol{r} \in \mathbb{R}^m$

Matrices as linear maps

- A matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ is a **linear map** from $\mathbb{R}^n$ to $\mathbb{R}^m$
  - If $\boldsymbol{x} \in \mathbb{R}^n$, then $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} \in \mathbb{R}^m$ is the **image** of $\boldsymbol{x}$
  - $\boldsymbol{y} = \sum_{j=1}^{n} \boldsymbol{a}_j x_j$, i.e., a linear combination of the columns of $\boldsymbol{A}$
- If $\boldsymbol{A} \in \mathbb{R}^{m \times r}$ and $\boldsymbol{B} \in \mathbb{R}^{r \times n}$, then $\boldsymbol{A}\boldsymbol{B}$ maps from $\mathbb{R}^n$ to $\mathbb{R}^m$
  - Product $\boldsymbol{A}\boldsymbol{B}$ corresponds to composition of linear maps $\boldsymbol{A}$ and $\boldsymbol{B}$
- Square matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is **invertible** (= **nonsingular**) iff there is matrix $\boldsymbol{B} \in \mathbb{R}^{n \times n}$ such that $\boldsymbol{A}\boldsymbol{B} = \boldsymbol{I}$
  - Matrix $\boldsymbol{B}$ is the **inverse** of $\boldsymbol{A}$, denoted $\boldsymbol{A}^{-1}$
  - If $\boldsymbol{A}$ is invertible, then $\boldsymbol{A}\boldsymbol{A}^{-1} = \boldsymbol{A}^{-1}\boldsymbol{A} = \boldsymbol{I}$
    - $\boldsymbol{A}\boldsymbol{A}^{-1}\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{A}\boldsymbol{x} = \boldsymbol{x}$
  - Non-square matrices do not have (general) inverses but can have **left** or **right inverses**: $\boldsymbol{A}\boldsymbol{R} = \boldsymbol{I}$ or $\boldsymbol{L}\boldsymbol{A} = \boldsymbol{I}$
- The **transpose** of $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ is a linear map $\boldsymbol{A}^T : \mathbb{R}^m \to \mathbb{R}^n$
  - $(\boldsymbol{A}^T)_{ij} = \boldsymbol{A}_{ji}$
  - Generally, transpose is **not** the inverse ($\boldsymbol{A}\boldsymbol{A}^T \neq \boldsymbol{I}$)

Matrix norms

- Matrix norms measure the magnitude of the matrix
  - Magnitude of the values in the matrix
  - Magnitude of the image
- **Operator norms** measure how large the image of an unit vector can get
  - Induced by a vector norm
  - For $p \geq 1$, $\|\boldsymbol{A}\|_p = \max\{\|\boldsymbol{A}\boldsymbol{x}\|_p \mid \|\boldsymbol{x}\|_p = 1\}$
  - $\|\boldsymbol{A}\|_1 =$ maximum sum of absolute values of a column
  - $\|\boldsymbol{A}\|_\infty =$ maximum sum of absolute values of a row
  - **Spectral norm**: $\|\boldsymbol{A}\|_2 =$ largest singular value of $\boldsymbol{A}$ (more later)
- The **Frobenius norm** is the vector-L_2 norm applied to matrices (treating them as a vector)
  - $\|\boldsymbol{A}\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}^2}$
  - Note: $\|\boldsymbol{A}\|_F \neq \|\boldsymbol{A}\|_2$ (but sometimes Frobenius norm is referred to as L_2 norm)

Matrix rank and linear independence

- A vector $\boldsymbol{u} \in \mathbb{R}^n$ is **linearly dependent** on set of vectors $V = \{\boldsymbol{v}_i\} \subset \mathbb{R}^n$ if $\boldsymbol{u}$ can be expressed as a linear combination of vectors in V
  - $\boldsymbol{u} = \sum_i a_i \boldsymbol{v}_i$ for some $a_1, \ldots, a_n \in \mathbb{R}$
  - Set V is linearly dependent if some $\boldsymbol{v}_i \in V$ is linearly dependent on $V \setminus \{\boldsymbol{v}_i\}$
  - If V is not linearly dependent, it is **linearly independent**
- The **column rank** of matrix $\boldsymbol{A}$ is the maximum number of linearly independent columns of $\boldsymbol{A}$
- The **row rank** of $\boldsymbol{A}$ is the maximum number of linearly independent rows of $\boldsymbol{A}$
- The **Schein rank** of $\boldsymbol{A}$ is the least integer r such that $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{R}$ for some $\boldsymbol{L} \in \mathbb{R}^{m \times r}$ and $\boldsymbol{R} \in \mathbb{R}^{r \times n}$
  - Equivalently, the least r such that $\boldsymbol{A}$ is a sum of r vector outer products
- **All these ranks are equivalent**
  - E.g., matrix has rank 1 iff it is an outer product of two (non-zero) vectors

Matrices as systems of linear equations

- A matrix can hold the coefficients of a system of linear equations (c.f. Chinese *Nine Chapters on Arithmetic*)

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ &\vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &= b_m \end{aligned} \quad \Leftrightarrow \quad \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}$$

- If the coefficient matrix $\boldsymbol{A}$ is invertible, the system has exact solution $\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b}$
- If $m < n$ the system is **underdetermined** and can have an infinite number of solutions
- If $m > n$ the system is **overdetermined** and (usually) does not have an exact solution
- The **least-squares** solution is the vector $\boldsymbol{x}$ that minimizes $\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2$ (cf. linear regression)

Special types of matrices

- The **diagonals** of matrix $\boldsymbol{A}$ go from top-left to bottom-right
  - The main diagonal contains the elements a_{ii}
  - The k-th upper diagonal contains the elements $a_{i,(i+k)}$
  - The k-th lower diagonal contains the elements $a_{(i+k),i)}$
  - The **anti-diagonals** go from top-right to bottom-left
- Matrix is **diagonal** if all its non-zero values are in a diagonal (typically main diagonal)
  - **Bi-diagonal** matrices have values in two diagonals, etc.
- Matrix $\boldsymbol{A}$ is **upper (right) triangular** if all of its non-zeros are in or above the main diagonal
  - **Lower (left) triangular** matrices have all non-zeros in or below main diagonal
  - Upper left and lower right triangular matrices: replace diagonal with anti-diagonal
- A square matrix $\boldsymbol{P}$ is **permutation matrix** if each row and each column of $\boldsymbol{P}$ has exactly one 1 and rest are 0s
  - If $\boldsymbol{P}$ is a permutation matrix, $\boldsymbol{PA}$ permutes the order of the rows and $\boldsymbol{AP}$ the order of the columns of $\boldsymbol{A}$

Orthogonal matrices

- A set $V = \{\boldsymbol{v}_i\} \subset \mathbb{R}^n$ is **orthogonal** if all vectors in V are mutually orthogonal
  - $\boldsymbol{v} \cdot \boldsymbol{u} = 0$ for all $\boldsymbol{v} \neq \boldsymbol{u} \in V$
  - If all vectors in V also have unit norm ($\|\boldsymbol{v}\|_2 = 1$), V is **orthonormal**
- A square matrix $\boldsymbol{A}$ is **orthogonal** if its columns are a set of orthonormal (!) vectors or equivalently
  - Its rows are orthonormal
  - $\boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{I}_n$
  - $\boldsymbol{A}^{-1} = \boldsymbol{A}^T$
- An $m \times n$ matrix $\boldsymbol{A}$ is
  - **column-orthogonal** if columns are a set of orthonormal vectors (only possible if $m \geq n$); then $\boldsymbol{A}^T$ is left inverse ($\boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{I}_n$)
  - **row-orthogonal** if rows are a set of orthonormal vectors (only possible if $m \leq n$); then $\boldsymbol{A}^T$ is right inverse ($\boldsymbol{A}\boldsymbol{A}^T = \boldsymbol{I}_m$)
- If $\boldsymbol{A}$ and $\boldsymbol{B}$ are orthogonal, so is $\boldsymbol{AB}$
  - Similarly: column-orthogonality and row-orthogonality is preserved

Outline

Lessons learned

- Many uses, many interpretations
  - Vectors
  - Matrices
  - Dot products
  - Matrix products
- Magnitudes and distances are measured by norms
- Basic concepts of linear algebra
- Special types of matrices: diagonal, triangular, orthogonal

Suggested reading

- Any (elementary) linear algebra text book
  - Carl Meyer
    Matrix Analysis and Applied Linear Algebra
    Society for Industrial and
    Applied Mathematics, 2000
    http://www.matrixanalysis.com
- Wolfram MathWorld articles
- Wikipedia articles

Data Mining and Matrices

03 – Singular Value Decomposition

Prof. Dr. Rainer Gemulla

Universität Mannheim

FSS 2018

The SVD is the Swiss Army knife of matrix decompositions.

—Diane O'Leary, 2006

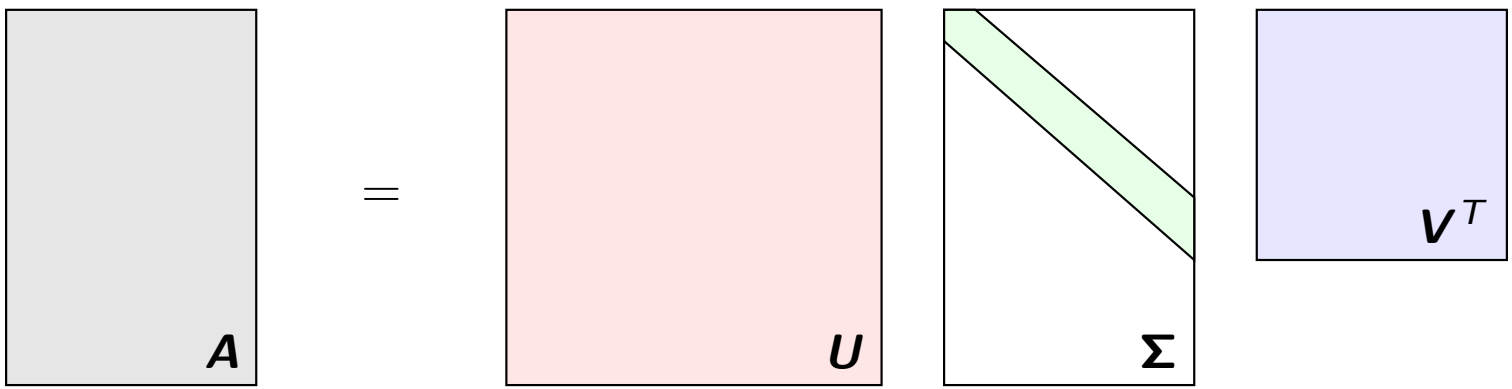
Outline

Definition

Theorem

For each $\boldsymbol{A} \in \mathbb{R}^{m \times n}$*, there are orthogonal matrices* $\boldsymbol{U}_{m \times m}$*,* $\boldsymbol{V}_{n \times n}$*, and a diagonal matrix* $\boldsymbol{\Sigma}_{m \times n}$ *with values* $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{\min(m,n)} \geq 0$ *on the main diagonal such that* $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$*.*

- $\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ is called the **singular value decomposition** (SVD) of $\boldsymbol{A}$
- Values σ_i are the **singular values** of $\boldsymbol{A}$
- Columns of $\boldsymbol{U}$ are the **left singular vectors** of $\boldsymbol{A}$
- Columns of $\boldsymbol{V}$ are the **right singular vectors** of $\boldsymbol{A}$



Outline

Characterization of the four fundamental subspaces

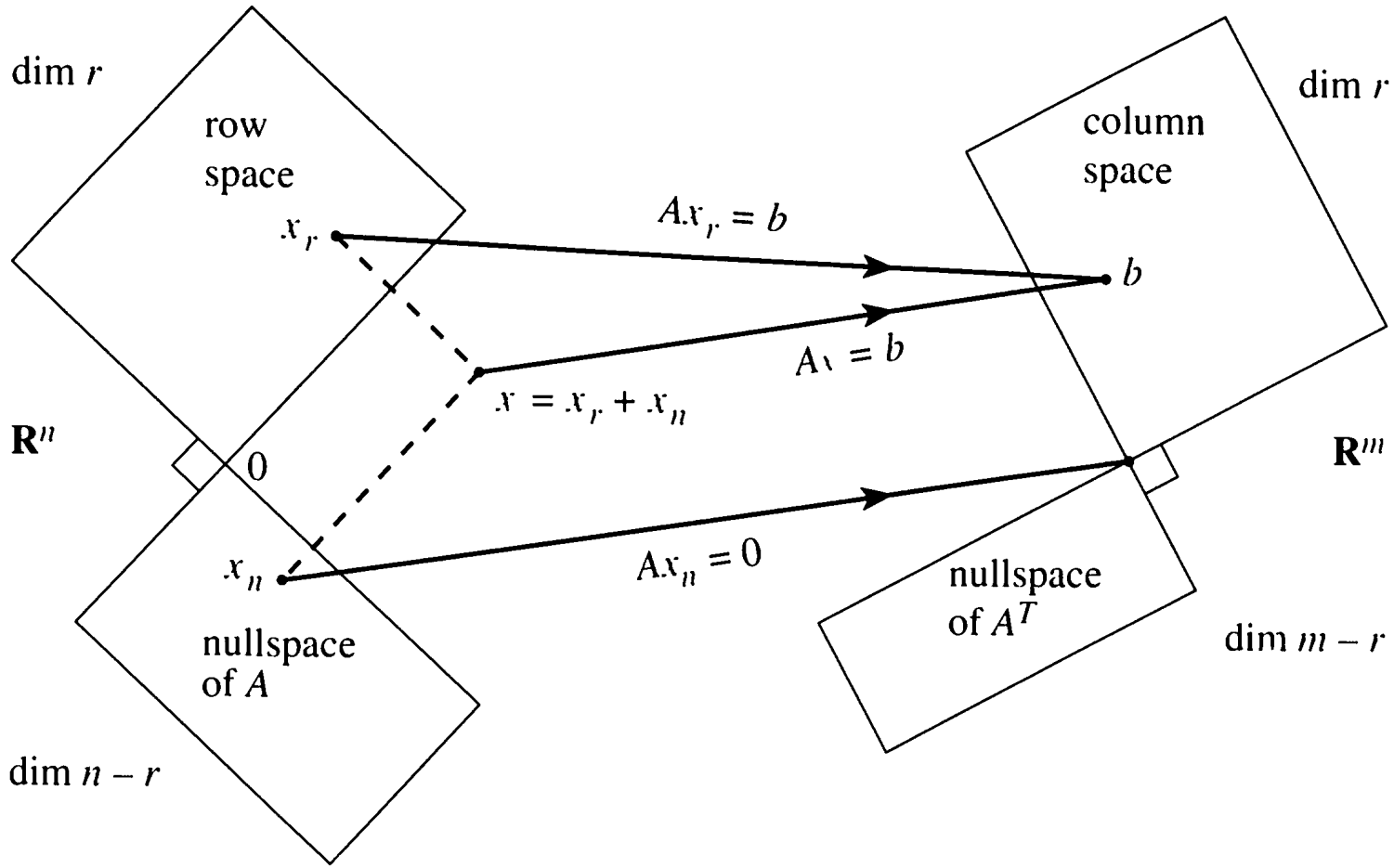
The **fundamental theorem of linear algebra** states that every matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ induces four fundamental subspaces:

- The **column space** (range, image) of dimension $\text{rank}(\boldsymbol{A}) = r$
  - The set of all possible linear combinations of columns of $\boldsymbol{A}$
- The **nullspace** (kernel) of dimension $n - r$
  - The set of all vectors $\boldsymbol{x} \in \mathbb{R}^n$ for which $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{0}$
- The **row space** (coimage) of dimension r
- The **left nullspace** (cokernel) of dimension $m - r$

Explicit bases for these subspaces can be obtained from the SVD:

- Column space: the first r columns of $\boldsymbol{U}$
- Null space: the last $(n - r)$ columns of $\boldsymbol{V}$
- Row space: the first r columns of $\boldsymbol{V}$
- Left null space: the last $(m - r)$ columns of $\boldsymbol{U}$

The four fundamental subspaces



The action of $\boldsymbol{A}$. Row space to column space, nullspace to zero.

Pseudo-inverse

Problem.

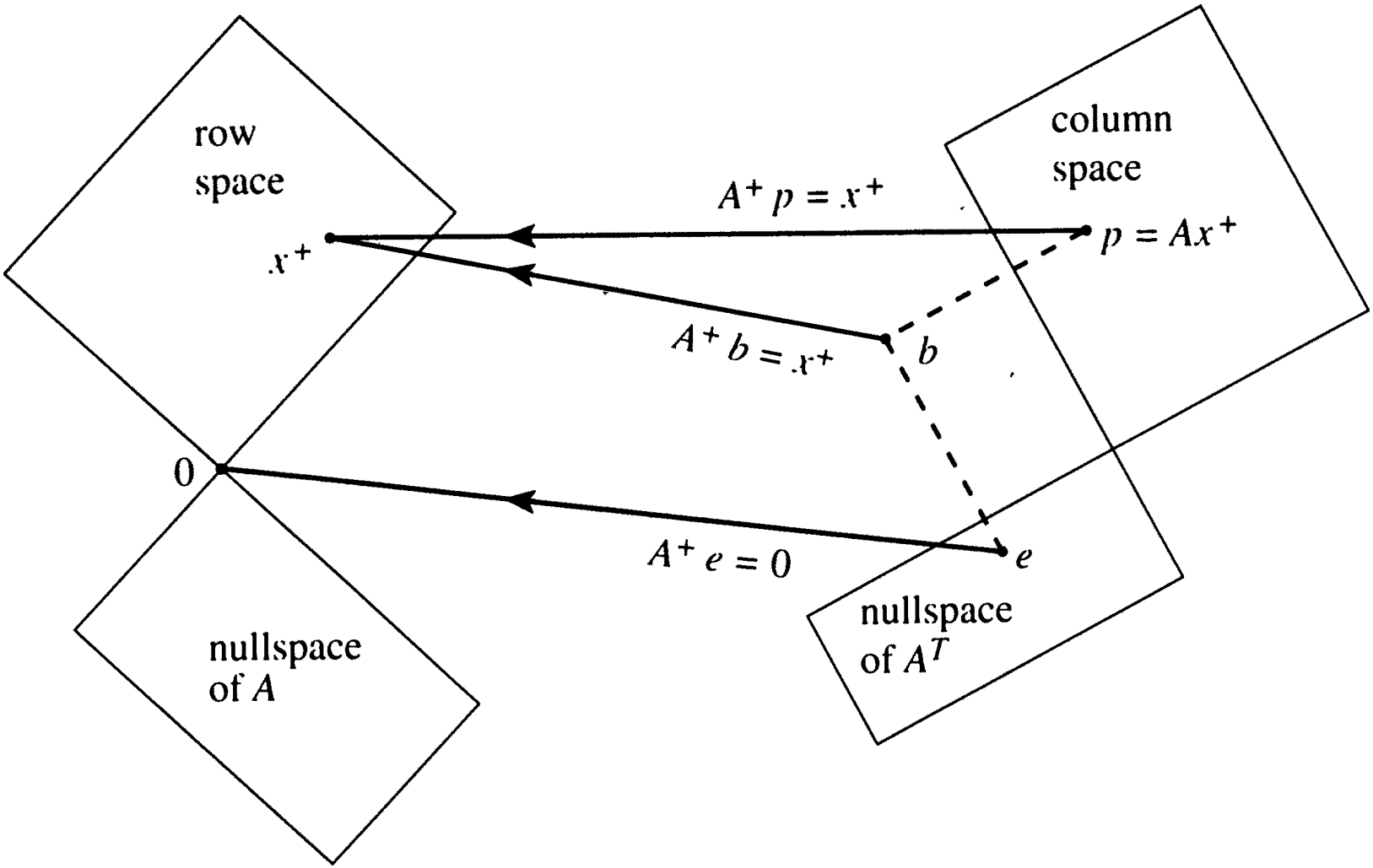
Given $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{b} \in \mathbb{R}^m$, find $\boldsymbol{x} \in \mathbb{R}^n$ minimizing $\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2$.

- If $\boldsymbol{A}$ is invertible, the solution is $\boldsymbol{A}^{-1}\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b} \Leftrightarrow \boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b}$
- A **pseudo-inverse** $\boldsymbol{A}^+$ captures some properties of inverse $\boldsymbol{A}^{-1}$
- The **Moose–Penrose pseudo-inverse** of $\boldsymbol{A}$ is a matrix $\boldsymbol{A}^+ \in \mathbb{R}^{n \times m}$ satisfying the following criteria
  - $\boldsymbol{A}\boldsymbol{A}^+\boldsymbol{A} = \boldsymbol{A}$ (but it is possible that $\boldsymbol{A}\boldsymbol{A}^+ \neq \boldsymbol{I}$)
  - $\boldsymbol{A}^+\boldsymbol{A}\boldsymbol{A}^+ = \boldsymbol{A}^+$ (cf. above)
  - $(\boldsymbol{A}\boldsymbol{A}^+)^T = \boldsymbol{A}\boldsymbol{A}^+$ ($\boldsymbol{A}\boldsymbol{A}^+$ is symmetric)
  - $(\boldsymbol{A}^+\boldsymbol{A})^T = \boldsymbol{A}^+\boldsymbol{A}$ (as is $\boldsymbol{A}^+\boldsymbol{A}$)
- If $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ is the SVD of $\boldsymbol{A}$, then $\boxed{\boldsymbol{A}^+ = \boldsymbol{V}\boldsymbol{\Sigma}^+\boldsymbol{U}^T}$
  - $\boldsymbol{\Sigma}^+$ replaces each $\sigma_i > 0$ by $1/\sigma_i$ and transposes

Theorem.

An optimum solution for the above problem is $\boldsymbol{x} = \boldsymbol{A}^+\boldsymbol{b}$.

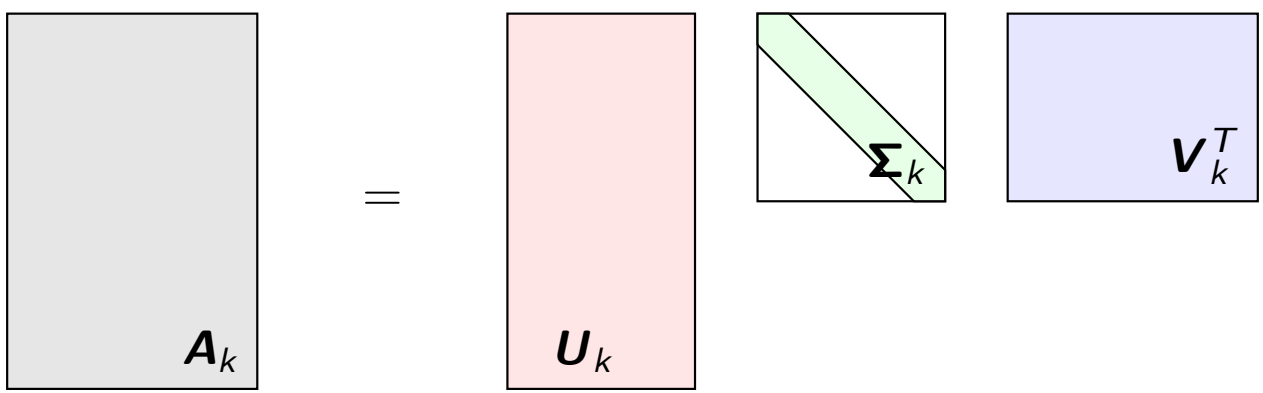
Pseudo inverse (illustration)



The inverse of $\boldsymbol{A}$ is (where possible) the pseudo-inverse $\boldsymbol{A}^+$.

Truncated SVD

- The rank of the matrix is the number of its non-zero singular values
  - Easy to see by writing $\boldsymbol{A} = \sum_{j=1}^{\min\{n,m\}} \sigma_j \boldsymbol{u}_j \boldsymbol{v}_j^T$
- The **truncated SVD** only takes the first k columns of $\boldsymbol{U}$ and $\boldsymbol{V}$ and the main $k \times k$ submatrix of $\boldsymbol{\Sigma}$
  - $\boldsymbol{A}_k = \sum_{j=1}^{k} \sigma_j \boldsymbol{u}_j \boldsymbol{v}_j^T = \boldsymbol{U}_k \boldsymbol{\Sigma}_k \boldsymbol{V}_k^T$
  - $\operatorname{rank}(\boldsymbol{A}_k) = k$ (if $\sigma_k > 0$)
  - $\boldsymbol{U}_k$ and $\boldsymbol{V}_k$ are not orthogonal anymore, but they are **column-orthogonal**
- If $k = \min\{m, n\}$, then $\boldsymbol{A}_k = \boldsymbol{A}$; called **thin SVD**
- If $k = \operatorname{rank}(\boldsymbol{A})$, then $\boldsymbol{A}_k = \boldsymbol{A}$; called **compact SVD**
- If $k < \operatorname{rank}(\boldsymbol{A})$, then $\boldsymbol{A}_k$ is **low-rank approximation** of $\boldsymbol{A}$



SVD and matrix norms

Let $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ be the SVD of $\boldsymbol{A}$. Then

- $\|\boldsymbol{A}\|_F^2 = \sum_{i=1}^{\min\{n,m\}} \sigma_i^2$
- $\|\boldsymbol{A}\|_2 = \sigma_1$
  - Remember: $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{\min\{n,m\}} \geq 0$
- Therefore $\|\boldsymbol{A}\|_2 \leq \|\boldsymbol{A}\|_F \leq \sqrt{n}\,\|\boldsymbol{A}\|_2$
- Sq. Frobenius norm of truncated SVD is $\|\boldsymbol{A}_k\|_F^2 = \sum_{i=1}^{k} \sigma_i^2$
  - And of the "approximation error" $\|\boldsymbol{A} - \boldsymbol{A}_k\|_F^2 = \sum_{i=k+1}^{\min\{n,m\}} \sigma_i^2$

The Eckart–Young theorem

Let $\boldsymbol{A}_k$ be the rank-k truncated SVD of $\boldsymbol{A}$. Then $\boldsymbol{A}_k$ is the rank-k matrix closest to $\boldsymbol{A}$ in the Frobenius sense. That is

$$\|\boldsymbol{A} - \boldsymbol{A}_k\|_F \leq \|\boldsymbol{A} - \boldsymbol{B}\|_F \quad \text{for all rank-}k\text{ matrices } \boldsymbol{B}.$$

Eigendecompositions

- An **eigenvector** of a square matrix $\boldsymbol{A}$ is a vector $\boldsymbol{v}$ such that $\boldsymbol{A}$ only changes the magnitude of $\boldsymbol{v}$
  - I.e. $\boldsymbol{A}\boldsymbol{v} = \lambda\boldsymbol{v}$ for some $\lambda \in \mathbb{R}$
  - Such λ is an **eigenvalue** of $\boldsymbol{A}$
  - Try it!
- The **eigendecomposition** of $\boldsymbol{A}$ is $\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{\Delta}\boldsymbol{Q}^{-1}$
  - The columns of $\boldsymbol{Q}$ are the eigenvectors of $\boldsymbol{A}$
  - Matrix $\boldsymbol{\Delta}$ is a diagonal matrix with the eigenvalues
- Not every (square) matrix has eigendecomposition
  - If $\boldsymbol{A}$ is of form $\boldsymbol{B}\boldsymbol{B}^T$, it always has eigendecomposition
- The SVD of $\boldsymbol{A}$ is closely related to the eigendecompositions of $\boldsymbol{A}\boldsymbol{A}^T$ and $\boldsymbol{A}^T\boldsymbol{A}$
  - The left singular vectors are the eigenvectors of $\boldsymbol{A}\boldsymbol{A}^T$
  - The right singular vectors are the eigenvectors of $\boldsymbol{A}^T\boldsymbol{A}$
  - The singular values are the square roots of the eigenvalues of both $\boldsymbol{A}\boldsymbol{A}^T$ and $\boldsymbol{A}^T\boldsymbol{A}$

Outline

Factor interpretation

- The most common way to interpret SVD is to consider the columns of $\boldsymbol{U}$ (or $\boldsymbol{V}$)
  - Let $\boldsymbol{A}$ be objects-by-attributes and $\boldsymbol{U\Sigma V}^T$ its SVD
  - If two rows have similar values in a column of $\boldsymbol{U}$, these objects are somehow similar
  - If two columns have similar values in a row of $\boldsymbol{V}^T$, these attributes are somehow similar
  - In both cases, first entries often matter most $\rightarrow$ truncated SVD

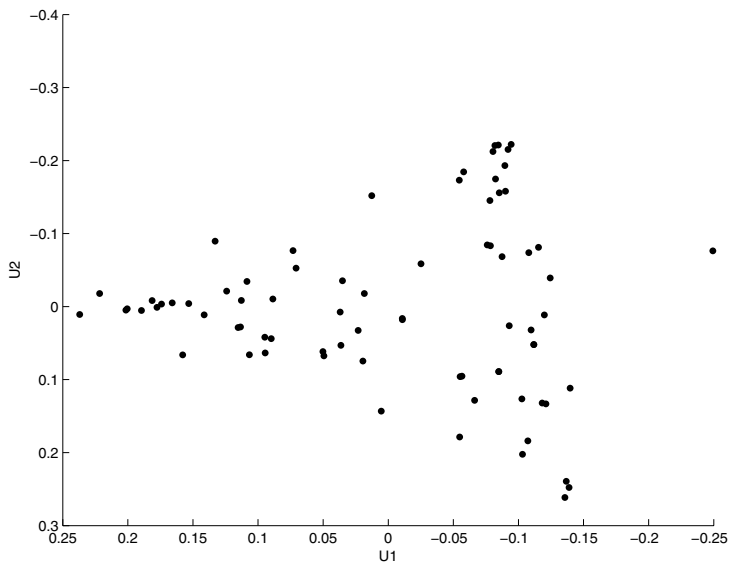


Figure 3.2. *The first two factors for a dataset ranking wines.*

- Example: people's ratings of different wines
- Scatterplot of first and second column of $\boldsymbol{U}$
  - left: likes wine
  - right: doesn't like
  - up: prefers red wine
  - bottom: prefers white vine
- Conclusion: winelovers like red and white, others care

Example: Weather data ($k = 2$)

1.00			9.05	16.55	26.73	18.75	17.81
	1.00		-4.14	0.27	2.32	-0.89	-0.69
			Jan	Apr	Jul	Oct	Year
0.62	1.69	Stockholm	-0.70	8.60	21.90	9.90	10.00
0.69	2.11	Minsk	-2.10	12.20	23.60	10.20	10.60
0.83	0.00	London	7.90	13.30	22.80	15.20	14.80
0.90	1.52	Budapest	1.20	16.30	26.50	16.10	15.00
0.88	0.30	Paris	6.90	14.70	24.40	15.80	15.50
0.98	1.59	Bucharests	1.50	18.00	28.80	18.00	16.50
1.09	-0.66	Barcelona	12.40	17.60	27.50	21.50	20.00
1.14	-0.31	Rome	11.90	17.70	30.30	21.40	20.40
1.16	-1.09	Lisbon	14.80	19.80	27.90	22.50	21.50
1.24	-0.35	Athens	12.90	20.30	32.60	23.10	22.30
1.21	-1.26	Valencia	16.10	20.20	29.10	23.60	22.30
1.27	-1.12	Malta	16.10	20.00	31.50	25.20	23.20
							$\boldsymbol{D}$

(RMSE: 0.60)

Thin SVD of Weather data ($\boldsymbol{U}$)

$$\boldsymbol{U} = \begin{array}{c} \\ \textit{Stockholm} \\ \textit{Minsk} \\ \textit{London} \\ \textit{Budapest} \\ \textit{Paris} \\ \textit{Bucharests} \\ \textit{Barcelona} \\ \textit{Rome} \\ \textit{Lisbon} \\ \textit{Athens} \\ \textit{Valencia} \\ \textit{Malta} \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{pmatrix} 0.18 & 0.41 & 0.61 & 0.28 & -0.32 \\ 0.19 & 0.51 & 0.08 & -0.54 & 0.40 \\ 0.24 & 0.00 & 0.20 & -0.15 & 0.04 \\ 0.25 & 0.37 & -0.39 & 0.18 & -0.10 \\ 0.25 & 0.07 & 0.05 & -0.25 & -0.22 \\ 0.28 & 0.39 & -0.49 & 0.30 & 0.08 \\ 0.31 & -0.16 & -0.01 & 0.33 & -0.26 \\ 0.32 & -0.07 & 0.30 & 0.10 & 0.07 \\ 0.33 & -0.27 & -0.23 & -0.27 & -0.46 \\ 0.35 & -0.08 & 0.10 & -0.23 & -0.09 \\ 0.34 & -0.31 & -0.12 & -0.21 & 0.17 \\ 0.36 & -0.27 & 0.12 & 0.37 & 0.59 \end{pmatrix} \end{array}$$

Thin SVD of Weather data ($\mathbf{\Sigma}$)

$$\mathbf{\Sigma} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{pmatrix} 147.55 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 20.09 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 4.25 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 1.77 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.32 \end{pmatrix} \end{array}$$

Thin SVD of Weather data ($\boldsymbol{V}$)

$$\boldsymbol{V} = \begin{array}{c} \\ \textit{Jan} \\ \textit{Apr} \\ \textit{Jul} \\ \textit{Oct} \\ \textit{Year} \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{pmatrix} 0.22 & -0.85 & 0.31 & -0.30 & 0.21 \\ 0.40 & 0.06 & -0.74 & -0.52 & 0.17 \\ 0.64 & 0.47 & 0.54 & -0.16 & 0.21 \\ 0.45 & -0.18 & -0.25 & 0.78 & 0.30 \\ 0.43 & -0.14 & -0.03 & 0.05 & -0.8 \end{pmatrix} \end{array}$$

Example: Weather data ($k = 2$), SVD

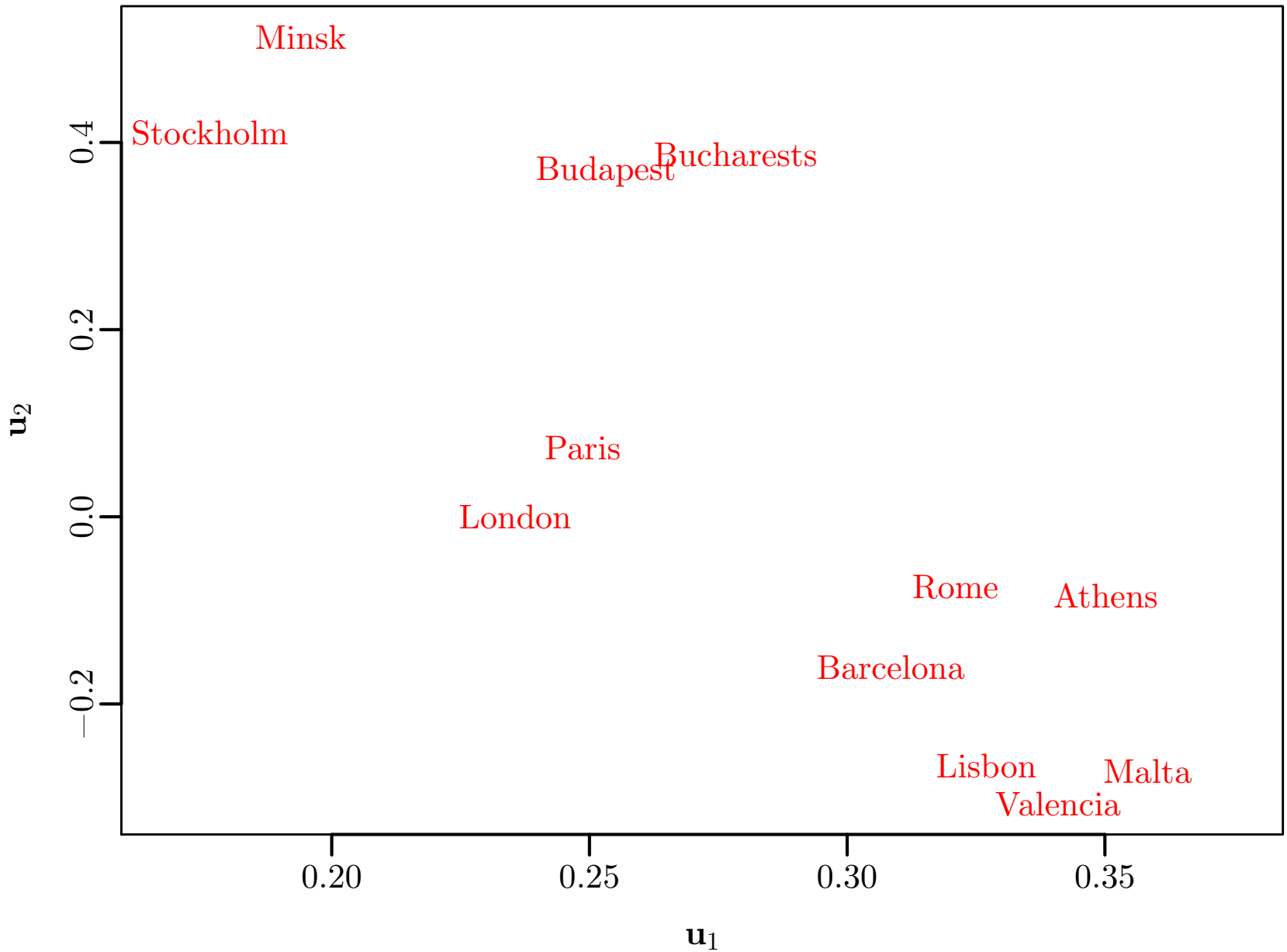
147.54			0.22	0.40	0.64	0.45	0.43
	20.09		-0.85	0.06	0.47	-0.18	-0.14

			Jan	Apr	Jul	Oct	Year
0.18	0.41	Stockholm	-0.70	8.60	21.90	9.90	10.00
0.19	0.51	Minsk	-2.10	12.20	23.60	10.20	10.60
0.24	0.00	London	7.90	13.30	22.80	15.20	14.80
0.25	0.37	Budapest	1.20	16.30	26.50	16.10	15.00
0.25	0.07	Paris	6.90	14.70	24.40	15.80	15.50
0.28	0.39	Bucharests	1.50	18.00	28.80	18.00	16.50
0.31	-0.16	Barcelona	12.40	17.60	27.50	21.50	20.00
0.32	-0.07	Rome	11.90	17.70	30.30	21.40	20.40
0.33	-0.27	Lisbon	14.80	19.80	27.90	22.50	21.50
0.35	-0.08	Athens	12.90	20.30	32.60	23.10	22.30
0.34	-0.31	Valencia	16.10	20.20	29.10	23.60	22.30
0.36	-0.27	Malta	16.10	20.00	31.50	25.20	23.20

$\boldsymbol{D}$

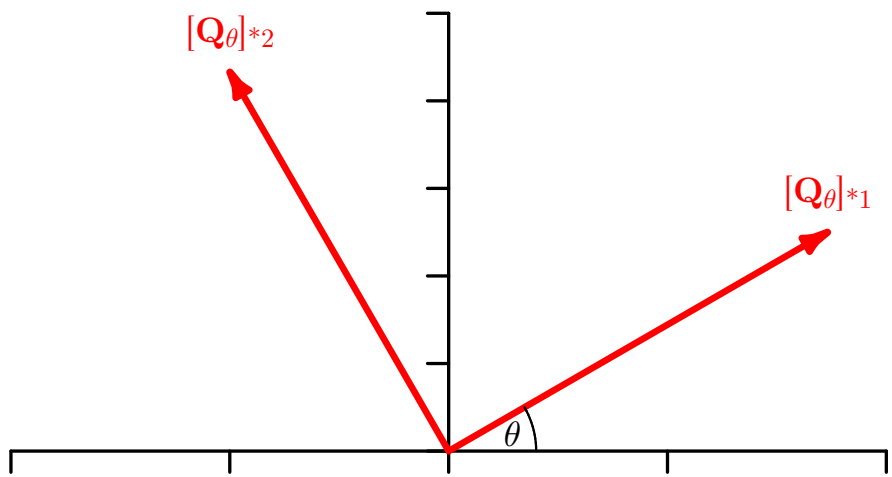
(RMSE: 0.60)

Example: Weather data ($r = 2$), SVD plot



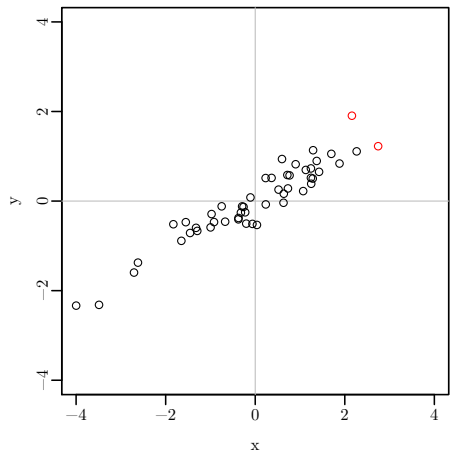
Orthogonal matrices and rotations

- Orthogonal matrices are **rotation matrices**
- Consider orthogonal matrix $\boldsymbol{Q}$
- Inner products are retained: $(\boldsymbol{Q}\boldsymbol{x})^T(\boldsymbol{Q}\boldsymbol{y}) = \boldsymbol{x}^T\boldsymbol{Q}^T\boldsymbol{Q}\boldsymbol{y} = \boldsymbol{x}^T\boldsymbol{y}$
- Thus Euclidean norms also retained: $\|\boldsymbol{Q}\boldsymbol{x}\| = \|\boldsymbol{x}\|$
- Implies that all angles are retained
- In 2D: $\boldsymbol{Q}_\theta = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$
  - Consider vector $\begin{pmatrix} x \\ y \end{pmatrix} = x\boldsymbol{e}_1 + y\boldsymbol{e}_2$
  - $\boldsymbol{Q}_\theta \begin{pmatrix} x \\ y \end{pmatrix} = x[\boldsymbol{Q}_\theta]_{*1} + y[\boldsymbol{Q}_\theta]_{*2}$
- Thus: the columns of $\boldsymbol{Q}$ form "new axes" for rotation $\boldsymbol{Q}\boldsymbol{v}$
- Similarly: rows of $\boldsymbol{Q}$ form "new axes" for rotation $\boldsymbol{v}^T\boldsymbol{Q}$ (rotates backwards)

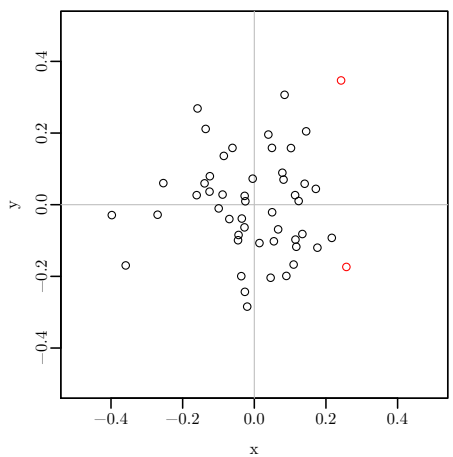


Geometric interpretation (1)

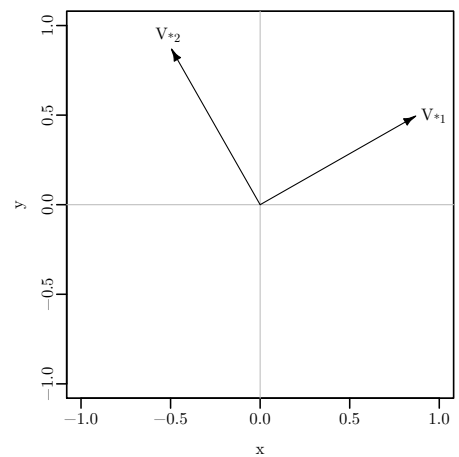
$$\boldsymbol{D}_{50\times 2}$$



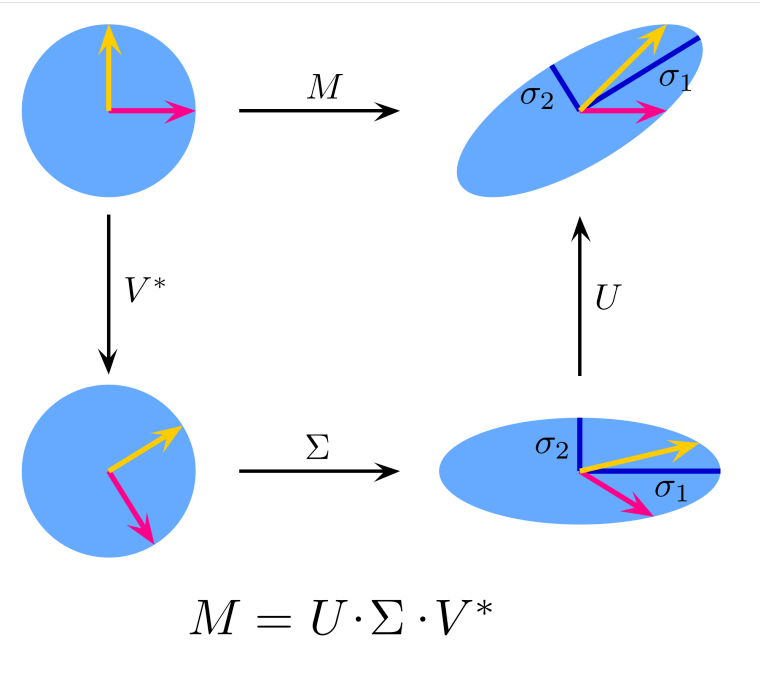
$$\boldsymbol{U}_r \qquad \boldsymbol{\Sigma}_r \qquad \boldsymbol{V}_r^T$$



$$\begin{pmatrix} 11.64 & 0 \\ 0 & 1.69 \end{pmatrix}$$



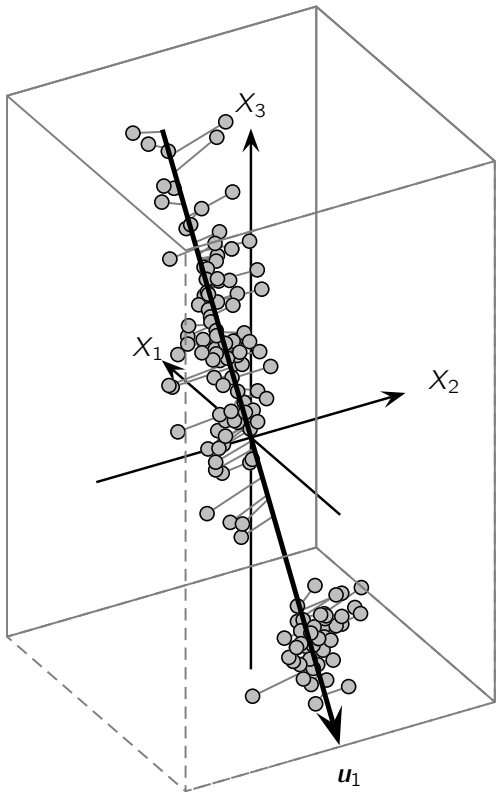
Geometric interpretation (2)



- Let $\boldsymbol{U\Sigma V}^T$ be the SVD of $\boldsymbol{M}$
- SVD shows that every linear mapping $\boldsymbol{y} = \boldsymbol{Mx}$ can be considered as a series of rotation, stretching, and rotation operations
  - Matrix $\boldsymbol{V}^T$ performs the first rotation $\boldsymbol{y}_1 = \boldsymbol{V}^T\boldsymbol{x}$
  - Matrix $\boldsymbol{\Sigma}$ performs the stretching $\boldsymbol{y}_2 = \boldsymbol{\Sigma y}_1$
  - Matrix $\boldsymbol{U}$ performs the second rotation $\boldsymbol{y} = \boldsymbol{U y}_2$

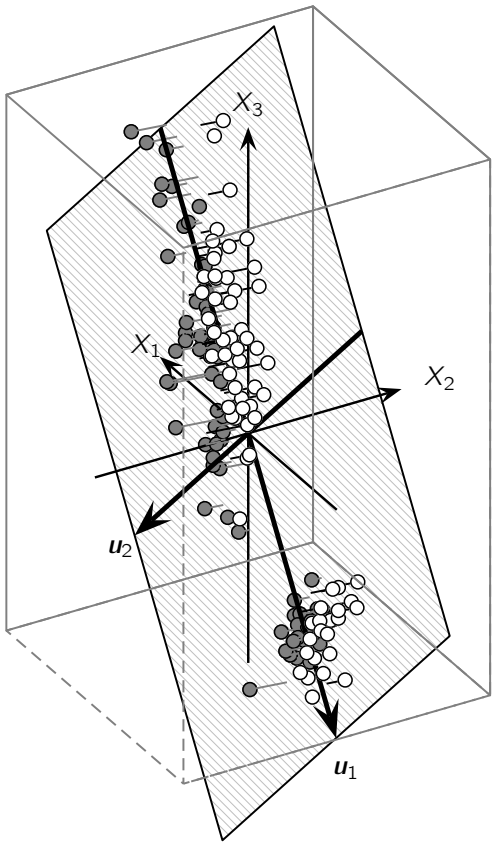
Wikipedia user Georg-Johann

Dimension of largest variance (1)



- The singular vectors give the dimensions of the variance in the data
  - The first singular vector is the dimension of the largest variance

Dimension of largest variance (2)

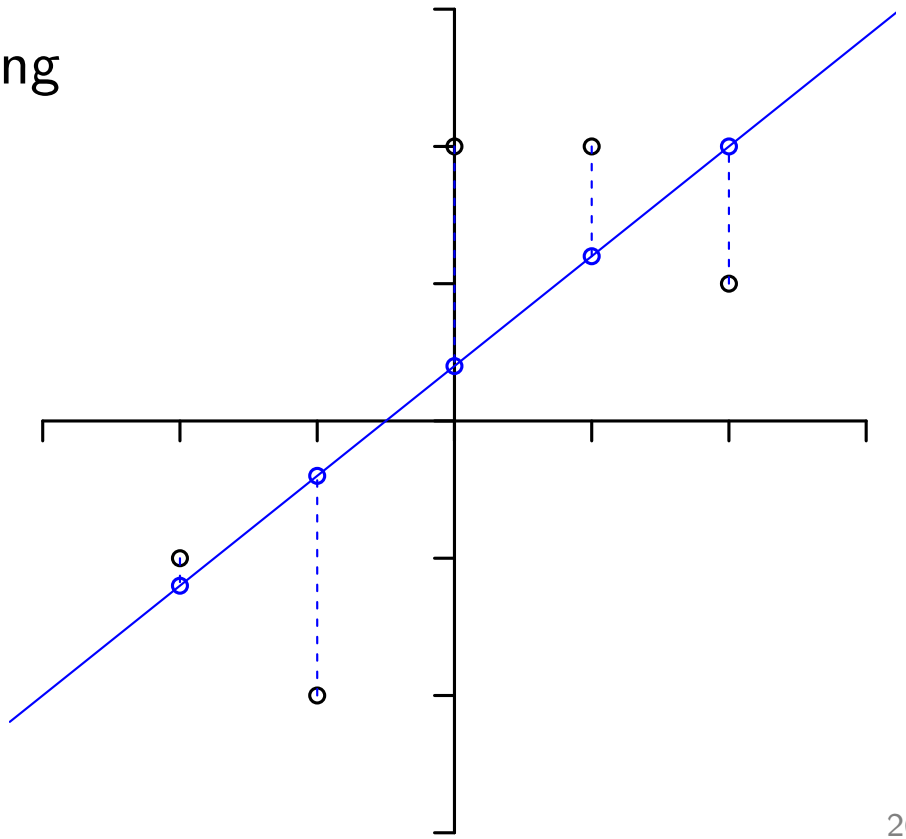


- The singular vectors give the dimensions of the variance in the data
  - The first singular vector is the dimension of the largest variance
  - The second singular vector is the orthogonal dimension of the second largest variance
    - First two dimensions span a hyperplane
- From Eckart–Young we know that if we project the data to the spanned hyperplanes, the distance of the projection is minimized

SVD and linear regression (1)

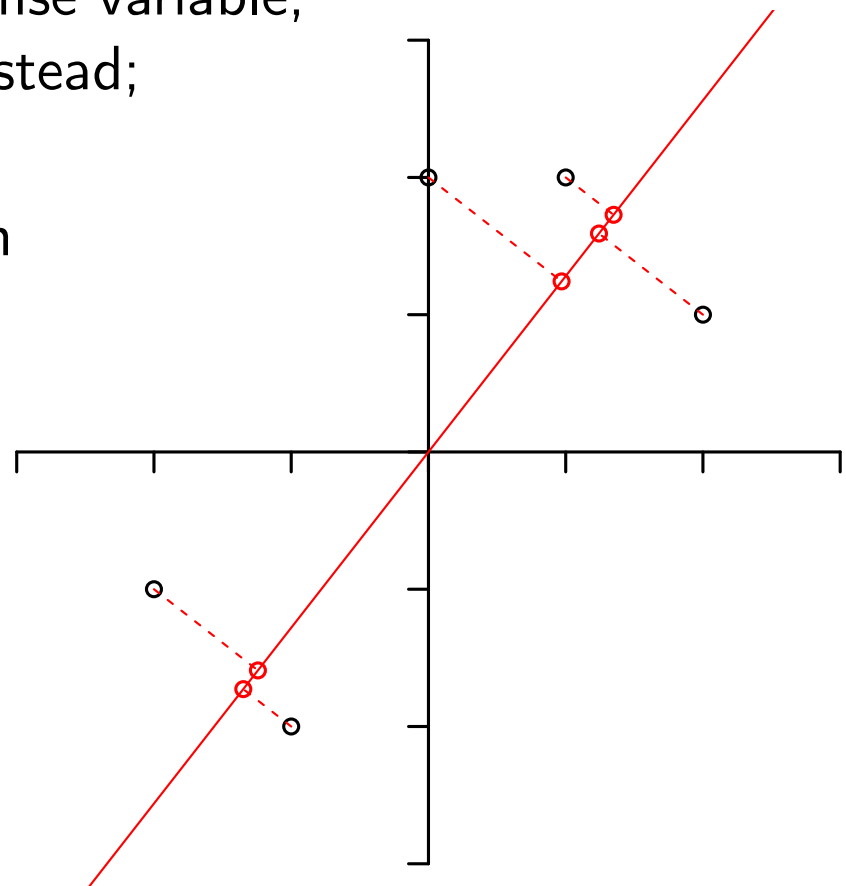
Consider the 2-dimensional case.

- When fitting a linear regression model, we want to predict the value of one distinguished variable (the response variable) given the others (the explanatory variable)
- Say y is the response variable, x the explanatory variable
- We obtain a coefficient vector $\boldsymbol{\beta} = \begin{pmatrix}\beta_0 & \beta_1\end{pmatrix}^T \in \mathbb{R}^2$, consisting of offset β_0 and slope β_1
- Prediction is $\hat{y} = \boldsymbol{\beta} \cdot \begin{pmatrix}1 & x\end{pmatrix}^T$, the error is $(y - \hat{y})^2$ (least squares)
- Goal is to minimize this error, i.e., $\|\boldsymbol{y} - \hat{\boldsymbol{y}}\|_2$
- Set $\boldsymbol{X} = \begin{pmatrix}\boldsymbol{1} & \boldsymbol{x}\end{pmatrix}$, then $\boldsymbol{\beta} = \boldsymbol{X}^+ \boldsymbol{y}$

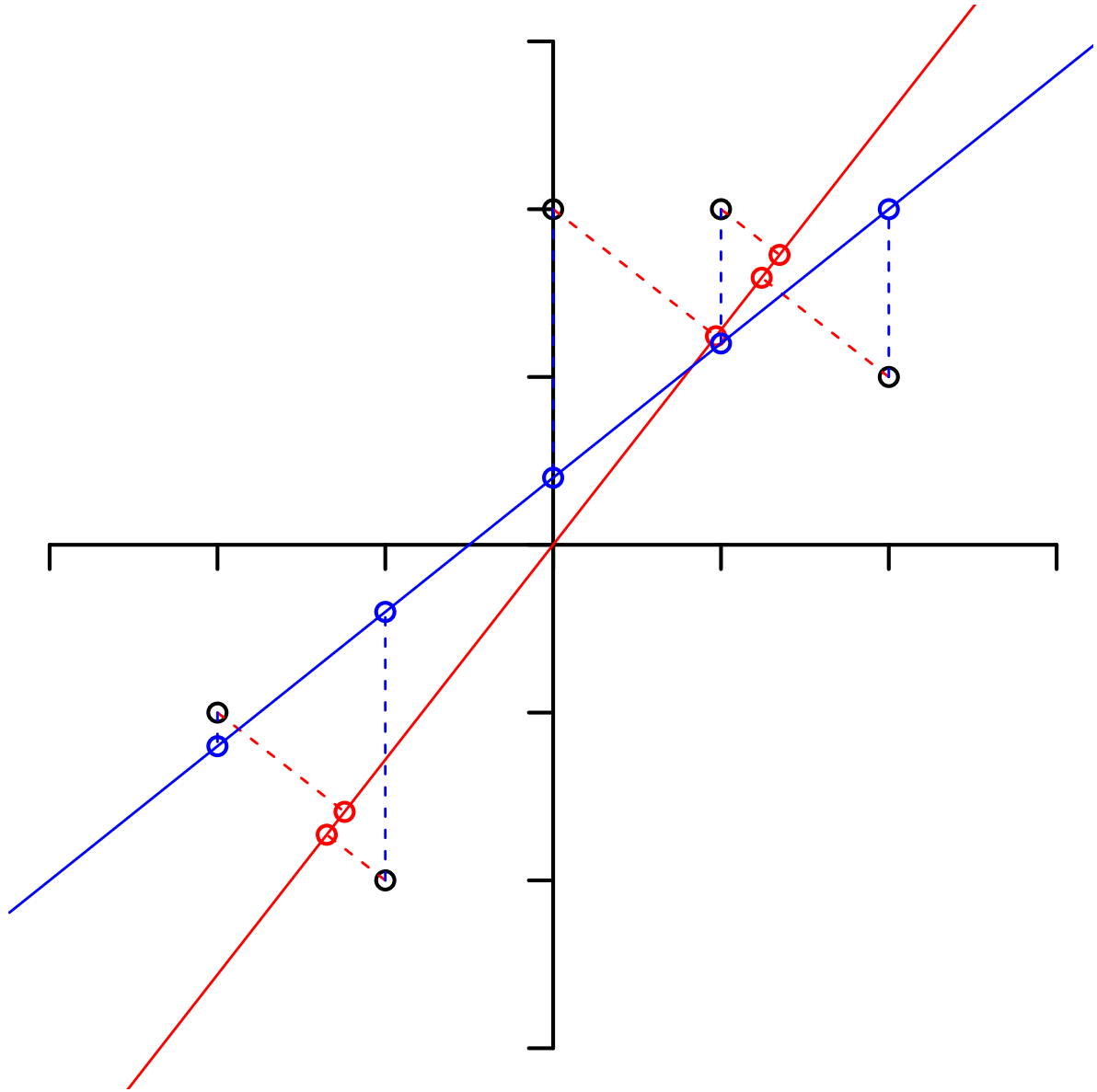


SVD and linear regression (2)

- Contrast this to the rank-1 truncated SVD of $\boldsymbol{A} = \begin{pmatrix} \boldsymbol{x} & \boldsymbol{y} \end{pmatrix}$
- We obtain a vector $\boldsymbol{u}_1$, a scaling coefficent σ_1, and a vector $\boldsymbol{v}_1$
- Let's look at the line described by $\boldsymbol{v}_1$ (roughly corresponds to β)
- Reconstructed data is $\boldsymbol{A}_1 = \boldsymbol{u}_1 \sigma_1 \boldsymbol{v}_1^T$; all points lie on the line
- There is no distinguished response variable;
  we minimize distance to line instead;
  i.e., $\|\boldsymbol{A} - \boldsymbol{A}_1\|_F$.
- This is different from regression



SVD and linear regression (3)



Component interpretation

- Recall that we can write
  $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T = \sum_{i=1}^{r} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T = \sum_{i=1}^{r} \boldsymbol{A}_i$
  - $\boldsymbol{A}_i = \sigma_i \boldsymbol{v}_i \boldsymbol{u}_i^T$
- This explains the data as a sums of (rank-1) layers
  - The first layer explains the most
  - The second corrects that by adding and removing smaller values
  - The third corrects that by adding and removing even smaller values
  - ...
- The layers don't have to be very intuitive

Outline

Outline

Problem

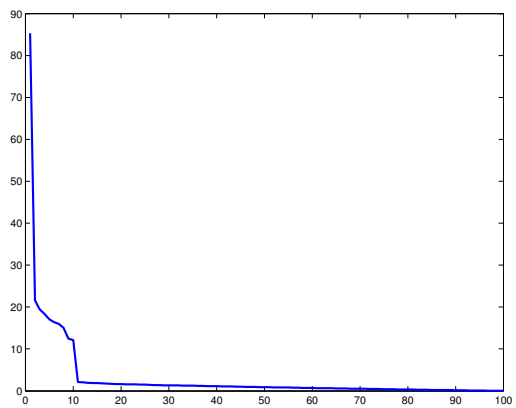
- Most data mining applications do not use full SVD, but truncated SVD
  - To concentrate on "the most important parts"
- But how to select the rank k of the truncated SVD?
  - What is important, what is unimportant?
  - What is structure, what is noise?
  - Too small rank: all subtlety is lost
  - Too big rank: all smoothing is lost
- Typical methods rely on singular values in a way or another

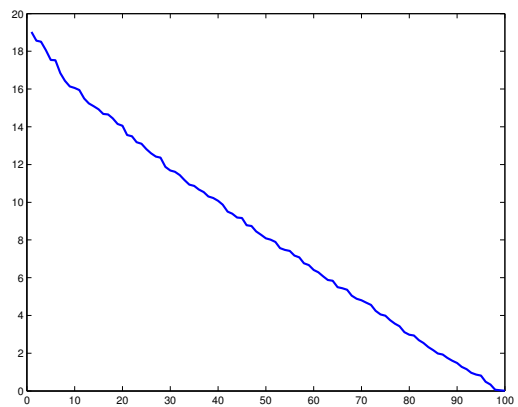
Guttman–Kaiser criterion and captured energy

- Perhaps the oldest method is the Guttman–Kaiser criterion:
  - Select k so that for all $i > k$, $\sigma_i < 1$
  - Motivation: all components with singular value less than unit are uninteresting
- Another common method is to select enough singular values such that the sum of their squares is 90% of the total sum of the squared singular values
  - The exact percentage can be different (80%, 95%)
  - Motivation: The resulting matrix "explains" 90% of the (sq.) Frobenius norm of the matrix
- **Problem:** Both of these methods are based on arbitrary thresholds and do not consider the "shape" of the data

Cattell's Scree test

- The **scree plot** plots the squared singular values in decreasing order
  - The plot looks like a side of the hill, hence the name
- The scree test is a subjective decision on the rank based on the shape of the scree plot
- The rank should be set to a point where
  - there is a clear drop in the magnitudes of the values values; or
  - the values start to even out
- **Problem:** Scree test is subjective, and many data don't have any clear shapes to use (or have many)
  - Automated methods have been developed to detect the shapes from the scree plot





Entropy-based method

- Consider the relative contribution of each singular value to the overall (sq.) Frobenius norm
  - Relative contribution of σ_k is $f_k = \sigma_k^2 / \sum_i \sigma_i^2$
- We can treat these as probabilities and define the (normalized) **entropy** of the singular values as

$$E = -\frac{1}{\log\big(\min\{n, m\}\big)} \sum_{i=1}^{\min\{n,m\}} f_i \log f_i$$

  - The basis of the logarithm doesn't matter
  - We assume that $0 \cdot \infty = 0$
  - Low entropy (close to 0): the first singular value has almost all mass
  - High entropy (close to 1): the singular values are almost equal
- The rank is selected to be the smallest k such that $\sum_{i=1}^{k} f_i \geq E$
- **Problem:** Why entropy?

Random flip of signs

- Multiply every element of the data $\boldsymbol{A}$ randomly with either 1 or -1 to get $\tilde{\boldsymbol{A}}$
  - The Frobenius norm doesn't change ($\|\boldsymbol{A}\|_F = \|\tilde{\boldsymbol{A}}\|_F$)
  - The spectral norm does change ($\|\boldsymbol{A}\|_2 \neq \|\tilde{\boldsymbol{A}}\|_2$)
    - How much this changes depends on how much "structure" $\boldsymbol{A}$ has
- We try to select k such that the residual matrix contains only noise
  - The residual matrix contains the last $m - k$ columns of $\boldsymbol{U}$, $\min\{n, m\} - k$ singular values, and last $n - k$ rows of $\boldsymbol{V}^T$
  - If $\boldsymbol{A}_{-k}$ is the residual matrix of $\boldsymbol{A}$ after rank-k truncated SVD, and $\tilde{\boldsymbol{A}}_{-k}$ is obtained from $\boldsymbol{A}_{-k}$ by randomly flipping signs, we select rank k to be such that $(\|\boldsymbol{A}_{-k}\|_2 - \|\tilde{\boldsymbol{A}}_{-k}\|_2)/\|\boldsymbol{A}_{-k}\|_F$ is small
- **Problem:** How small is small?

Outline

Normalization

- SVD is sensitive to data scaling
- Data should usually be normalized before SVD is applied
  - If one attribute is height in meters and another weights in grams, weight seems to carry much more importance in data about humans
  - If data is all positive, the first singular vector just explains where in the positive quadrant the data is
- The *z*-**scores** are attributes whose values are transformed by
  - Centering them to 0
    - Remove the mean of the attribute's values from each value
  - Normalizing the magnitudes
    - Divide every value with the standard deviation of the attribute
- Notice that the *z*-scores assume that
  - all attributes are equally important
  - attribute values are approximately normally distributed
- Values that have larger magnitude than importance can also be normalized by first taking logarithms (from positive values) or cubic roots
- The effects of normalization should always be considered

Relationship to PCA (1)

- Truncated SVD can also be used to battle the **curse of dimensionality**
  - All points are far from each other in very high dimensional spaces
  - High dimensionality slows down data mining algorithms
  - If we use the truncated SVD, every object is represented by its row in $\boldsymbol{U}_k$ (k values instead of n)
  - If $k \ll n$, we performed **dimensionality reduction**
- SVD is closely related to **principal components analysis** (PCA)
  - Key difference is "centering" of the data

1. Center each column of $\boldsymbol{A}$ to obtain $\boldsymbol{M}$
2. Compute the sample covariance matrix $\boldsymbol{S} = \boldsymbol{M}^T\boldsymbol{M}/(m-1)$
3. Compute the eigendecomposition $\boldsymbol{S} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^T$ s.t. $\boldsymbol{Q}$ orthogonal

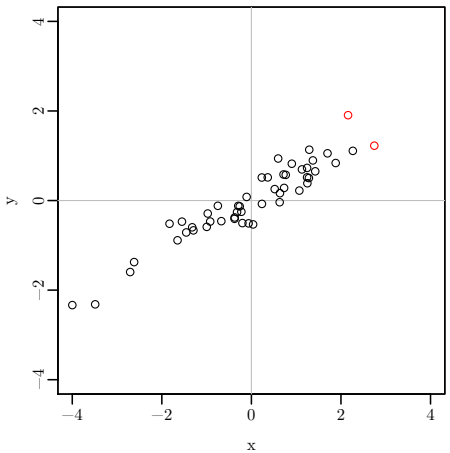
  - The columns of $\boldsymbol{Q}$ are called **principal components**
  - The corresponding eigenvalues in $\boldsymbol{\Lambda}$ are the component weights
  - Try it!

Relationship to PCA (2)

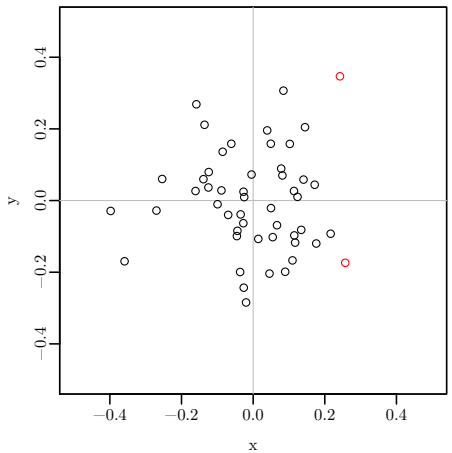
- Relationship between SVD and PCA
  - $\boldsymbol{Q}$ = eigenvectors of $\boldsymbol{M}^T\boldsymbol{M}/(m-1)$, $\boldsymbol{\Lambda}$ = corresponding eigenvalues
  - Let $\boldsymbol{M} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ be the SVD of $\boldsymbol{M}$
  - The SVD of $\boldsymbol{M}/\sqrt{m-1}$ is then $\boldsymbol{U}(\boldsymbol{\Sigma}/\sqrt{m-1})\boldsymbol{V}^T$
  - From slide 12, we know that $\boldsymbol{Q} = \boldsymbol{V}$
  - From slide 12, we know that $\boldsymbol{\Sigma}^2/(m-1) = \boldsymbol{\Lambda}$
- PCA associates each data object with a set of **scores**
  - One per principal component
  - $m \times n$ "score matrix" given by $\boldsymbol{M}\boldsymbol{Q}$
  - We have: $\boldsymbol{M}\boldsymbol{Q} = (\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T)\boldsymbol{V} = \boldsymbol{U}\boldsymbol{\Sigma}$
  - Usually we only take first k-components: $\boldsymbol{M}\boldsymbol{Q}_k = \boldsymbol{U}_k\boldsymbol{\Sigma}_k$
  - Also known as the **Karhunen–Loève transform** (KLT) of the rows of $\boldsymbol{M}$

Relationship to PCA (3)

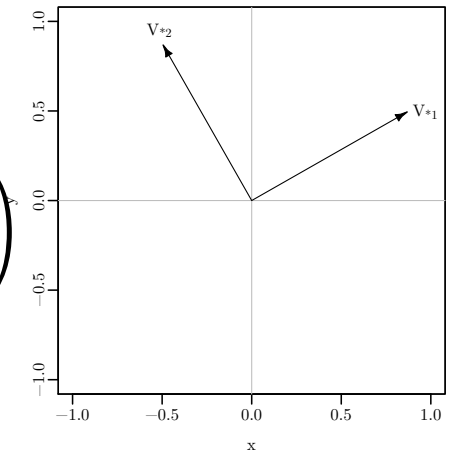
$$\boldsymbol{M}_{50\times 2} = \boldsymbol{U}_2 \boldsymbol{\Sigma}_2 \boldsymbol{V}_2^T = \boldsymbol{Q}_2^T$$



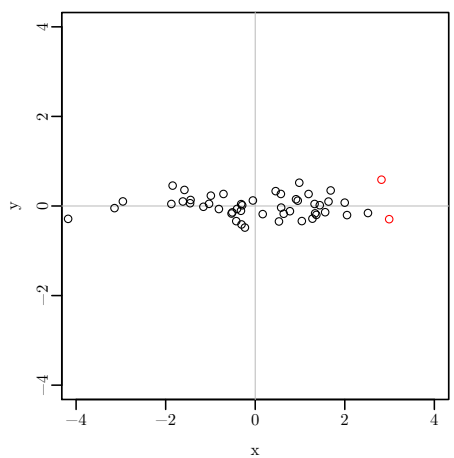
=



$$\begin{pmatrix} 11.64 & 0 \\ 0 & 1.69 \end{pmatrix}$$



$$\boldsymbol{M}\boldsymbol{Q}_2 = \boldsymbol{U}_2\boldsymbol{\Sigma}_2$$



What is the difference between the PCA and the SVD of $\boldsymbol{A}$ ($\neq \boldsymbol{M}$)?

Relationship to PCA (4)

- Assumptions of PCA
  - *Linearity*: The data set is a linear combination of the (retained) components.
  - *Large variances indicate important structure*: PCA scores may or may not be suitable for subsequent tasks (e.g., classification).
  - *Components are orthogonal*
- Recall that $[\boldsymbol{M}^T\boldsymbol{M}/(m-1)]_{ij}$ holds the sample covariance between attributes i and j (i.e., columns $\boldsymbol{A}_{*i}$ and $\boldsymbol{A}_{*j}$)
  - When each row of $\boldsymbol{M}$ is an i.i.d. observation of some underlying distribution, then $\boldsymbol{M}^T\boldsymbol{M}/(m-1)$ is an *unbiased estimator* of the covariance matrix of that distribution

Relationship to PCA (5)

- For our example data, $\boldsymbol{M}$ is a 50×2 matrix
- Here is its sample covariance matrix

$$\boldsymbol{M}^T\boldsymbol{M}/(m-1) = \begin{array}{|c|c|}\hline 2.1 & 1.16 \\ \hline 1.16 & 0.72 \\ \hline \end{array}$$

- And here the sample covariance matrix of the PCA scores

$$(\boldsymbol{MQ})^T(\boldsymbol{MQ})/(m-1) = \begin{array}{|c|c|}\hline 2.76 & 0 \\ \hline 0 & 0.06 \\ \hline \end{array}$$

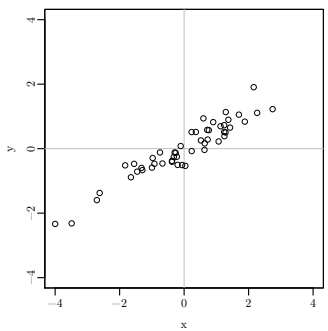
- This is "by construction"

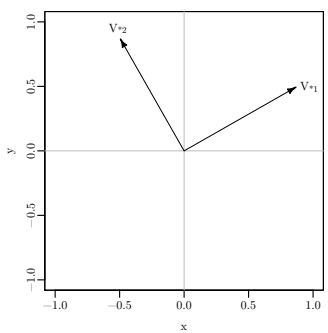
Relationship to PCA (6)

- Only information used by PCA is sample means and covariances
  - Data assumed i.i.d. from a multivariate normal distribution
  - Mean and covariance parameters are estimated from data (and ordered)
  - We obtain distribution $\mathcal{N}(\mathbf{0}, \mathbf{\Lambda})$ after rotation (with correct estimates and assuming normality assumption holds)
- We already established that $f_k = \sigma_k^2 / \sum_i \sigma_i^2$ can be seen as the relative contribution of a singular value to the sq. Frobenius norm
- We also know that $\sigma_k^2/(m-1) = \lambda_k$
- Implies that f_k can also be seen as the fraction of the **total variance explained** by the k-th principal component
  - Interpretation only valid when data is centered (!)
  - In a Scree plot, one may plot f_k (or their cumulative sum) instead of the σ_k^2
- To make sure PCA is adequate, ensure that data "looks" normally distributed (or transform it to look normal)

Removing noise

- Very common application of SVD is to remove the noise from the data
- This works simply by taking the truncated SVD from the (normalized) data
  - The big problem is to select the rank of the truncated SVD
- Example:





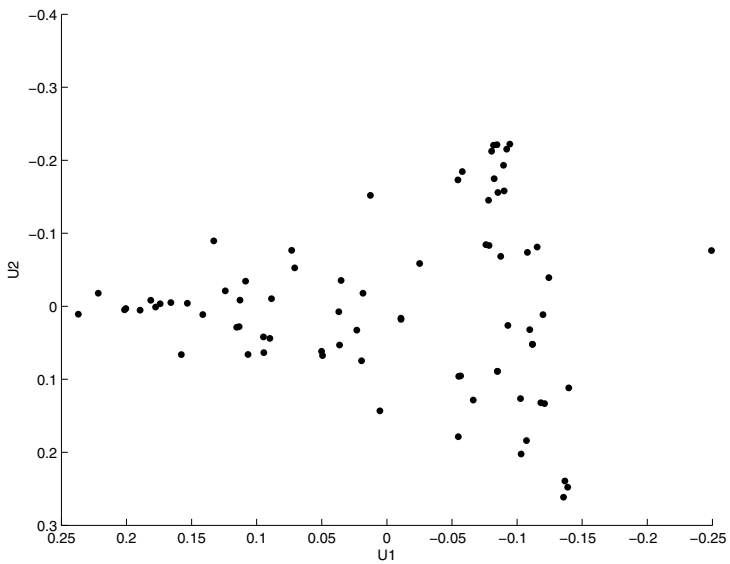
$\sigma_1 = 11.64$

$\sigma_2 = 1.69$

- Original data
  - Looks like 1-dimensional with some noise
- The right singular vectors show the directions
  - The first looks like the data direction
  - The second looks like the noise direction
- The singular values confirm this

Visualization

- Truncated SVD with $k = 2, 3$ allows us to visualize the data
  - We can plot the projected data points after 2D or 3D PCA
  - Or we can plot the scatter plot of two or three singular vectors
  - Or we color data points based on their entries in a singular vector
  - ...



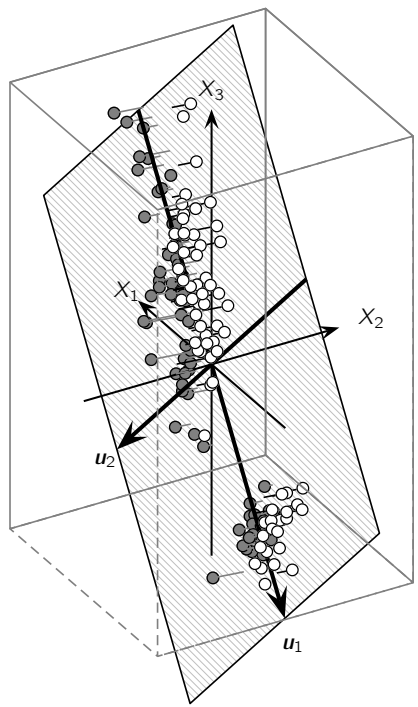


Figure 3.2. *The first two factors for a dataset ranking wines.*

Latent semantic analysis

- The **latent semantic analysis** (LSA) is an information retrieval method that uses SVD
- The data: a term-document matrix $\boldsymbol{A}$
  - Values are (weighted) term frequencies
  - Typically tf-idf values (the frequency of the term in the document divided by the global frequency of the term)
- The truncated SVD $\boldsymbol{A}_k = \boldsymbol{U}_k \boldsymbol{\Sigma}_k \boldsymbol{V}_k^T$ of $\boldsymbol{A}$ is computed
  - Matrix $\boldsymbol{U}_k$ associates documents to "topics"
  - Matrix $\boldsymbol{V}_k$ associates "topics" to terms
  - If two rows of $\boldsymbol{U}_k$ are similar, the corresponding documents "talk about same things"
- A query q can be answered by considering its term vector $\boldsymbol{q}$
  - $\boldsymbol{q}$ is projected to $\boldsymbol{q}_k = \boldsymbol{q}\boldsymbol{V}\boldsymbol{\Sigma}^{-1}$ (called: **fold in**)
  - $\boldsymbol{q}_k$ is compared to rows of $\boldsymbol{U}$ and most similar rows are returned
- Later more (when we consider NMF and LDA)

Landauer & Dumais, 1997

Outline

Algorithms for SVD

- In principle, the SVD of $\boldsymbol{A}$ can be computed by computing the eigendecomposition of $\boldsymbol{A}\boldsymbol{A}^T$
  - This gives us left singular vectors and squares of singular values
  - Right singular vectors can be solved: $\boldsymbol{V}^T = \boldsymbol{\Sigma}^{-1}\boldsymbol{U}^T\boldsymbol{A}$
  - **Bad for numerical stability!**
- Full SVD can be computed in time $O\left(nm\min\{n, m\}\right)$
  - Matrix $\boldsymbol{A}$ is first reduced to a bidiagonal matrix
  - The SVD of the bidiagonal matrix is computed using iterative methods (similar to eigendecompositions)
- Methods that are fast in practice exist
  - Especially for truncated SVD
- Efficient implementation of an SVD algorithm requires considerable work and knowledge
  - Luckily (almost) all numerical computation packages and programs implement SVD

Outline

Lessons learned

- SVD is the Swiss Army knife of (numerical) linear algebra
  $\rightarrow$ Ranks, kernels, norms, inverses, ...
- SVD is also very useful in data analysis
  $\rightarrow$ Dimensionality reduction, noise removal, visualization, ...
- Truncated SVD is best low-rank factorization of the data in terms of Frobenius norm
- Selecting the correct rank for truncated SVD is still a problem
- Interpretation of results can be challenging

Suggested reading

- Skillicorn, Ch. 3
- Meyer, Ch. 5.12
- Gene H. Golub & Charles F. Van Loan: *Matrix Computations*, 3rd ed. Johns Hopkins University Press, 1996
  - Excellent source for the algorithms and theory, but very dense

Data Mining and Matrices

04 – Matrix Completion

Prof. Dr. Rainer Gemulla

Universität Mannheim

FSS 2018

Recap: Singular Value Decomposition

- SVD is useful in data analysis
  $\rightarrow$ Noise removal, visualization, dimensionality reduction, ...
- Provides a means to understand the hidden structure in the data

We may think of $\boldsymbol{A}_k$ and its factor matrices as a **low-rank model** of the data:

- Used to capture the important aspects of the data
  (cf. principal components)
- Ignores the rest
- Truncated SVD is best low-rank factorization of the data in terms of Frobenius norm
- Truncated SVD $\boldsymbol{A}_k = \boldsymbol{U}_k \boldsymbol{\Sigma}_k \boldsymbol{V}_k^T$ of $\boldsymbol{A}$ thus satisfies

$$\|\boldsymbol{A} - \boldsymbol{A}_k\|_F = \min_{\text{rank}(\boldsymbol{B})=k} \|\boldsymbol{A} - \boldsymbol{B}\|_F$$

Incomplete data

- If all entries of the data matrix are available, computing a low-rank model is easy
- We now look at the case of partially observed data
  - Many data mining methods cannot handle missing data well
  - Need for imputation
- Our goal is to build a low-rank model...
  - ...and ultimately approximately recover the full matrix
  - Cannot be done in general; assumptions needed
  - Ongoing research topic, large body of results
- Some reasons for missing entries
  - Failure in data acquisition processes
  - Expensive to obtain all entries
  - Some entries cannot be measured

	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69	112.5
2	Alice		13	56.5	84
3	Barbara	F	13	65.3	98
4	Carol	F	14		102.5
5	Henry	M	14	63.5	102.5
6	James	M	12	57.3	83
7	Jane	F	12		84.5
8	Janet	F	15	62.5	112.5
9	Jeffrey	M	13	62.5	
10	John	M	12	59	99.5
11	Joyce	F	11	51.3	50.5
12	Judy	F	14	64.3	90
13	Louise	F	12	56.3	77
14		F	15	66.5	112
15	Philip	M	16	72	150
16	Robert	M	12	64.8	128

Outline

Outline

Recommender systems

- Problem
  - Set of users
  - Set of items (movies, books, jokes, products, stories, ...)
  - Feedback (ratings, purchase, click-through, tags, ...)
  - Sometimes: metadata (user profiles, item properties, ...)
- Goal: Predict preferences of users for items
- Ultimate goal: Create item recommendations for each user
- Example

$$\begin{array}{c} \\ \textit{Alice} \\ \textit{Bob} \\ \textit{Charlie} \end{array} \begin{array}{c} \begin{array}{ccc} \textit{Avatar} & \textit{The Matrix} & \textit{Up} \end{array} \\ \begin{pmatrix} ? & 4 & 2 \\ 3 & 2 & ? \\ 5 & ? & 3 \end{pmatrix} \end{array}$$

Collaborative filtering

- Key idea: Make use of past user behavior
- No domain knowledge required
- No expensive data collection needed
- Allows discovery of complex and unexpected patterns
- Widely adopted: Amazon, TiVo, Netflix, Microsoft
- Key techniques: neighborhood models, latent factor models

	Avatar	*The Matrix*	*Up*
Alice	?	4	2
Bob	3	2	?
Charlie	5	?	3

Leverage past behavior of other users and/or on other items.

A simple baseline

- m users, n items, $m \times n$ rating matrix $\boldsymbol{D}$
- Revealed entries $\Omega = \{\,(i,j) \mid \text{rating } d_{ij} \text{ is revealed}\,\}$, $N = |\Omega|$
- **Baseline predictor**: $b_{ij} = \mu + b_i + b_j$
  - $\mu = \frac{1}{N}\sum_{(i,j)\in\Omega} d_{ij}$ is the overall average rating
  - b_i is a *user bias* (user's tendency to rate low/high)
  - b_j is an *item bias* (item's tendency to be rated low/high)
- Least squares estimates: $\operatorname{argmin}_{b_*} \sum_{(i,j)\in\Omega}(d_{ij} - \mu - b_i - b_j)^2$

$\boldsymbol{D}$	*Avatar* (1.01)	*Matrix* (0.34)	*Up* (−1.32)
Alice (0.32)	? (4.5)	**4** (3.8)	**2** (2.1)
Bob (−1.34)	**3** (2.8)	**2** (2.2)	? (0.5)
Charlie (0.99)	**5** (5.2)	? (4.5)	**3** (2.8)

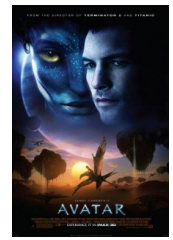
$$m = 3$$
$$n = 3$$
$$\Omega = \{\,(1,2),(1,3),(2,1),\dots\}$$
$$N = 6$$
$$\mu = 3.17$$
$$b_{12} = 3.17 + 0.32 + 0.34 = 3.8$$
$$b_{32} = 3.17 + 0.99 + 0.34 = 4.5$$

Baseline does not account for personal tastes.

When does a user like an item?

- **Neighborhood models** (kNN):
  When the user likes similar items
  - Find the top-k most similar items the user has rated
  - Combine the ratings of these items (e.g., average)
  - Requires a similarity measure (e.g., Pearson correlation coefficient)

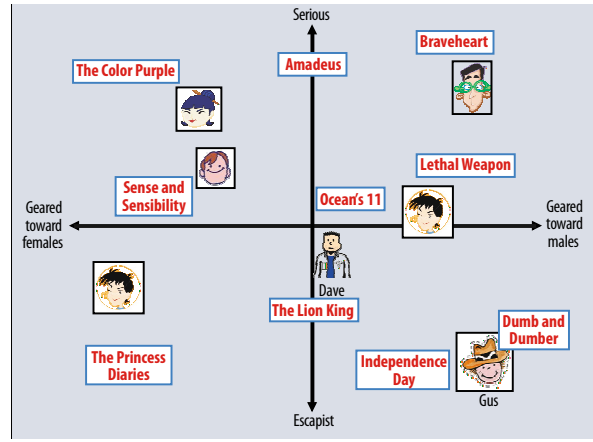
is similar to



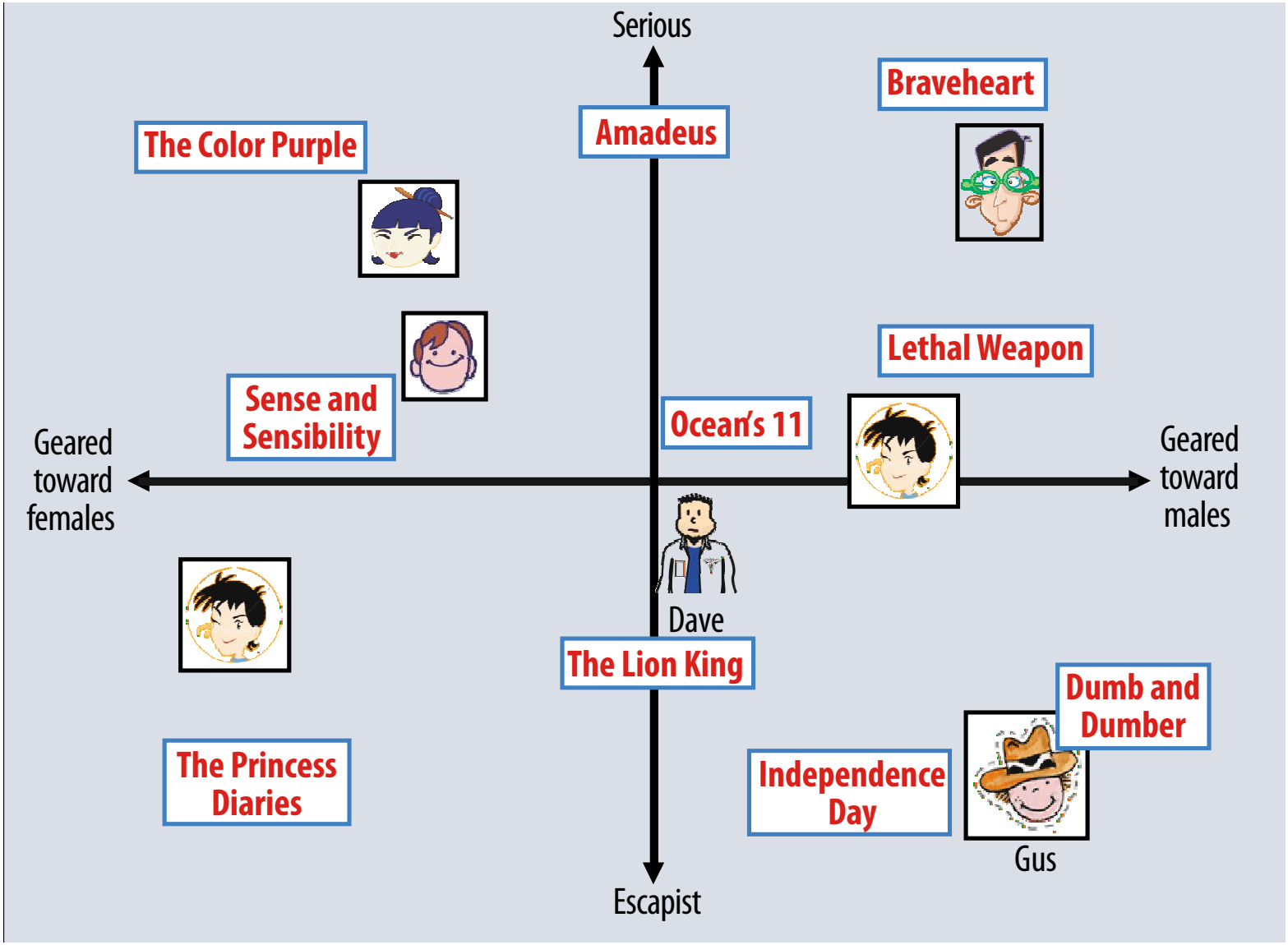
Unrated by Bob
$\rightarrow$ predict 4

Bob rated 4

- **Latent factor models** (LFM):
  When *similar* users like similar items
  - More holistic approach
  - Users and items are placed in the same "latent factor space"
  - Position of a user and an item related to preference (via dot products)



Intuition behind latent factor models (1)



Koren et al., 2009.

Intuition behind latent factor models (2)

- Does user $\boldsymbol{u}$ like item $\boldsymbol{v}$?
- Quality: measured via **direction** from origin ($\cos \angle(\boldsymbol{u}, \boldsymbol{v})$)
  - Same direction $\rightarrow$ attraction: $\cos \angle(\boldsymbol{u}, \boldsymbol{v}) \approx 1$
  - Opposite direction $\rightarrow$ repulsion: $\cos \angle(\boldsymbol{u}, \boldsymbol{v}) \approx -1$
  - Orthogonal direction $\rightarrow$ oblivious: $\cos \angle(\boldsymbol{u}, \boldsymbol{v}) \approx 0$
- Strength: measured via **distance** from origin ($\|\boldsymbol{u}\| \, \|\boldsymbol{v}\|$)
  - Far from origin $\rightarrow$ strong relationship: $\|\boldsymbol{u}\| \, \|\boldsymbol{v}\|$ large
  - Close to origin $\rightarrow$ weak relationship: $\|\boldsymbol{u}\| \, \|\boldsymbol{v}\|$ small
- Overall preference: measured via **dot product** ($\boldsymbol{u} \cdot \boldsymbol{v}$)

$$\boldsymbol{u} \cdot \boldsymbol{v} = \|\boldsymbol{u}\| \, \|\boldsymbol{v}\| \frac{\boldsymbol{u} \cdot \boldsymbol{v}}{\|\boldsymbol{u}\| \, \|\boldsymbol{v}\|} = \|\boldsymbol{u}\| \, \|\boldsymbol{v}\| \cos \angle(\boldsymbol{u}, \boldsymbol{v})$$

  - Same direction, far out $\rightarrow$ strong attraction: $\boldsymbol{u} \cdot \boldsymbol{v}$ large positive
  - Opposite direction, far out $\rightarrow$ strong repulsion: $\boldsymbol{u} \cdot \boldsymbol{v}$ large negative
  - Orthogonal direction, any distance $\rightarrow$ oblivious: : $\boldsymbol{u} \cdot \boldsymbol{v} \approx 0$

But how to select dimensions and where to place items and users?
Key idea: Pick dimensions that explain the known data well.

Latent factor models (simple form)

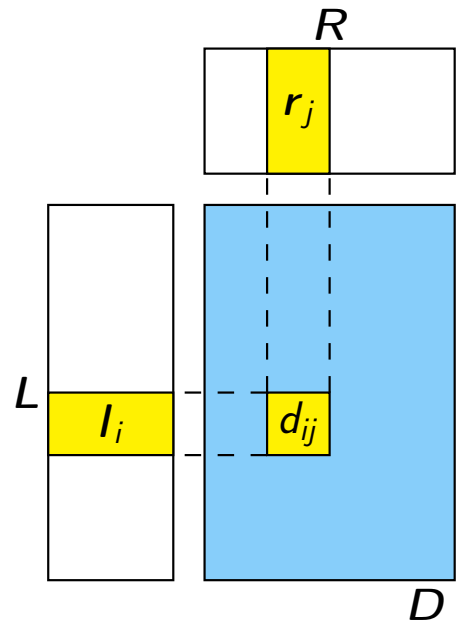
- Given rank r, find $m \times r$ matrix $\boldsymbol{L}$ and $r \times n$ matrix $\boldsymbol{R}$ such that

$$d_{ij} \approx [\boldsymbol{LR}]_{ij} \qquad \text{for } (i,j) \in \Omega$$

- Least squares formulation

$$\min_{\boldsymbol{L},\boldsymbol{R}} \sum_{(i,j)\in\Omega} (d_{ij} - [\boldsymbol{LR}]_{ij})^2$$

- Example ($r = 1$)



		$\boldsymbol{R}$		
		Avatar	*The Matrix*	*Up*
		(2.24)	(1.92)	(1.18)
$\boldsymbol{L}$	*Alice*	?	**4**	**2**
	(1.98)	(4.4)	(3.8)	(2.3)
	Bob	**3**	**2**	?
	(1.21)	(2.7)	(2.3)	(1.4)
	Charlie	**5**	?	**3**
	(2.30)	(5.2)	(4.4)	(2.7)

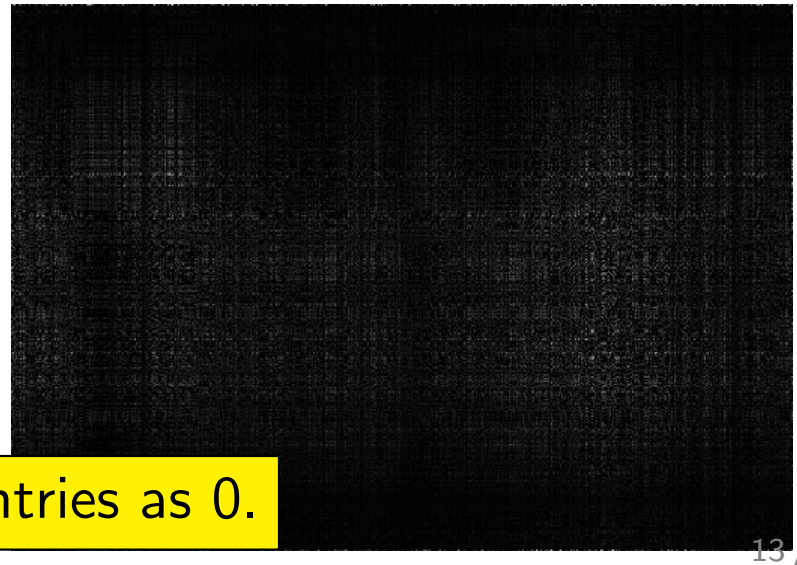
SVD and missing values

Input data

Rank-10 truncated SVD

10% of input data

Rank-10 truncated SVD



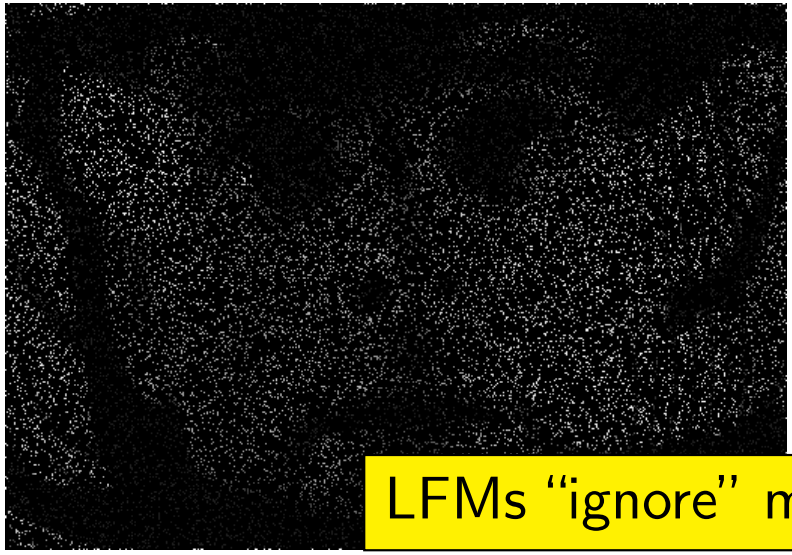
SVD treats missing entries as 0.

Latent factor models and missing values

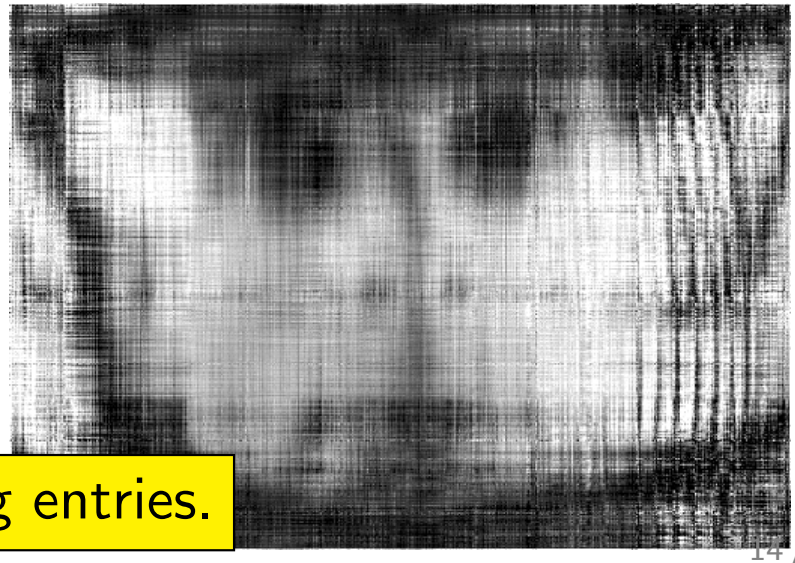
Input data

Rank-10 LFM

10% of input data



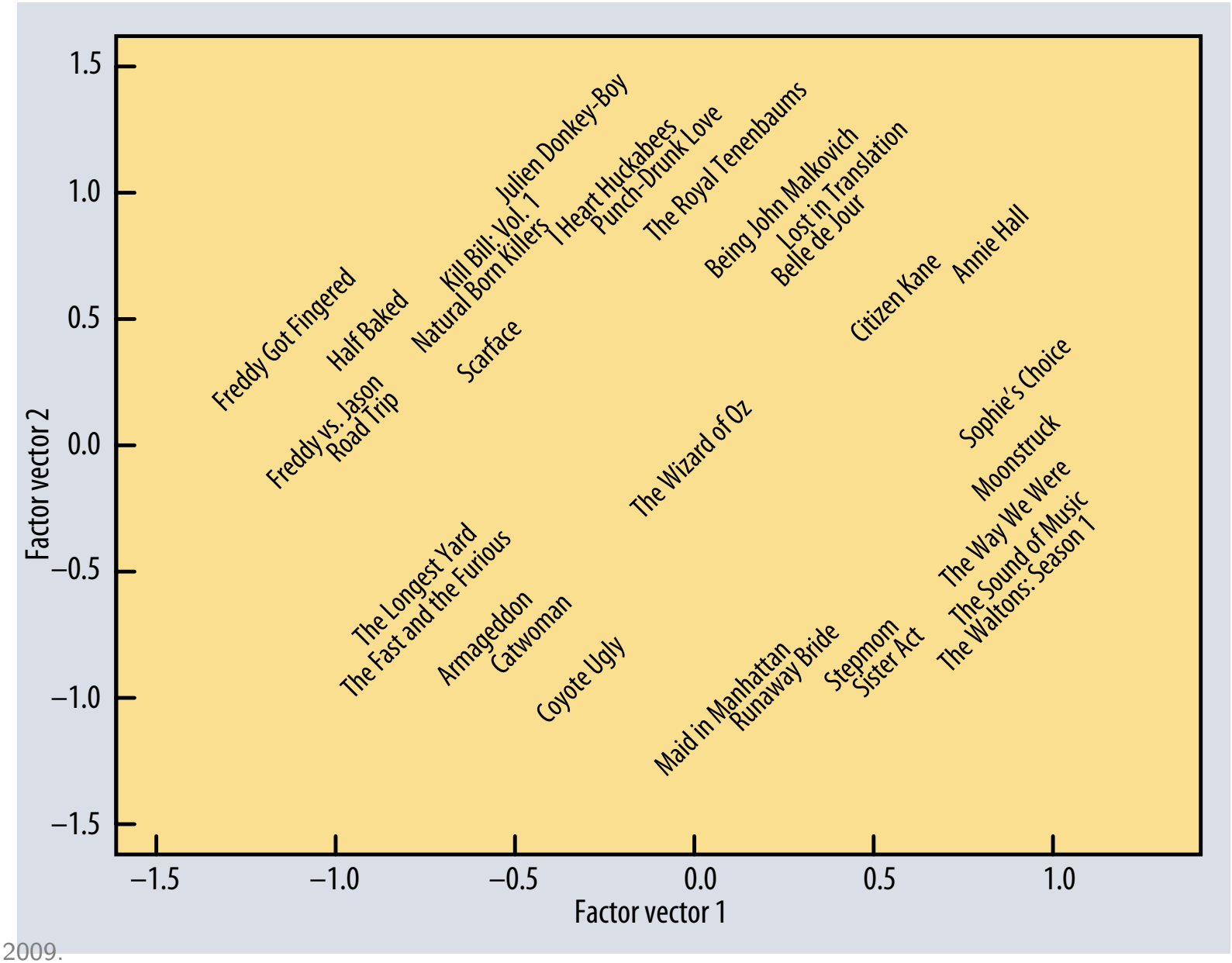
Rank-10 LFM



LFMs “ignore” missing entries.

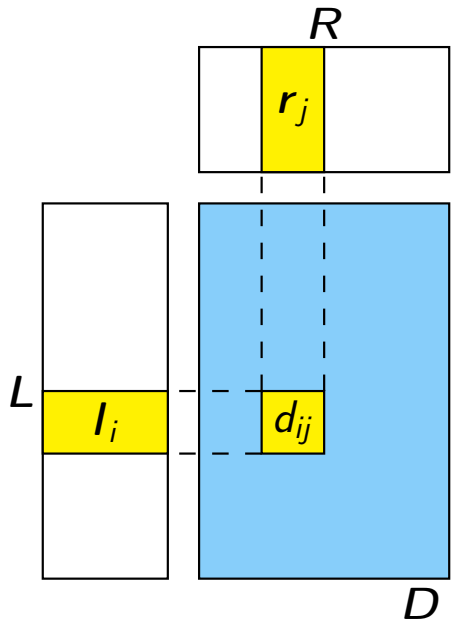
Example: Netflix prize data

(≈ 500k users, ≈ 17k movies, ≈ 100M ratings)



Koren et al., 2009.

Latent factor models (summation form)

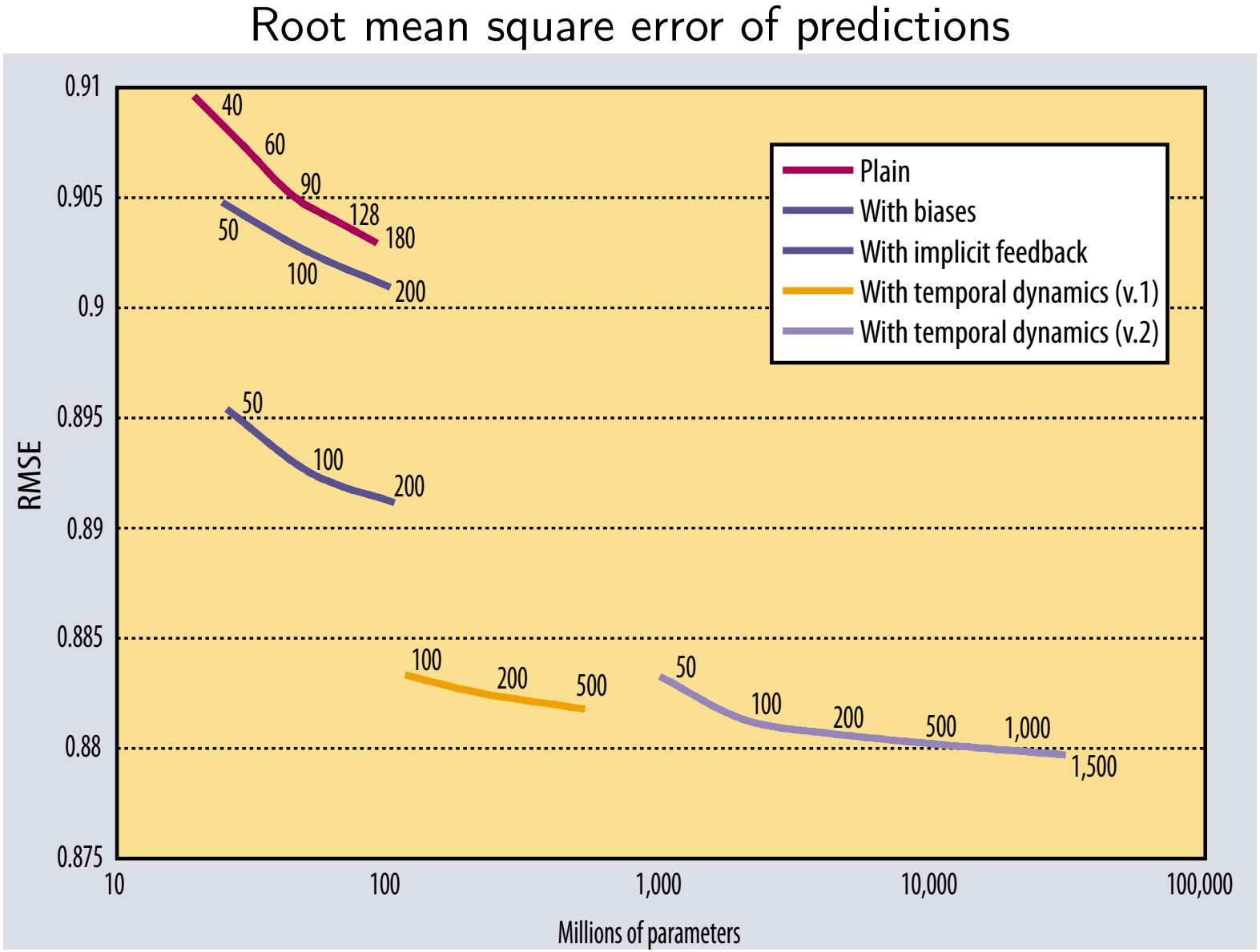


- Least squares formulation prone to overfitting
- More general **summation form**:

$$L = \sum_{(i,j)\in\Omega} L_{ij}(\boldsymbol{l}_i, \boldsymbol{r}_j) + R(\boldsymbol{L}, \boldsymbol{R}),$$

  - L is **global loss**
  - $\boldsymbol{l}_i$ and $\boldsymbol{r}_j$ are user and item **parameters**, resp.
  - L_{ij} is **local loss**, e.g., $L_{ij} = (d_{ij} - [\boldsymbol{L}\boldsymbol{R}]_{ij})^2$
  - R is **regularization term**, e.g., $R = \lambda(\|\boldsymbol{L}\|_F^2 + \|\boldsymbol{R}\|_F^2)$
- As before, goal is to solve $\min_{\boldsymbol{L},\boldsymbol{R}} L(\boldsymbol{L}, \boldsymbol{R})$
- Loss function can be more sophisticated
  - Improved predictors (e.g., include user and item bias)
  - Additional feedback data (e.g., time, implicit feedback)
  - Regularization terms (e.g., weighted depending on amount of feedback)
  - Available metadata (e.g., demographics, genre of a movie)

Example: Netflix prize data



Koren et al., 2009.

Outline

The matrix completion problem

Complete these matrices!

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & ? & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} \qquad \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & ? & ? & ? & ? \\ 1 & ? & ? & ? & ? \\ 1 & ? & ? & ? & ? \\ 1 & ? & ? & ? & ? \end{pmatrix}$$

Matrix completion is impossible without additional assumptions!

Let's assume that underlying full matrix is "simple" (here: rank 1).

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

When/how can we recover a low-rank matrix from a sample of its entries?

Rank minimization

Definition (rank minimization problem)

Given an $n \times n$ data matrix $\boldsymbol{D}$ and an index set Ω of revealed entries. The *rank minimization problem* is

$$\begin{array}{ll} \text{minimize} & \operatorname{rank}(\boldsymbol{X}) \\ \text{subject to} & d_{ij} = x_{ij} \qquad (i,j) \in \Omega \\ & \boldsymbol{X} \in \mathbb{R}^{n \times n}. \end{array}$$

- Seeks for "simplest explanation" fitting the data
- If unique and sufficient samples, recovers $\boldsymbol{D}$ (i.e., $\boldsymbol{X} = \boldsymbol{D}$)
- NP-hard

Time complexity of existing rank minimization algorithms double exponential in n (and also slow in practice).

Nuclear norm minimization

- Denote by $\boldsymbol{\sigma} = \begin{pmatrix} \sigma_1 & \sigma_2 & \dots & \sigma_n \end{pmatrix}^T$ the singular values of $\boldsymbol{X}$
- $\text{rank}(\boldsymbol{X}) = |\{\, \sigma_k > 0 \,\}| = \sum_{k=1}^{n} I_{\sigma_k > 0} = \|\boldsymbol{\sigma}\|_0$
- **Nuclear norm**: $\|\boldsymbol{X}\|_* = \sum_{k=1}^{n} \sigma_k = \|\boldsymbol{\sigma}\|_1$
- Nuclear norm can be seen as approximation to rank

Definition (nuclear norm minimization)

Given an $n \times n$ data matrix $\boldsymbol{D}$ and an index set Ω of revealed entries. The *nuclear minimization problem* is

$$\begin{array}{ll} \text{minimize} & \|\boldsymbol{X}\|_* \\ \text{subject to} & d_{ij} = x_{ij} \qquad (i,j) \in \Omega \\ & \boldsymbol{X} \in \mathbb{R}^{n \times n}. \end{array}$$

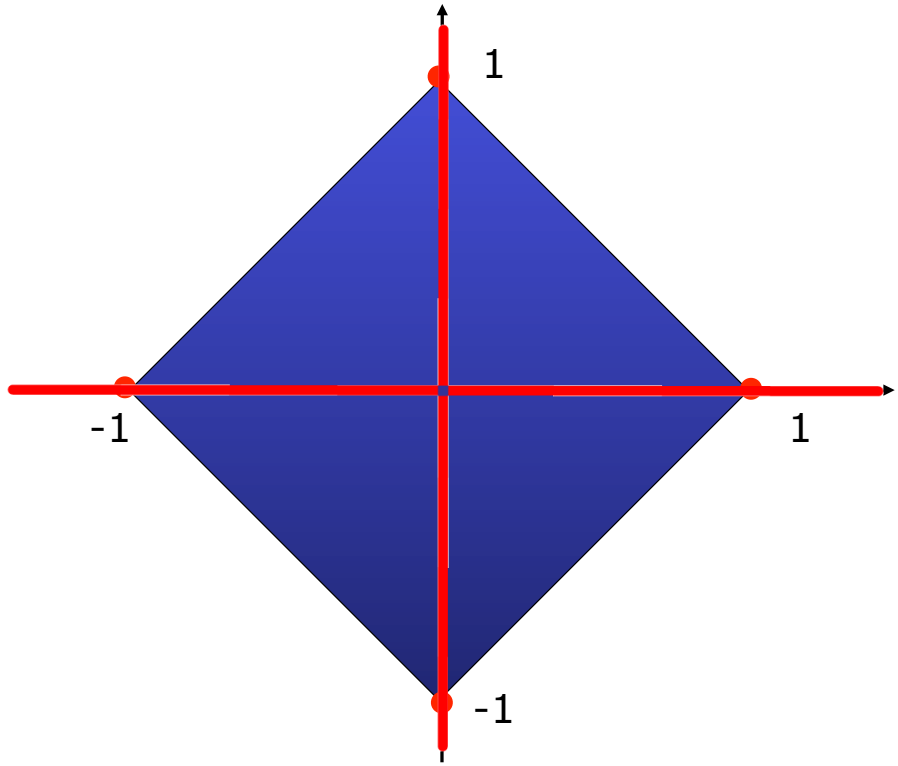
- A convex relaxation of rank minimization
- Nuclear norm is convex function (thus local optimum is global optimum)

Can be optimized (more) efficiently via semidefinite programming.

Why nuclear norm minimization? (1)

Let's look at two-dimensional vectors first.

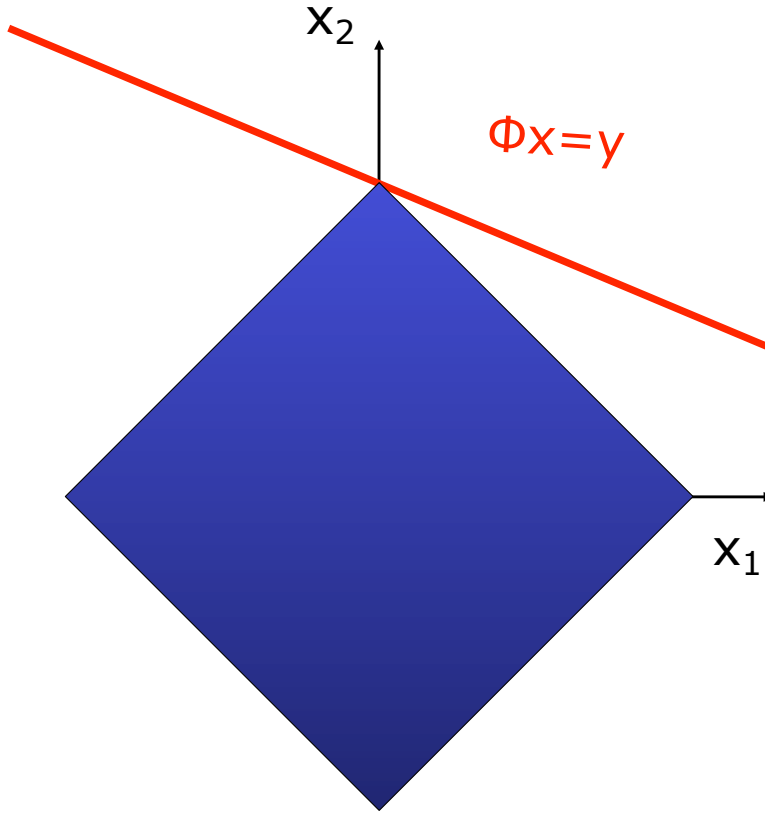
- Border of blue area = L_1 unit ball
- Red lines = L_0 "unit ball" $\rightarrow$ axes except $\mathbf{0}$
- L_0 unit ball intersects L_1 unit ball at extreme points



Why nuclear norm minimization? (2)

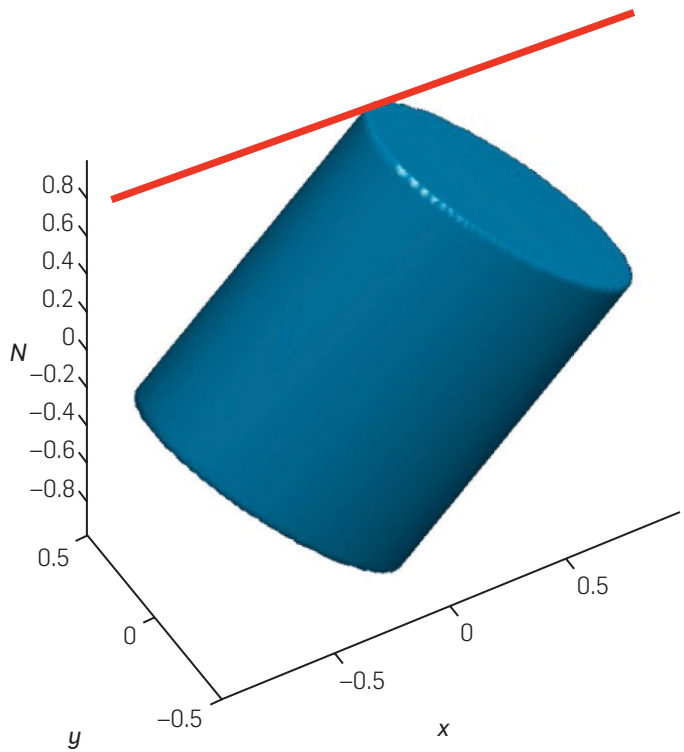
Let's find a solution to the problem $\Phi \boldsymbol{x} = y$.

- Underdetermined system with infinitely many solution
- We usually pick one that has certain structure
- E.g. sparsity: $\min \|\boldsymbol{x}\|_0$ s.t. $\Phi \boldsymbol{x} = y$
  - Example solution: $x_1 = y/\phi_1$, $x_2 = 0 \rightarrow L_0 = 1$
- Approximation: minimize L_1
  - Recall: $\|\boldsymbol{x}\|_1 = \sum |x_i|$
  - Increasing L_1 norm can be seen as "inflating" the L_1 unit ball
  - Minimum L_1 norm = minimum inflation
  - Achieved at intersection with x_1 or x_2 axis (whatever is smaller)



Why nuclear norm minimization (3)

Figure 1. Unit ball of the nuclear norm for symmetric 2 × 2 matrices. The red line depicts a random one-dimensional affine space. Such a subspace will generically intersect a sufficiently large nuclear norm ball at a rank one matrix.



$$\begin{pmatrix} x & y \\ y & z \end{pmatrix}$$

- Consider SVD of $\boldsymbol{D} = \boldsymbol{U}\Sigma\boldsymbol{V}^T$
- Unit nuclear norm ball = convex combination (σ_k) of rank-1 matrices of unit Frobenius norm ($\boldsymbol{U}_{*k}\boldsymbol{V}_{*k}^T$)
- Extreme points have low rank (in figure: rank-1 matrices of unit Frobenius norm)
- Nuclear norm minimization: inflate unit ball as little as possible to reach $d_{ij} = x_{ij}$
- Solution lies at extreme point of inflated ball $\rightarrow$ (hopefully) low rank

Candès and Recht, 2012

Relationship to LFMs

- Recall regularized LFM ($\boldsymbol{L}$ is $m \times r$, $\boldsymbol{R}$ is $r \times n$):

$$\min_{\boldsymbol{L},\boldsymbol{R}} \sum_{(i,j)\in\Omega} (d_{ij} - [\boldsymbol{L}\boldsymbol{R}]_{ij})^2 + \lambda(\|\boldsymbol{L}\|_F^2 + \|\boldsymbol{R}\|_F^2)$$

- View as matrix completion problem by enforcing $d_{ij} = [\boldsymbol{L}\boldsymbol{R}]_{ij}$:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\left(\|\boldsymbol{L}\|_F^2 + \|\boldsymbol{R}\|_F^2\right) \\ \text{subject to} & d_{ij} = x_{ij} \qquad (i,j) \in \Omega \\ & \boldsymbol{L}\boldsymbol{R} = \boldsymbol{X}. \end{array}$$

- One can show: for r chosen larger than rank of nuclear norm optimum, equivalent to nuclear norm minimization
- For some intuition, suppose $\boldsymbol{X} = \boldsymbol{L}\boldsymbol{R} = \boldsymbol{U}\Sigma\boldsymbol{V}^T$ at optimum $\boldsymbol{L}$ and $\boldsymbol{R}$: $\frac{1}{2}\left(\|\boldsymbol{L}\|_F^2 + \|\boldsymbol{R}\|_F^2\right) \leq \frac{1}{2}\left(\left\|\boldsymbol{U}\Sigma^{1/2}\right\|_F^2 + \left\|\Sigma^{1/2}\boldsymbol{V}^T\right\|_F^2\right)$
$= \frac{1}{2}\sum_{i=1}^{n}\sum_{k=1}^{r}(u_{ik}^2\sigma_k + v_{ik}^2\sigma_k)$
$= \sum_{k=1}^{r}\sigma_k = \|\boldsymbol{X}\|_*$

Candès and Recht, 2012

When can we hope to recover $\boldsymbol{D}$? (1)

Assume $\boldsymbol{D}$ is the 5×5 all-ones matrix (rank 1).

$$\begin{pmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \mathbf{1} & ? & ? & ? & ? \\ \mathbf{1} & ? & ? & ? & ? \\ \mathbf{1} & ? & ? & ? & ? \\ \mathbf{1} & ? & ? & ? & ? \end{pmatrix}$$

Ok

$$\begin{pmatrix} \mathbf{1} & ? & ? & \mathbf{1} & ? \\ ? & ? & \mathbf{1} & ? & ? \\ ? & \mathbf{1} & ? & ? & \mathbf{1} \\ \mathbf{1} & ? & \mathbf{1} & ? & ? \\ ? & \mathbf{1} & \mathbf{1} & ? & ? \end{pmatrix}$$

Ok

$$\begin{pmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & ? \\ \mathbf{1} & \mathbf{1} & ? & ? & ? \\ \mathbf{1} & ? & ? & ? & ? \\ \mathbf{1} & ? & ? & \mathbf{1} & ? \\ \mathbf{1} & ? & ? & ? & ? \end{pmatrix}$$

Not unique
(column missed)

$$\begin{pmatrix} \mathbf{1} & ? & ? & ? & ? \\ ? & \mathbf{1} & ? & ? & ? \\ ? & ? & \mathbf{1} & ? & ? \\ ? & ? & ? & \mathbf{1} & ? \\ ? & ? & ? & ? & \mathbf{1} \end{pmatrix}$$

Not unique
(insufficient samples)

Sampling strategy and sample size matter.

When can we hope to recover D? (2)

Consider the following rank-1 matrices and assume **50% revealed entries**.

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

Ok ("incoherent")

$$\begin{pmatrix} 1 & \mathbf{1} & 2 & 1 & 1 \\ \mathbf{1} & \mathbf{1} & \mathbf{2} & \mathbf{1} & 1 \\ 2 & \mathbf{2} & 4 & 2 & \mathbf{2} \\ \mathbf{1} & 1 & \mathbf{2} & \mathbf{1} & \mathbf{1} \\ 1 & 1 & \mathbf{2} & 1 & \mathbf{1} \end{pmatrix}$$

Ok ("incoherent")

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

Bad ("coherent")

$\rightarrow$ first row required

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

Bad ("coherent")

$\rightarrow$ $(1,1)$-entry required

Properties of D matter.

When can we hope to recover $\boldsymbol{D}$? (3)

Exact conditions under which matrix completion "works" is active research area:

- Which sampling schemes? (e.g., random, WR/WOR, active)
- Which sample size?
- Which matrices? (e.g., "incoherent" matrices)
- Noise (e.g., independent, normally distributed noise)

Theorem (Candès and Recht, 2009)

Let $\boldsymbol{D} = \boldsymbol{U}\Sigma\boldsymbol{V}^T$. *If* $\boldsymbol{D}$ *is incoherent in that*

$$\max_{ij} u_{ij}^2 \leq \frac{\mu_B}{n} \qquad \text{and} \qquad \max_{ij} v_{ij}^2 \leq \frac{\mu_B}{n}$$

for some $\mu_B = O(1)$, *and if* $\text{rank}(\boldsymbol{D}) \leq \mu_B^{-1} n^{1/5}$, *then* $O(n^{6/5} r \log n)$ *random samples without replacement suffice to recover* $\boldsymbol{D}$ *exactly with high probability.*

Outline

Overview

Latent factor models in practice

- Millions of users and items
- Billions of ratings
- Sometimes quite complex models

Many algorithms have been applied to large-scale problems

- Gradient descent and quasi-Newton methods
- Coordinate-wise gradient descent
- Stochastic gradient descent
- Alternating least squares

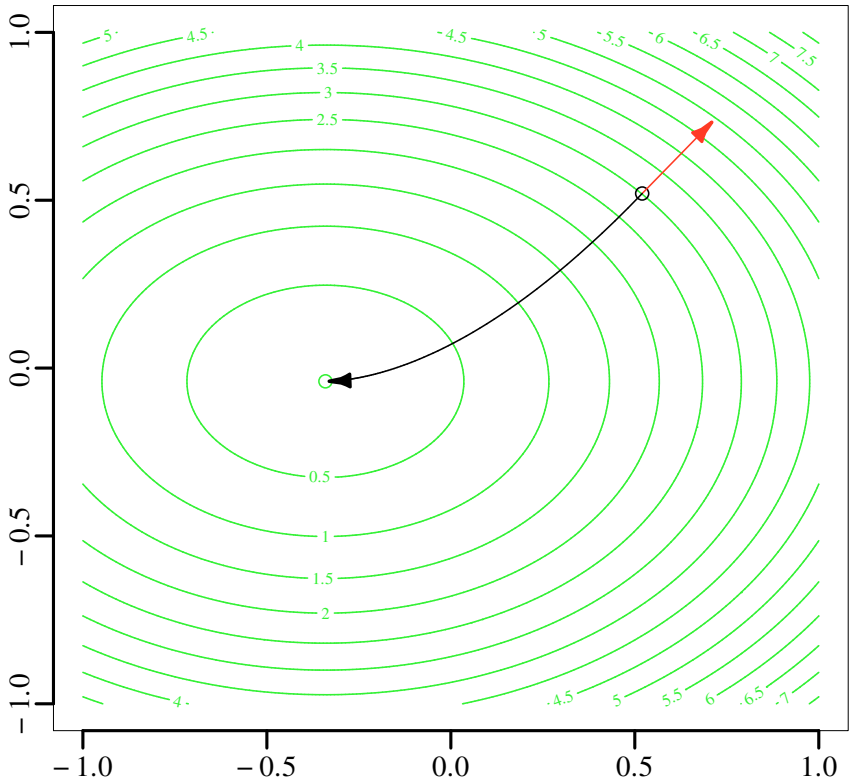
Continuous gradient descent

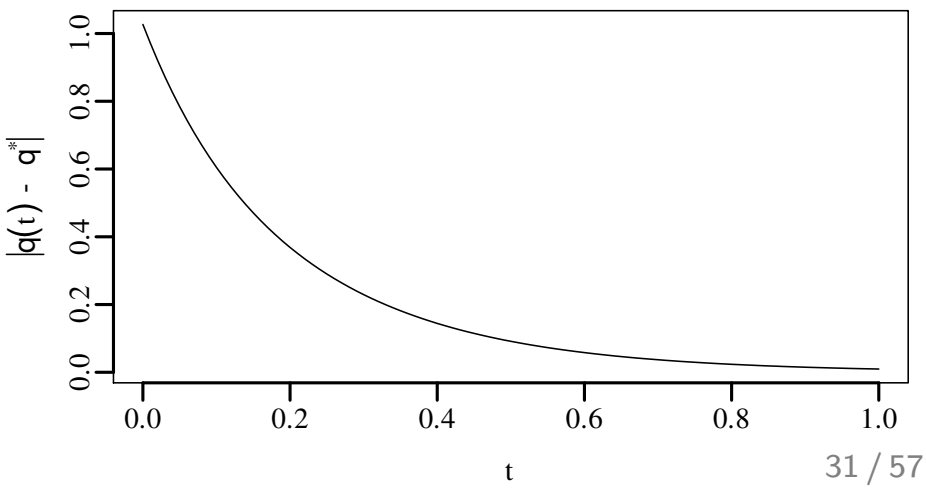
- Find minimum $\boldsymbol{\theta}^*$ of function L
- Pick a starting point $\boldsymbol{\theta}_0$
- Compute gradient $L'(\boldsymbol{\theta}_0)$
- Walk downhill
- Differential equation

$$\frac{\partial \boldsymbol{\theta}(t)}{\partial t} = -L'(\boldsymbol{\theta}(t))$$

  with boundary cond. $\boldsymbol{\theta}(0) = \boldsymbol{\theta}_0$
- Under certain conditions

$$\boldsymbol{\theta}(t) \to \boldsymbol{\theta}^*$$





Discrete gradient descent

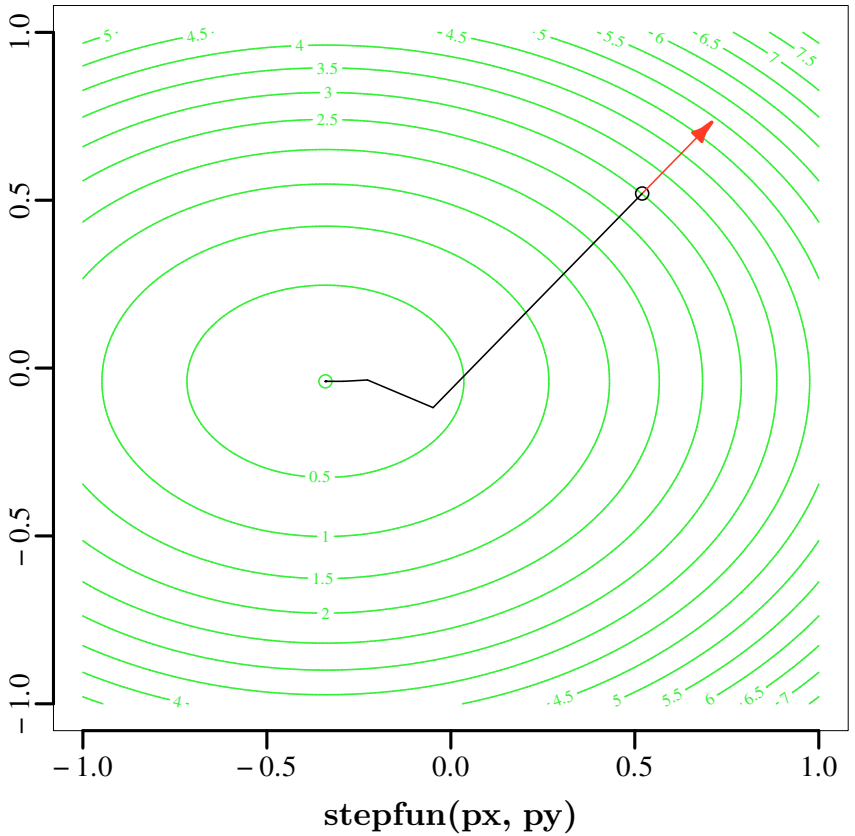
- Find minimum θ^* of function L
- Pick a starting point θ_0
- Compute gradient $L'(\theta_0)$
- Jump downhill
- Difference equation

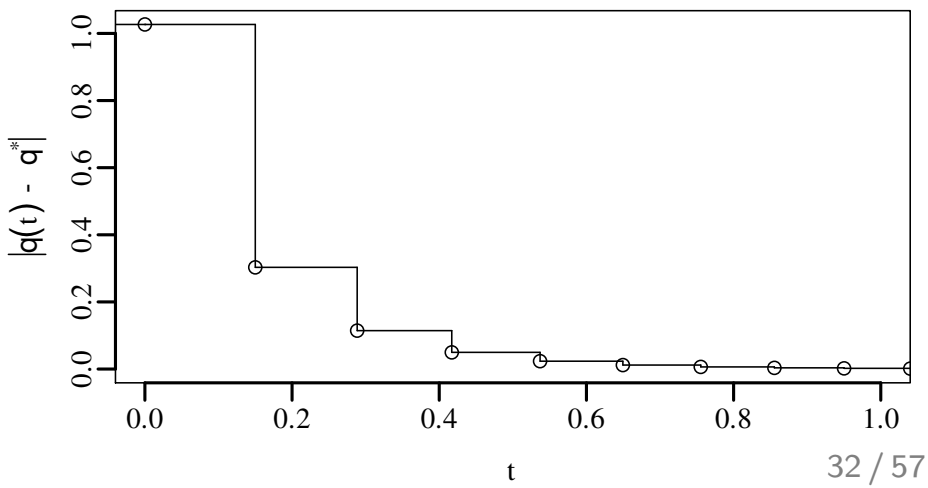
$$\theta_{n+1} = \theta_n - \epsilon_n L'(\theta_n)$$

- Under certain conditions, approximates CGD in that

$$\theta^n(t) = \theta_n + \text{“steps of size } t\text{”}$$

satisfies the ODE as $n \to \infty$





Recap: Gradient computation

- You know: gradient computations for functions on one input; e.g.,

$$f(x) = x^2$$
$$\nabla_x f(x) = 2x$$

- For functions with multiple inputs, there are multiple **partial derivatives**

$$f(l, r) = (d - lr)^2$$
$$\nabla_l f(l, r) = -2(d - lr)r$$
$$\nabla_r f(l, r) = -2(d - lr)l$$

Matrix calculus (1)

- We focus on functions from $\mathbb{R}^n \to \mathbb{R}$
- Let $\boldsymbol{\theta} = \begin{pmatrix} l & r \end{pmatrix}^T$
- Then

$$
\begin{aligned}
f(\boldsymbol{\theta}) &= (d - lr)^2 \\
\nabla_l f(\boldsymbol{\theta}) &= -2(d - lr)r \\
\nabla_r f(\boldsymbol{\theta}) &= -2(d - lr)l
\end{aligned}
$$

- We can write this in matrix form

$$
\nabla_{\boldsymbol{\theta}^T} f = \begin{pmatrix} -2(d - lr)r & -2(d - lr)l \end{pmatrix} = -2(d - lr) \begin{pmatrix} r & l \end{pmatrix}
$$

- The resulting matrix is called the **Jacobian matrix** of f

$$
\boldsymbol{J}_f = -2(d - lr) \begin{pmatrix} r & l \end{pmatrix}
$$

- Note: The Jacobian is a matrix of functions of $\boldsymbol{\theta}$

Matrix calculus (2)

- To recap

$$\boldsymbol{J}_f \stackrel{\text{def}}{=} \nabla_{\boldsymbol{x}^T} f \stackrel{\text{def}}{=} \begin{pmatrix} \nabla_{x_1} f & \nabla_{x_2} f & \cdots & \nabla_{x_n} f \end{pmatrix}$$

- The following rules hold

$$\begin{aligned}
\nabla_{\boldsymbol{x}^T} \quad & c = \begin{pmatrix} 0 & 0 & \cdots & 0 \end{pmatrix} \\
\nabla_{\boldsymbol{x}^T} \quad & \boldsymbol{c}^T \boldsymbol{x} = \boldsymbol{c}^T \\
\nabla_{\boldsymbol{x}^T} \quad & \boldsymbol{x}^T \boldsymbol{c} = \boldsymbol{c}^T \\
\nabla_{\boldsymbol{x}^T} \quad & \boldsymbol{x}^T \boldsymbol{x} = 2\boldsymbol{x}^T \\
\nabla_{\boldsymbol{x}^T} \quad & \boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x} = \boldsymbol{x}^T (\boldsymbol{A} + \boldsymbol{A}^T),
\end{aligned}$$

  where constants c, $\boldsymbol{c}$, and $\boldsymbol{A}$ do not depend on $\boldsymbol{x}$

- Also: multiplicative rule, product rule, chain rule, ...

Gradient computation for LFMs

Set $\boldsymbol{\theta} = (\boldsymbol{L}, \boldsymbol{R})$, $L(\boldsymbol{\theta}) = L(\boldsymbol{L}, \boldsymbol{R})$, and write

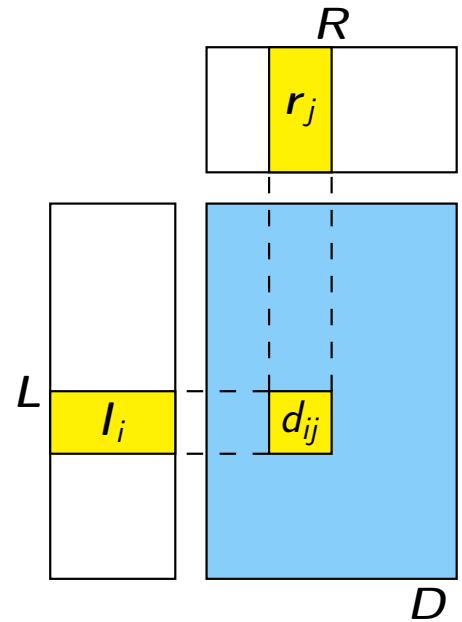
$$L(\boldsymbol{\theta}) = \sum_{(i,j)\in\Omega} L_{ij}(\boldsymbol{l}_i, \boldsymbol{r}_j)$$

$$L'(\boldsymbol{\theta}) = \sum_{(i,j)\in\Omega} L'_{ij}(\boldsymbol{l}_i, \boldsymbol{r}_j)$$

$$\nabla_{l_{ik}} L(\boldsymbol{\theta}) = \sum_{(i',j)\in\Omega} \nabla_{l_{ik}} L_{i'j}(\boldsymbol{l}_{i'}, \boldsymbol{r}_j)$$

$$= \sum_{j\in\{\, j' \mid (i,j')\in\Omega \,\}} \nabla_{l_{ik}} L_{ij}(\boldsymbol{l}_i, \boldsymbol{r}_j)$$

since $\nabla_{l_{ik}} L_{i'j}(\boldsymbol{l}_{i'}, \boldsymbol{r}_j) = 0$ for $i \neq i'$



Local gradient of entry $(i, j) \in \Omega$ nonzero only for row $\boldsymbol{l}_i$ and column $\boldsymbol{r}_j$.

Example gradient computation

Simplest form of LFM (unregularized)

$$L_{ij}(\boldsymbol{l}_i, \boldsymbol{r}_j) = (d_{ij} - \boldsymbol{l}_i^T \boldsymbol{r}_j)^2$$

Gradient computation

$$\nabla_{l_{ik}} L_{ij}(\boldsymbol{l}_i, \boldsymbol{r}_j) = -2(d_{ij} - \boldsymbol{l}_i^T \boldsymbol{r}_j) r_{kj}$$
$$\nabla_{r_{kj}} L_{ij}(\boldsymbol{l}_i, \boldsymbol{r}_j) = -2 l_{ik} (d_{ij} - \boldsymbol{l}_i^T \boldsymbol{r}_j)$$

$$\nabla_{\boldsymbol{l}_i^T} L_{ij}(\boldsymbol{l}_i, \boldsymbol{r}_j) = -2(d_{ij} - \boldsymbol{l}_i^T \boldsymbol{r}_j) \boldsymbol{r}_j^T$$
$$\nabla_{\boldsymbol{r}_j^T} L_{ij}(\boldsymbol{l}_i, \boldsymbol{r}_j) = -2\boldsymbol{l}_i^T (d_{ij} - \boldsymbol{l}_i^T \boldsymbol{r}_j)$$

Gradient descent for LFMs

GD epoch

1. Compute gradient
   - Initialize zero matrices $\boldsymbol{L}^{\nabla}$ and $\boldsymbol{R}^{\nabla}$
   - For each entry $(i, j) \in \Omega$, update gradients

$$\boldsymbol{l}_i^{\nabla} \leftarrow \boldsymbol{l}_i^{\nabla} + \nabla_{\boldsymbol{l}_i} L_{ij}(\boldsymbol{l}_i, \boldsymbol{r}_j)$$
$$\boldsymbol{r}_j^{\nabla} \leftarrow \boldsymbol{r}_j^{\nabla} + \nabla_{\boldsymbol{r}_j} L_{ij}(\boldsymbol{l}_i, \boldsymbol{r}_j)$$

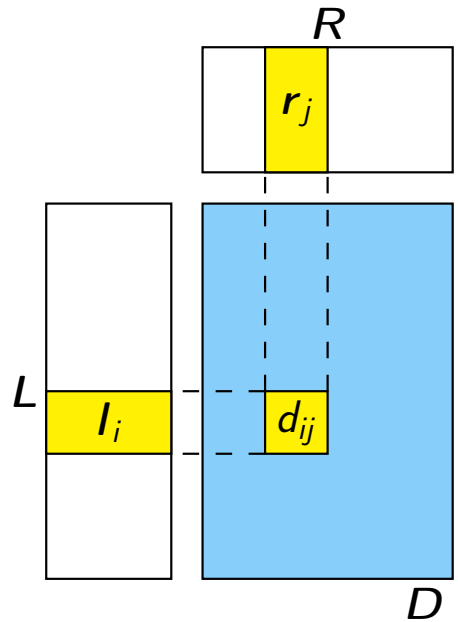
   - After this step

$$\boldsymbol{l}_i^{\nabla} = \sum_{j \in \{ j' \mid (i,j') \in \Omega \}} \nabla_{\boldsymbol{l}_i} L_{ij}(\boldsymbol{l}_i, \boldsymbol{r}_j)$$

   as desired ($\boldsymbol{r}_j^{\nabla}$ analog)

2. Update parameters

$$\boldsymbol{L} \leftarrow \boldsymbol{L} - \epsilon_n \boldsymbol{L}^{\nabla}$$
$$\boldsymbol{R} \leftarrow \boldsymbol{R} - \epsilon_n \boldsymbol{R}^{\nabla}$$

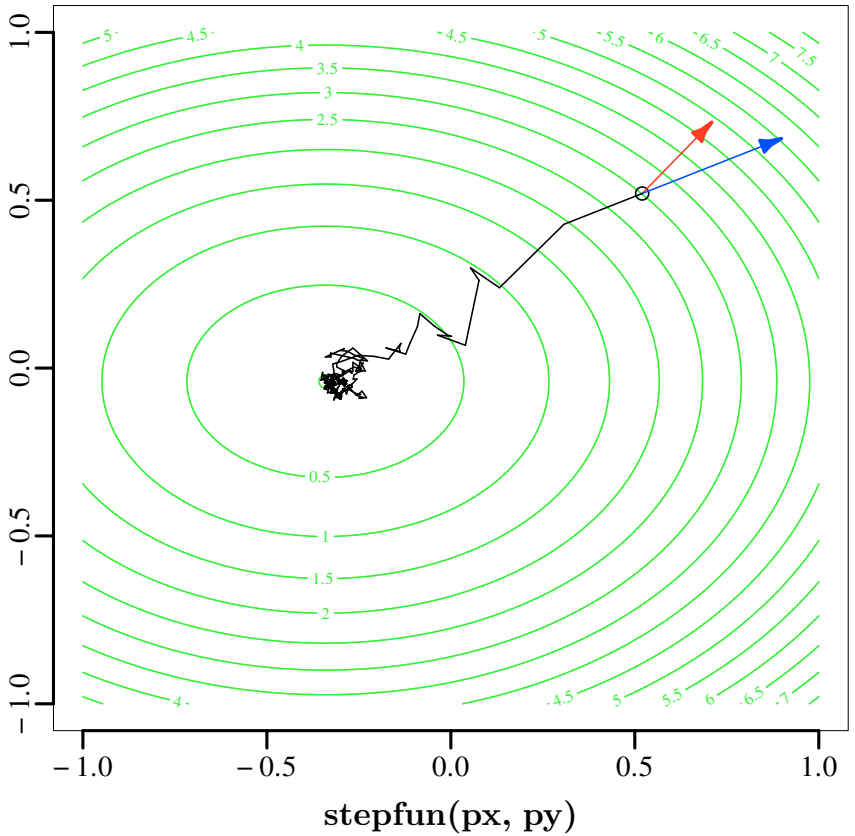


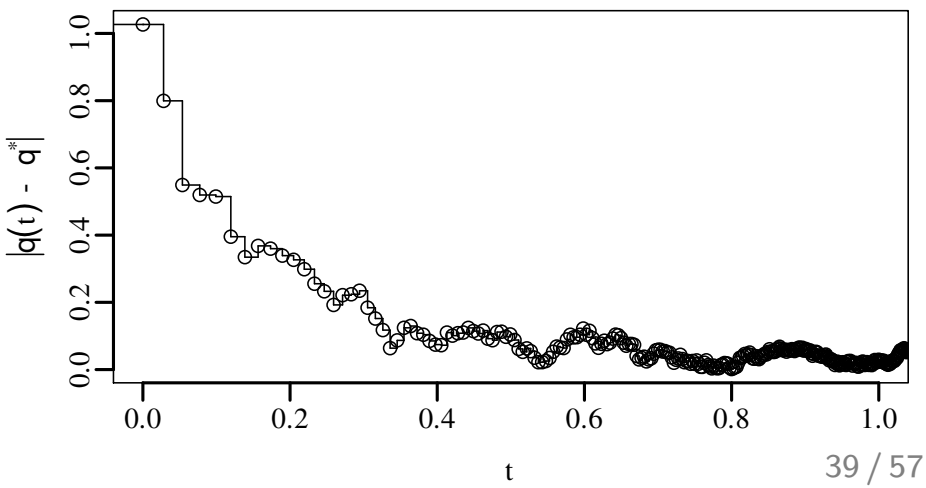
Stochastic gradient descent

- Find minimum $\boldsymbol{\theta}^*$ of function L
- Pick a starting point $\boldsymbol{\theta}_0$
- Approximate gradient $\hat{L}'(\boldsymbol{\theta}_0)$
- Jump "approximately" downhill
- Stochastic difference equation

$$\boldsymbol{\theta}_{n+1} = \boldsymbol{\theta}_n - \epsilon_n \hat{L}'(\boldsymbol{\theta}_n)$$

- Under certain conditions, asymptotically approximates (continuous) gradient descent





Stochastic gradient descent for LFMs

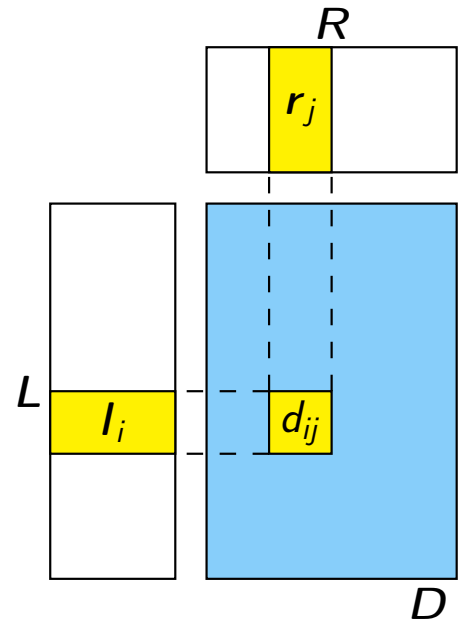
- Set $\boldsymbol{\theta} = (\boldsymbol{L}, \boldsymbol{R})$ and use

$$L(\boldsymbol{\theta}) = \sum_{(i,j)\in\Omega} L_{ij}(\boldsymbol{l}_i, \boldsymbol{r}_j)$$

$$L'(\boldsymbol{\theta}) = \sum_{(i,j)\in\Omega} L'_{ij}(\boldsymbol{l}_i, \boldsymbol{r}_j)$$

$$\hat{L}'(\boldsymbol{\theta}, z) = N L'_{i_z j_z}(\boldsymbol{l}_{i_z}, \boldsymbol{r}_{j_z}),$$

where $N = |\Omega|$ and $z = (i_z, j_z) \in \Omega$



- SGD epoch (with or without replacement)
  1. Pick a random entry $z \in \Omega$ (with or without replacement)
  2. Compute approximate gradient $\hat{L}'(\boldsymbol{\theta}, z)$
  3. Update parameters
$$\boldsymbol{\theta}_{n+1} = \boldsymbol{\theta}_n - \epsilon_n \hat{L}'(\boldsymbol{\theta}_n, z)$$
  4. Repeat N times

SGD step affects only current row and column.

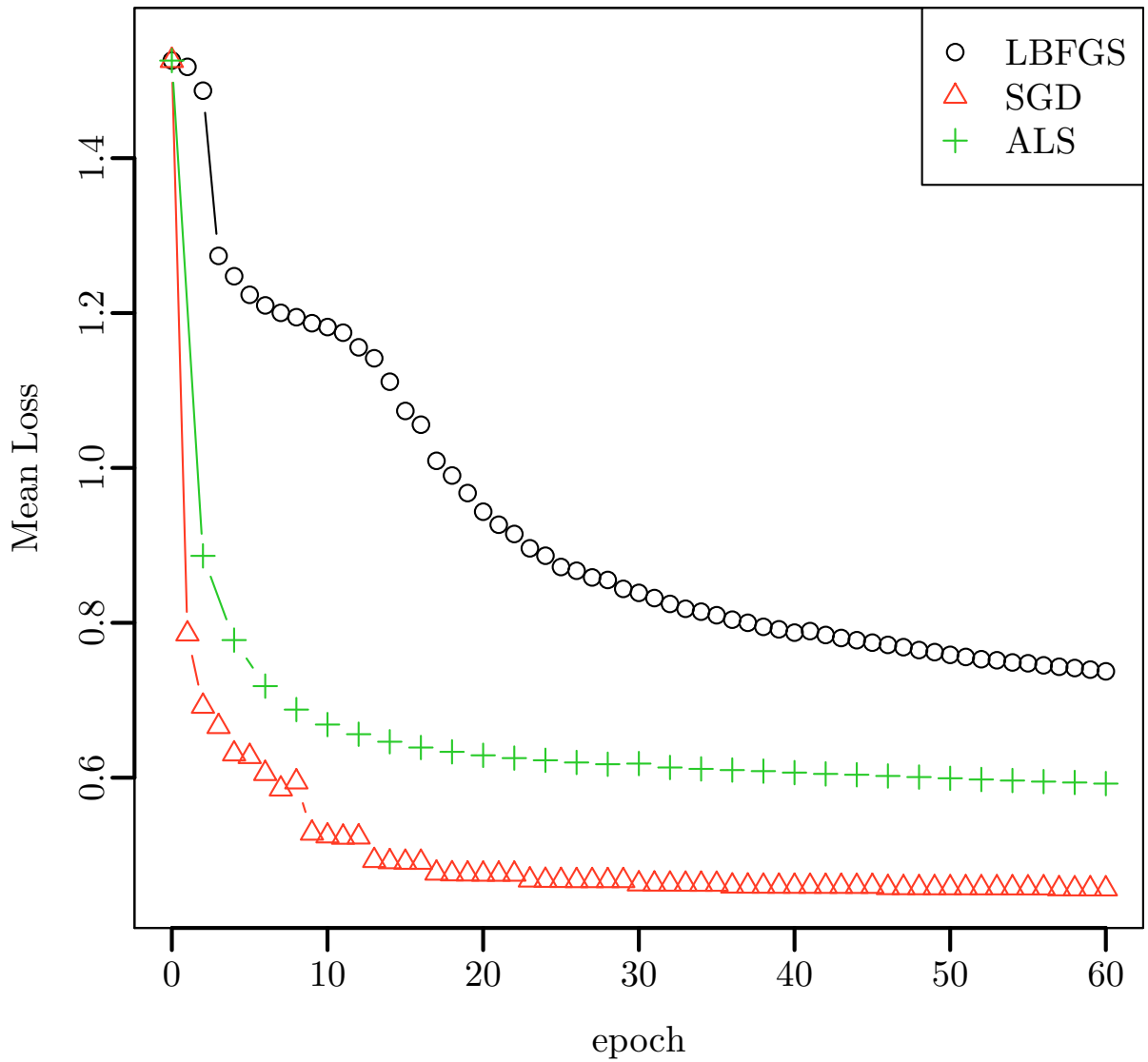
Comparison

- Per epoch, assuming $O(r)$ gradient computation per element

	GD	**SGD**
Algorithm	Deterministic	Randomized
Gradient computations	1	N
Gradient types	Exact	Approximate
Parameter updates	1	N
Time	$O(rN)$	$O(rN)$
Space	$O((m+n)r)$	$O((m+n)r)$

- Why stochastic?
  - *Fast convergence* to vicinity of optimum
  - Randomization may help escape local minima
  - Exploitation of "repeated structure"

Example: Netflix data, unregularized



GD/SGD in practice (1)

Step size (or **learning rate**) sequence $\{\epsilon_n\}$ needs to be chosen carefully; widely studied, many options:

- Large $\rightarrow$ good initially (move quickly), bad later on (juggle around optimum)
- Keep step size throughout or reduce it gradually
  - E.g., constant (useful for online learning)
  - E.g., $\epsilon_n = a/(b+n)$ for some constants a, b
  - E.g., pick $\epsilon_n \leq 1/L(\nabla f)$ if f has bounded gradient
- **Bold driver heuristic**: After every epoch
  - Increase step size slightly when loss decreased (by, say, 5%)
  - Decrease step size sharply when loss increased (by, say, 50%)
  - Not provably correct, but works well in practice
- Pick initial step size based on sample
- **Line search:** optimize the step size directly

$$\epsilon_n = \operatorname*{argmin}_{\epsilon} L(\boldsymbol{\theta}_n - \epsilon L'(\boldsymbol{\theta}_n))$$

GD/SGD in practice (2)

- SGD is a common learning algorithm
  - E.g., training neural networks
  - Related to **incremental gradient descent** and **online learning**
- Many variants exist; e.g.,
  - Use more than one example per step (**mini-batch**)
  - Polyak averaging
  - Momentum
  - Adaptive, per-parameter step sizes (AdaGrad, RMSprop, AdaDelta, Adam)
  - ...
- And it can (often) be parallelized; e.g.,
  - Hogwild
  - Vowpal Wabbit (for regression problems)
  - DSGD++ (for latent factor models)
    E.g.: 10M$\times$1M, 10B observed entries, $\approx$ 1h on 8 machines
  - ...

Outline

Variants of latent factor models

- In practice, the basic latent factor model is often modified in an application-dependent way
- We discuss two variants here
  - How to handle **implicit feedback**?
    (All observed entries are 1.)
  - How to make use of additional **contextual information**?
    (E.g., attributes for users and items.)
- There are many other variants; e.g.
  - Bayesian variants (partly discussed later)
  - Combination of explicit and implicit feedback
  - Additional constraints (e.g., non-negativity of factors)
  - ...
- Picking the right model requires thought, experience, and experimentation

Implicit feedback

- Implicit feedback is an indicator for preference
  - Which product pages do users look at?
  - Which movies do users watch?
  - How much time do users spend on a news article?
- Absence of implicit feedback is **not** necessarily an indicator for non-preference
- In the basic latent factor model, we used explicit feedback, which contains
  - **Positive evidence** (a user gave an item a high rating)
  - **Negative evidence** (a user gave an item a low rating)
- If we only have implicit feedback
  - We may think of the implicit feedback as positive evidence
  - But we do not have negative evidence
  - This implies that the basic latent factor model won't work

Implicit feedback and the basic LFM

- A matrix of implicit feedback (1=present, ?=absent)

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

- Its completion by an (unregularized) LFM

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

- That's not helpful!

How to handle implicit feedback

- One option is to build a different model
  - E.g., a nearest-neighbor model
- We can also learn variant of an LFM
  - Good approach if low-rank assumption applies
- How? Move from "predicting values" to "ranking values"
  - Ranking is per row (e.g, the items for each user) or per column
  - Goal is not to predict the preference of a user for an item, but to rank items such that high-preference items appear early on
- Key idea is to find a ranking that
  - Tends to rank items with implicit feedback before items without
  - Is simple (e.g., can be described with a low-rank matrix)

Bayesian personalized ranking (intuition)

- Goal is to learn a personalized ordering $>_i$ for each user i
  - Consider a user i, an item j_1 and an item j_2
  - Idea: $j_1 >_i j_2$ if user i prefers item j_1 over item j_2
  - Assumption: If user i provided implicit feedback for j_1 but not for j_2, then he prefers j_1 over j_2
  - We want to learn $>_i$ for all pairs of items
  - Ordering should be total, antisymmetric, and transitive

- The BPR model
  - Associate each triple (i, j_1, j_2) with a **score** $\hat{x}_{ij_1j_2} \in \mathbb{R}$
  - Score is $\begin{cases} \text{positive} & \text{if } j_1 \text{ is preferred by user } i \\ \text{negative} & \text{if } j_2 \text{ is preferred by user } i \\ 0 & \text{if oblivious} \end{cases}$
  - Model **probability** that i prefers j_1 over j_2 as

$$p(j_1 >_i j_2) = \sigma(\hat{x}_{ij_1j_2}),$$

where $\sigma(x) = 1/(1 + \exp(-x)) \in [0, 1]$ is the logistic function

BPR for latent factor models

- Use low-rank matrix factorization to model scores

$$\hat{x}_{ij_1j_2} = \boldsymbol{l}_i^T \boldsymbol{r}_{j_1} - \boldsymbol{l}_i^T \boldsymbol{r}_{j_2},$$

  where as before $\boldsymbol{L}$ and $\boldsymbol{R}$ are factor matrices

- Construct database $D \subseteq [m] \times [n] \times [n]$ of "observed orderings"
  - $(i, j^+, j^-) \in D$ if user i has implicit feedback for j^+ but not for j^-
  - Can be large, but does not really need to be constructed

- Under certain assumptions, maximum likelihood estimate for $\boldsymbol{L}$ and $\boldsymbol{R}$ is

$$\begin{aligned} &\operatorname*{argmax}_{\boldsymbol{L} \in \mathbb{R}^{m \times r}, \boldsymbol{R} \in \mathbb{R}^{r \times n}} \prod_{(i,j^+,j^-) \in D} p(j^+ >_i j^-) \\ = &\operatorname*{argmax}_{\boldsymbol{L} \in \mathbb{R}^{m \times r}, \boldsymbol{R} \in \mathbb{R}^{r \times n}} \sum_{(i,j^+,j^-) \in D} \log p(j^+ >_i j^-) \end{aligned}$$

- We can use stochastic gradient ascent to find the estimates
  - Gradient estimate obtained by sampling a single triple
  - In practice, add regularization

BPR in action

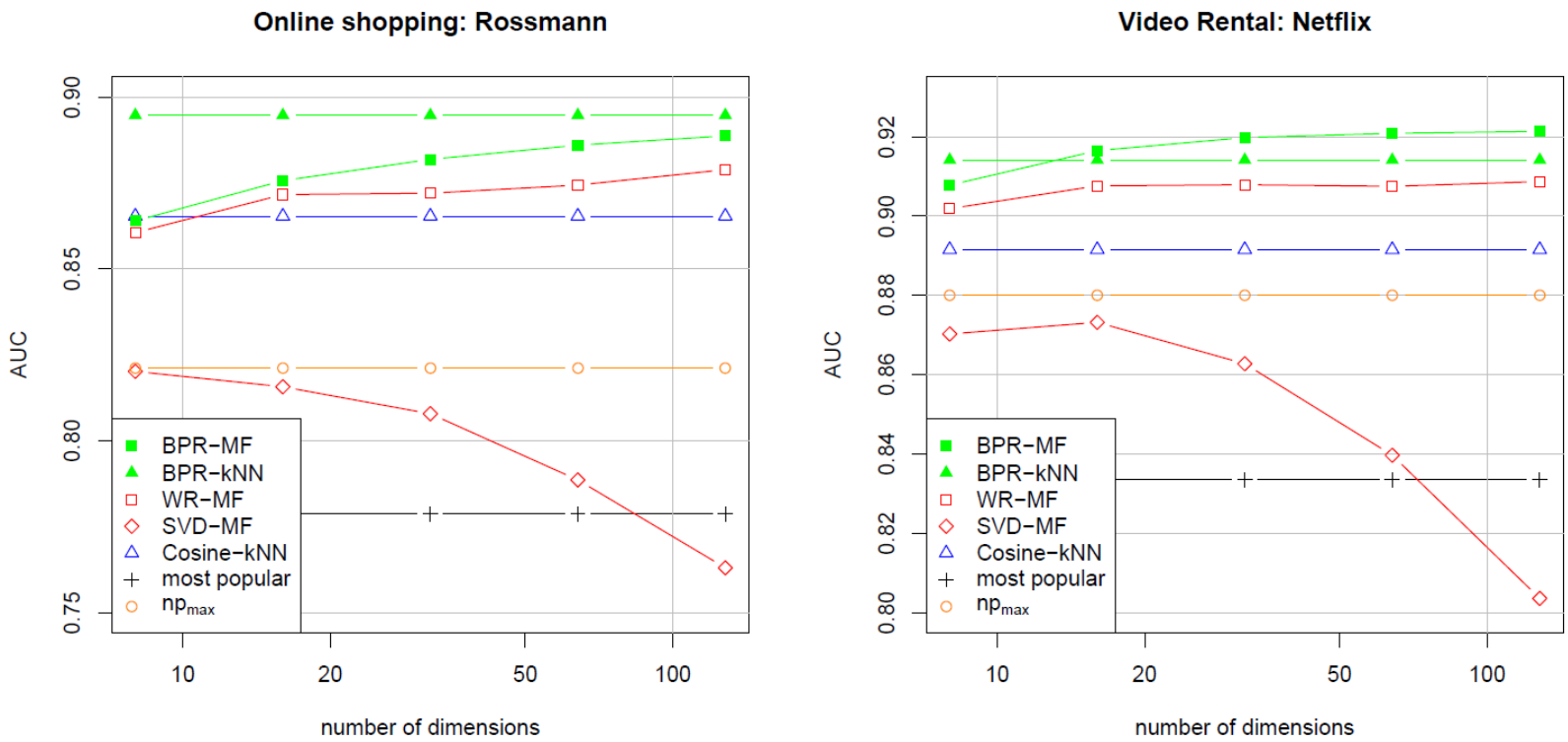


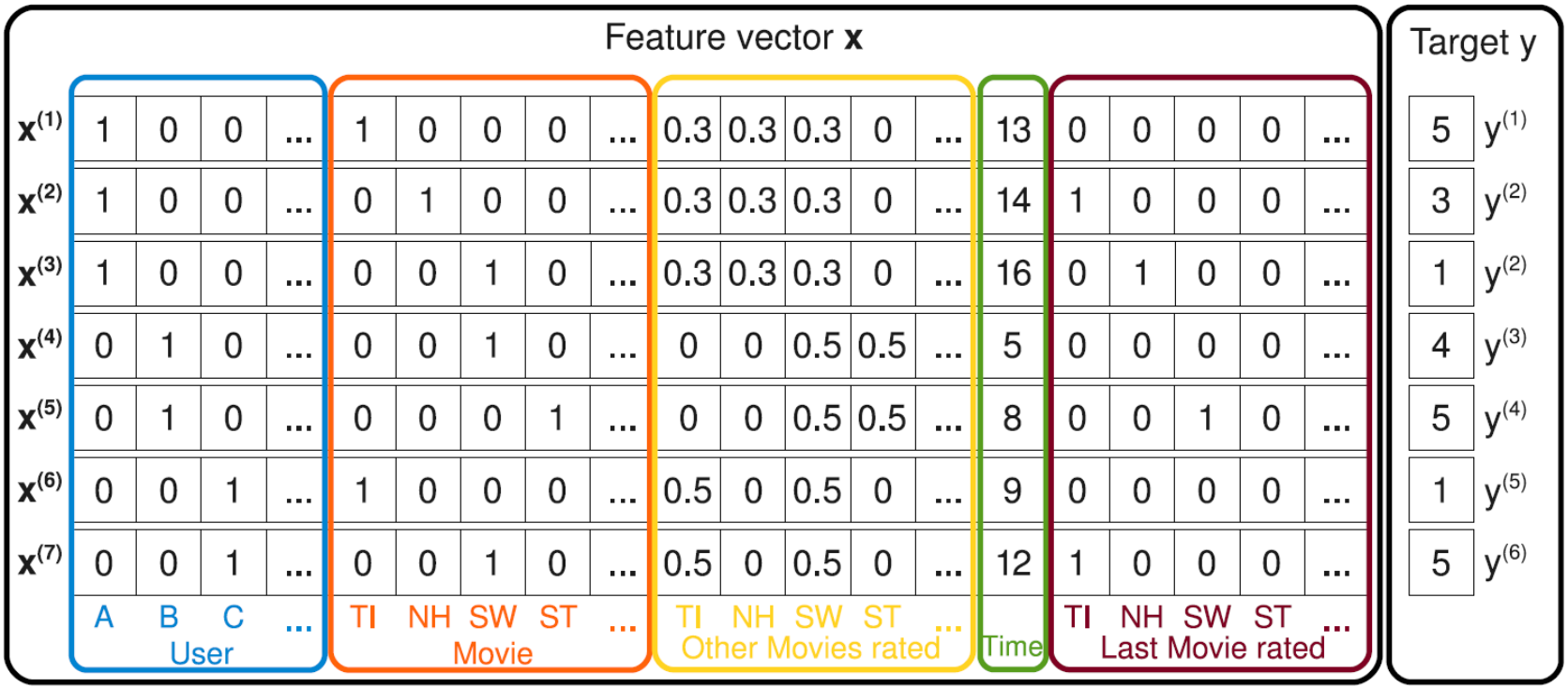
Figure 6: Area under the ROC curve (AUC) prediction quality for the Rossmann dataset and a Netflix subsample. Our BPR optimizations for matrix factorization BPR-MF and k-nearest neighbor BPR-kNN are compared against weighted regularized matrix factorization (WR-MF) [5, 10], singular value decomposition (SVD-MF), k-nearest neighbor (Cosine-kNN) [2] and the most-popular model. For the factorization methods BPR-MF, WR-MF and SVD-MF, the model dimensions are increased from 8 to 128 dimensions. Finally, np_{max} is the theoretical upper bound for any non-personalized ranking method.

Rendle et al., 2009

Contextual information

- We often have additional information; e.g.,
  - Demographics of users (age, sex, city, . . .)
  - Information about items (e.g., genre, actors, directors, . . .)
- How to exploit this additional information with LFM models?
  1. Build a separate model and combine predictions (**stacking**)
  2. As above, but but learn model parameters jointly
  3. Extend LFMs to directly incorporate context
- A popular approach for (3) are factorization machines
  - Open source implementation: libfm
  - Can be combined with BPR
  - Advertisement: also useful for extracting relations from natural-language text

Factorization machines (intuition)



- Rows = observed entries
- Columns = information about observed entry
  - Which user rated which item? In what context?
  - Must be constructable at prediction time for unobserved entries (!)
- Associate a latent feature vector with each column
  - Prediction $\approx$ sum of pairwise inner products of weighted feature vectors
  - As before, learn latent features using SGD to fit observed data

Rendle, 2010

Outline

Lessons learned

- Collaborative filtering methods learn from past user behavior
  - Latent factor models are best-performing single approach
  - Many variants exist
  - In practice, often combined with other methods
- Users and items are represented in common latent factor space
  - Holistic matrix-factorization approach
  - Similar users/item placed at similar positions
  - Low-rank assumption = few "factors" influence user preferences
- Close relationship to matrix completion problem
  - Reconstruct a partially observed low-rank matrix
  - Many applications
- SGD is simple and practical algorithm to solve LFMs in summation form

Suggested reading

- Y. Koren, R. Bell, C. Volinsky
  Matrix factorization techniques for recommender systems
  IEEE Computer, 42(8), p. 30–37, 2009
  http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5197422

- E. Candès, B. Recht
  Exact matrix completion via convex optimization
  Communications of the ACM, 55(6), p. 111–119, 2012
  http://doi.acm.org/10.1145/2184319.2184343

- https://en.wikipedia.org/wiki/Matrix_calculus

- And references in the above articles and slides

Data Mining and Matrices

05 – Non-Negative Matrix Factorization

Prof. Dr. Rainer Gemulla

Universität Mannheim

FSS 2018

Non-negative datasets

Some datasets are intrinsically non-negative:

- Counters
  (e.g., no. occurrences of each word in a text document)
- Quantities
  (e.g., amount of each ingredient in a chemical experiment)
- Intensities
  (e.g., intensity of each color in an image)

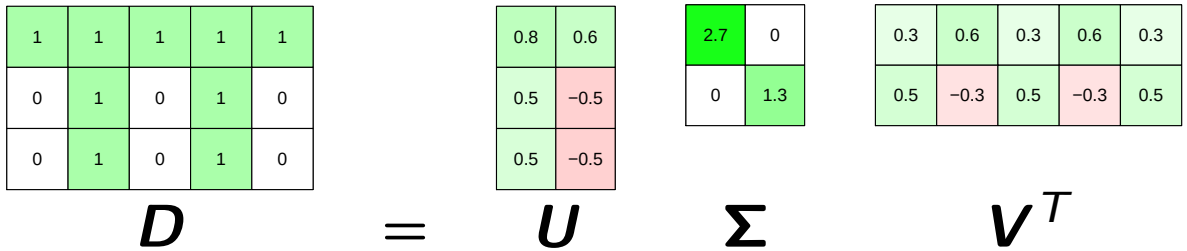
The corresponding data matrix $\boldsymbol{D}$ has only non-negative values.

- Decompositions such as SVD may involve negative values in factors and components
- Negative values describe the absence of something
- Often no natural interpretation

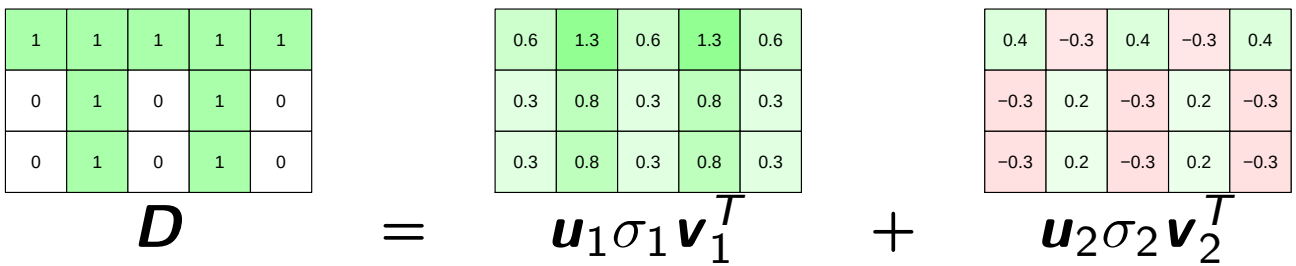
Goal: Find a decomposition that is more natural to non-negative data.

Example (SVD)

Consider the following "bridge" matrix and its compact SVD:



Here are the corresponding components:



Negative values can make interpretation unnatural or difficult.

Outline

Non-negative matrix factorization (NMF)

Definition (Non-negative matrix factorization, basic form)

Given a non-negative matrix $\boldsymbol{D} \in \mathbb{R}_+^{m \times n}$, a **non-negative matrix factorization** of rank k is

$$\boldsymbol{D} \approx \boldsymbol{L}\boldsymbol{R},$$

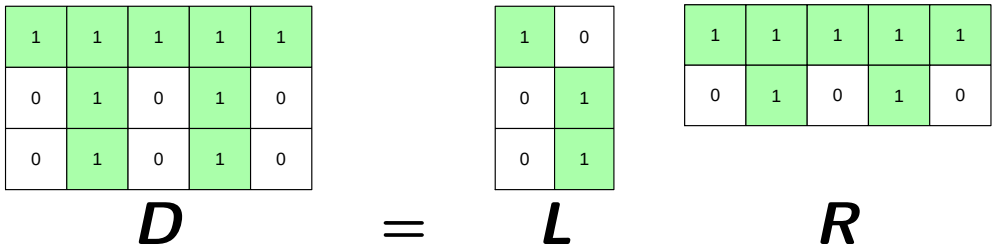
where $\boldsymbol{L} \in \mathbb{R}_+^{m \times r}$ and $\boldsymbol{R} \in \mathbb{R}_+^{r \times n}$ are both non-negative.

- Additive decomposition: factors and components non-negative
  $\rightarrow$ No cancellation effects
- Rows of $\boldsymbol{R}$ can be thought as "parts"
- Row of $\boldsymbol{D}$ obtained by mixing (or "assembling") parts as described in $\boldsymbol{L}$
- Smallest r such that $\boldsymbol{D} = \boldsymbol{L}\boldsymbol{R}$ exists is called **non-negative rank** of $\boldsymbol{D}$

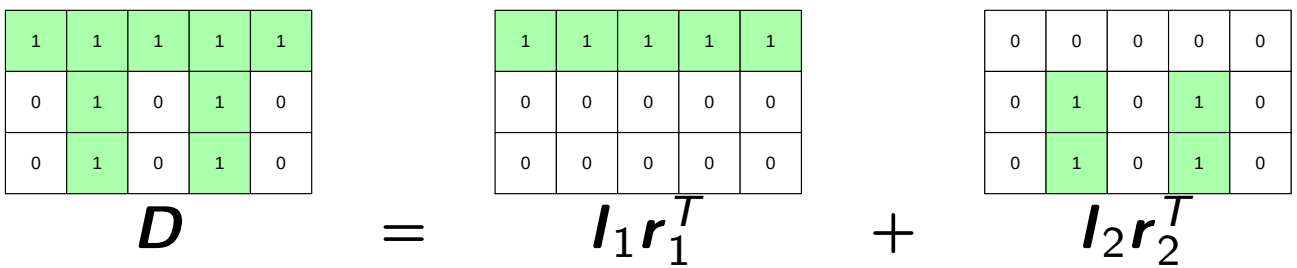
$$\operatorname{rank}(\boldsymbol{D}) \leq \operatorname{rank}_+(\boldsymbol{D}) \leq \min\{m, n\}$$

Example (NMF)

Consider the following "bridge" matrix and its rank-2 NMF:



Here are the corresponding components:



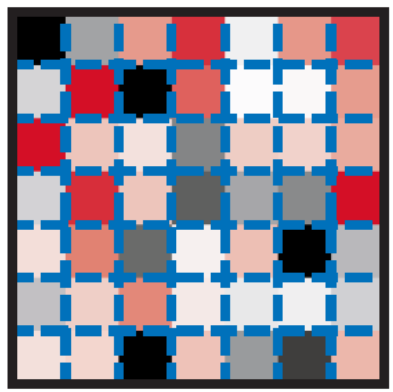
Non-negative matrix decomposition encourage a more natural, part-based representation and (sometimes) sparsity.

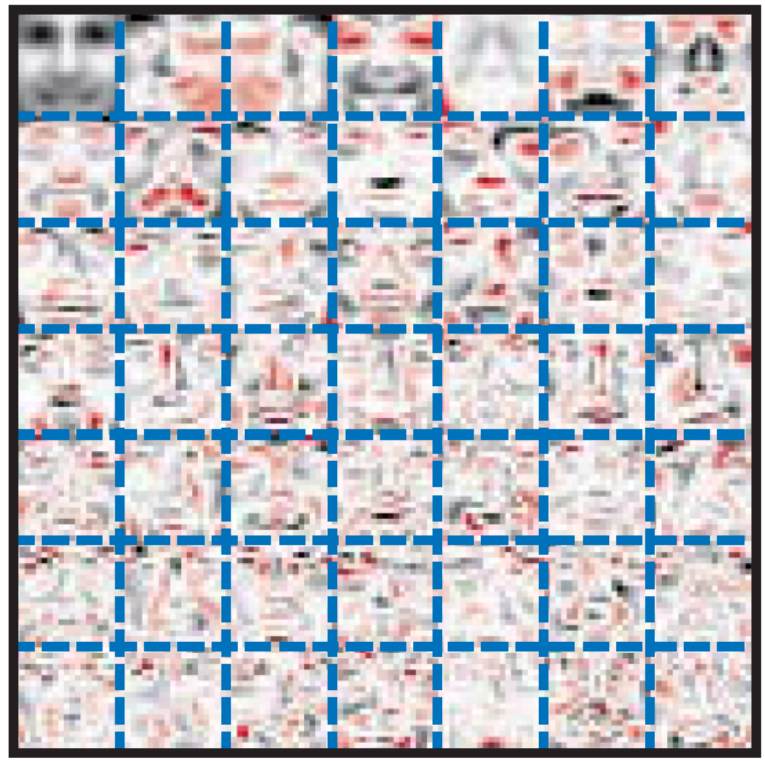
Decomposing faces (SVD)

$\boldsymbol{d}_i^T$ (original)

$[\boldsymbol{LR}]_{i*}$ $=$ $\boldsymbol{l}_i^T$ $\boldsymbol{R}$

$[\boldsymbol{U\Sigma V}^T]_{i*}$ $=$ $\boldsymbol{u}_i^T\boldsymbol{\Sigma}$ $\boldsymbol{V}^T$





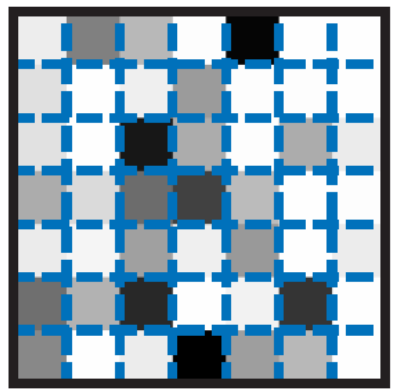
SVD factors can be hard to interpret.

Lee and Seung, 1999.

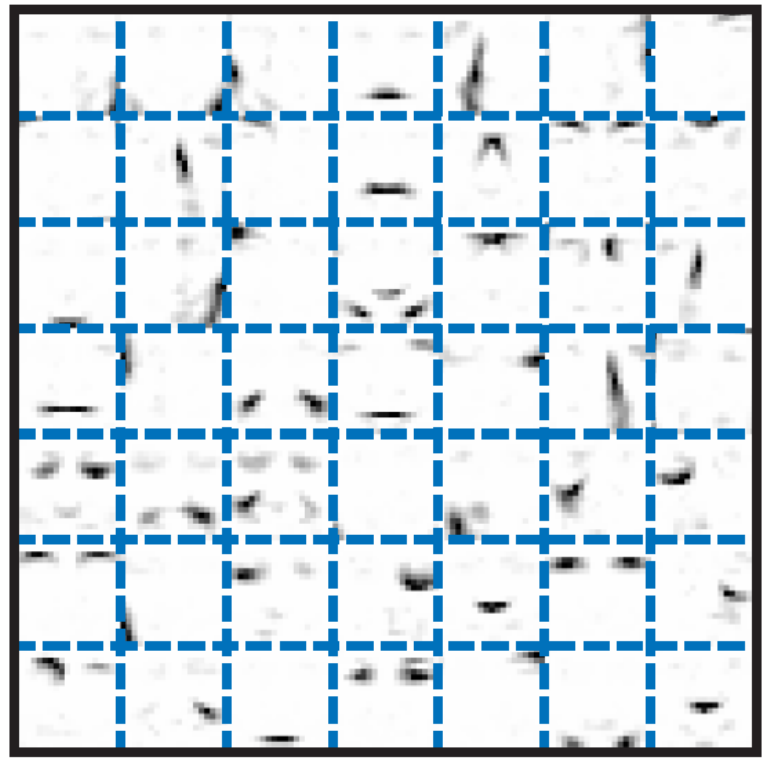
Decomposing faces (NMF)

$\boldsymbol{d}_i^T$ (original)

$[\boldsymbol{LR}]_{i*}$ $=$



$\boldsymbol{l}_i^T$



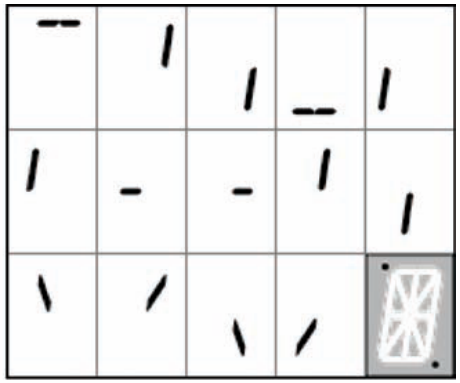
$\boldsymbol{R}$

NMF factors correspond to parts of faces.

Lee and Seung, 1999.

Decomposing digits (NMF)

D



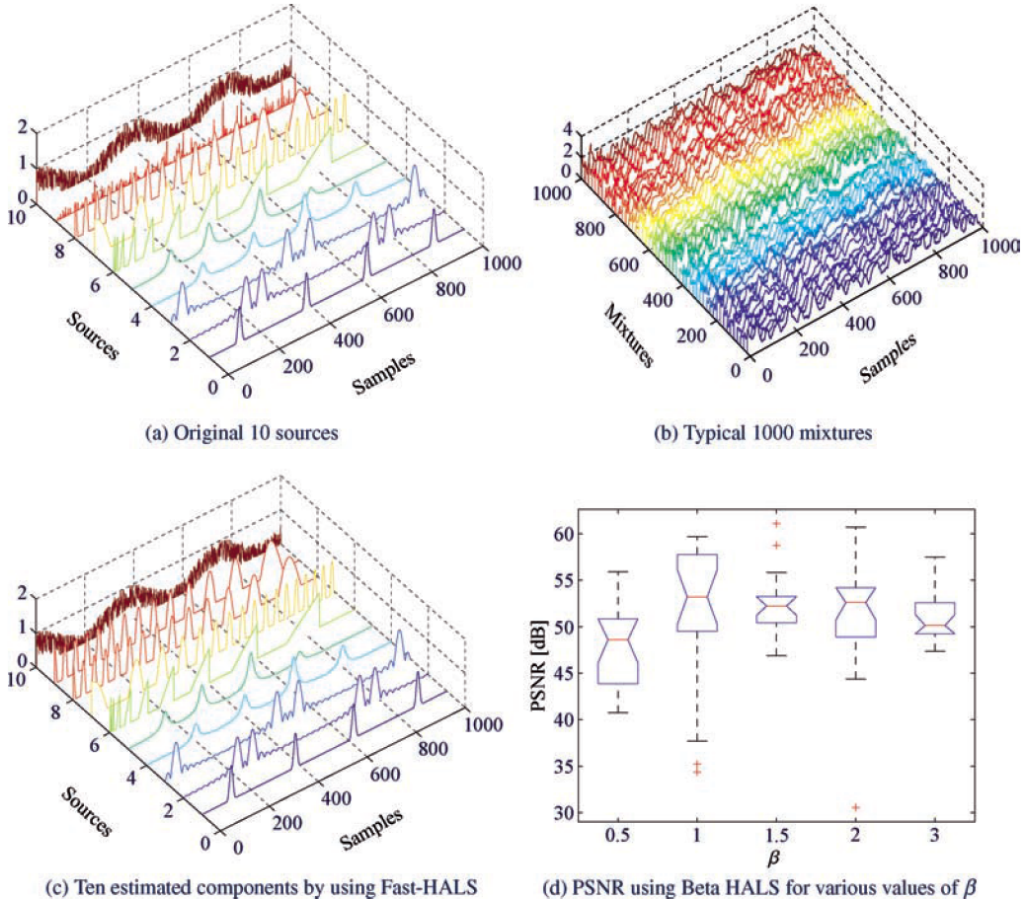
R

NMF factors correspond to parts of digits and "background".

Cichocki et al., 2009.

Some applications

- Text mining (more later)
- Bioinformatics
- Microarray analysis
- Mineral exploration
- Neuroscience
- Image understanding
- Air pollution research
- Chemometrics
- Spectral data analysis
- Linear sparse coding
- Image classification
- Clustering
- Neural learning process
- Sound recognition
- Remote sensing
- Object characterization
- ...



(a) Original 10 sources (b) Typical 1000 mixtures

(c) Ten estimated components by using Fast-HALS (d) PSNR using Beta HALS for various values of β

Figure 4.8 Illustration for (a) benchmark used in large-scale experiments with 10 nonnegative sources; (b) Typical 1000 mixtures; (c) Ten estimated components by using FAST HALS NMF from the observations matrix $\mathbf{Y}$ of dimension 1000×1000. (d) Performance expressed via the PSNR using the Beta HALS NMF algorithm for $\beta = 0.5, 1, 1.5, 2$ and 3.

Cichocki et al., 2009.

Gaussian NMF

- Gaussian NMF is the most basic form of non-negative factorizations:

$$\begin{aligned} \text{minimize} \quad & \|\boldsymbol{D} - \boldsymbol{L}\boldsymbol{R}\|_F^2 \\ \text{s.t.} \quad & \boldsymbol{L} \in \mathbb{R}_+^{m \times r} \\ & \boldsymbol{R} \in \mathbb{R}_+^{r \times n} \end{aligned}$$

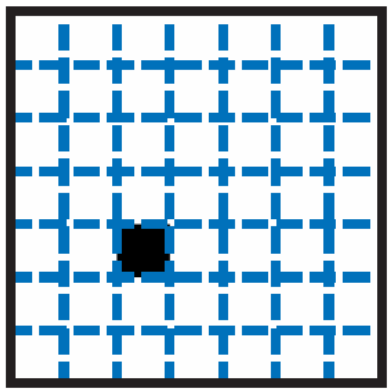
- Truncated SVD minimizes the same objective (but without non-negativity constraints)
- Many other variants exist
  - Different objective functions (e.g., KL-divergence)
  - Additional regularizations (e.g., L_1-regularization)
  - Different constraints (e.g., orthogonality of $\boldsymbol{R}$)
  - Different compositions (e.g., 3 matrices)
  - multi-layer NMF, semi-NMF, sparse NMF, tri-NMF, symmetric NMF, orthogonal NMF, non-smooth NMF (nsNMF), overlapping NMF, convolutive NMF (CNMF), k-Means, ...

k-Means can be seen as a variant of NMF

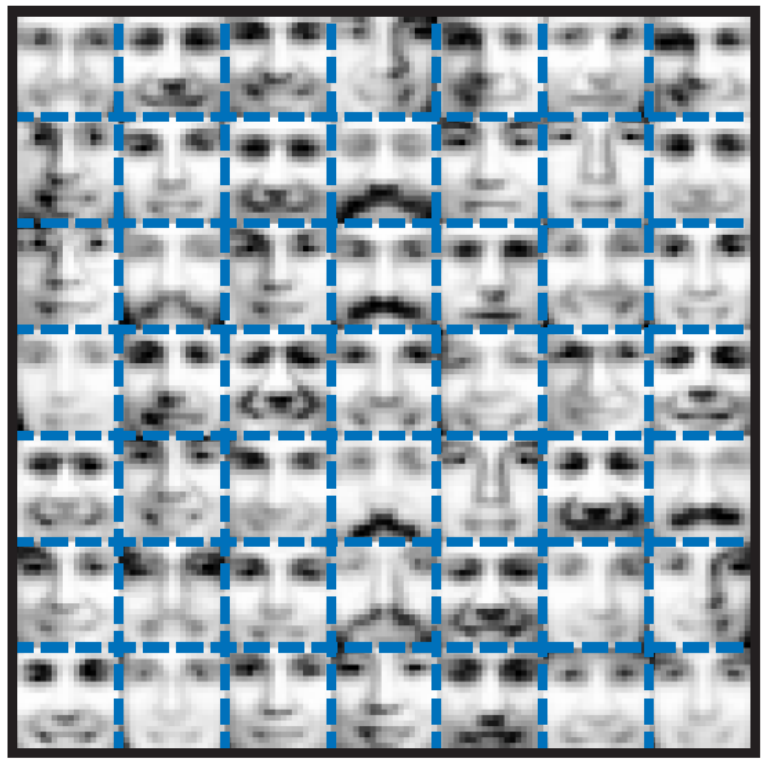
$\boldsymbol{d}_i^T$ (original)

Additional constraint: $\boldsymbol{L}$ contains exactly one 1 in each row, rest 0

$[\boldsymbol{LR}]_{i*}$ $=$



$\boldsymbol{I}_i^T$



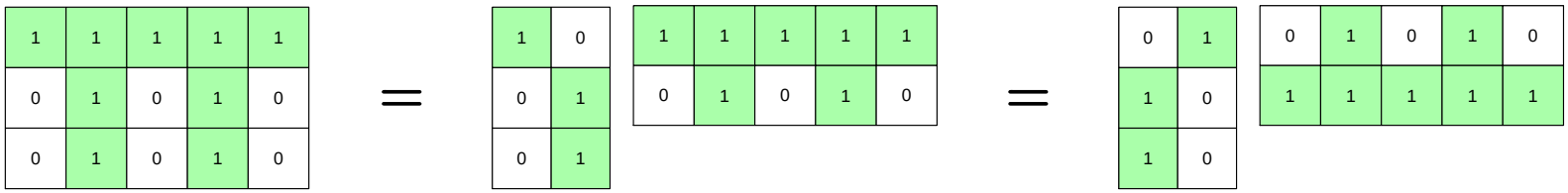
$\boldsymbol{R}$

k-Means factors correspond to prototypical faces.

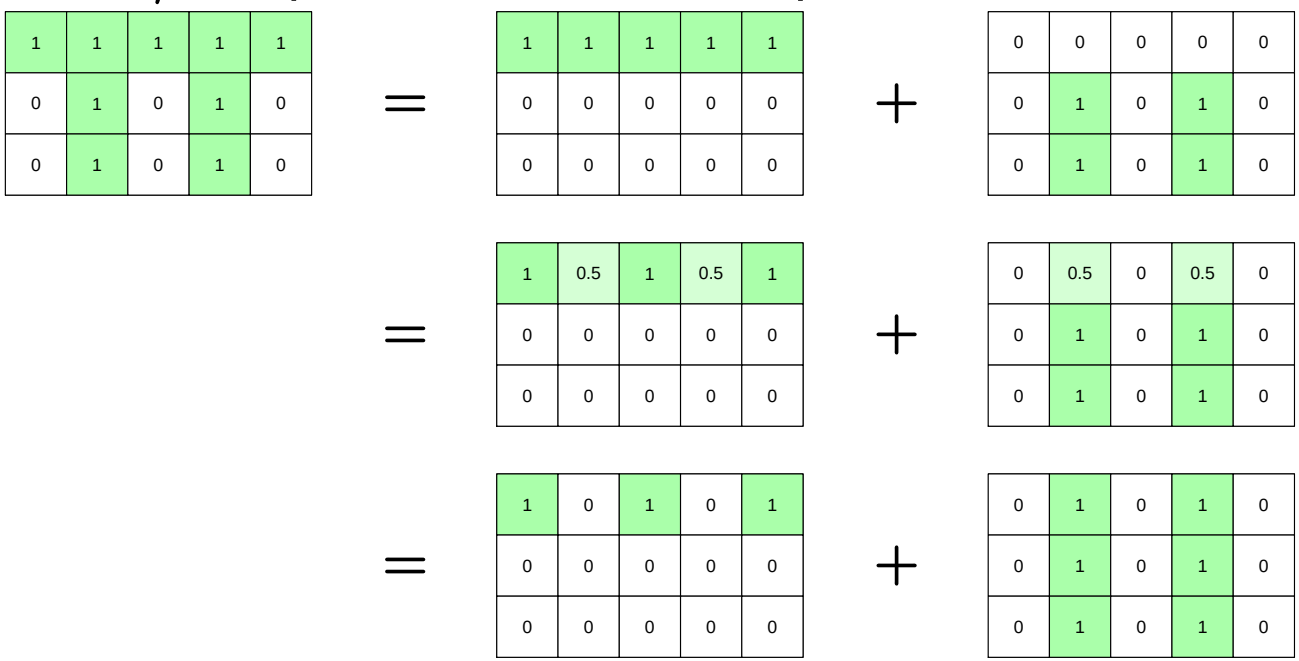
Lee and Seung, 1999.

NMF is not unique

- Factors are not "ordered"



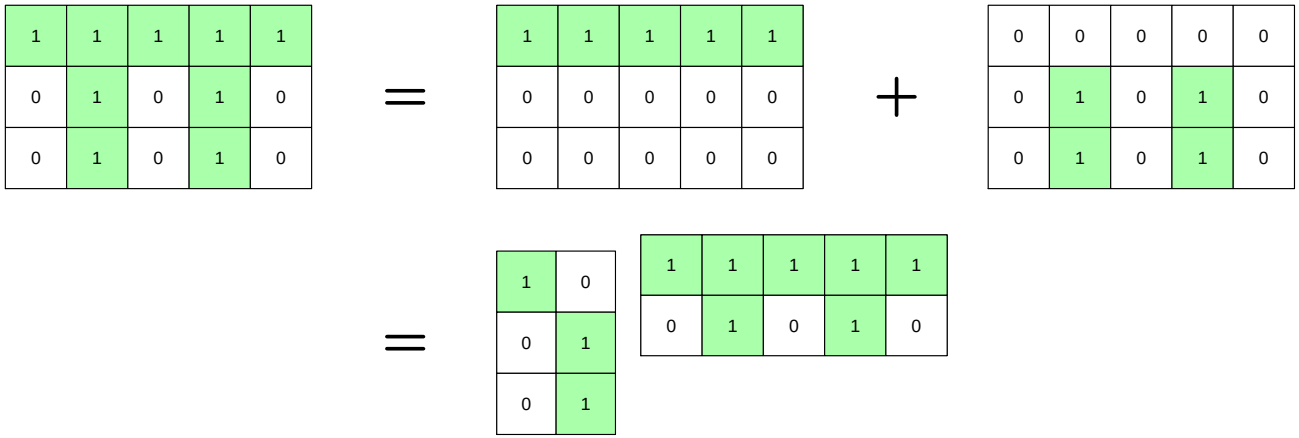
- One way of ordering: decreasing Frobenius norm of components (i.e., order by $\|\boldsymbol{l}_k \boldsymbol{r}_k^T\|_F = \|\boldsymbol{l}_k\| \, \|\boldsymbol{r}_k\|$)
- Factors/components are not unique



Additional constraints or regularization can encourage uniqueness.

NMF is not hierarchical

- Rank-2 NMF



- Rank-1 NMF

- Best rank-k approximation may differ significantly from best rank-$(k-1)$ approximation
- Rank influences sparsity & interpretability
- Optimum choice of rank is not well-studied (often requires experimentation)

Outline

NMF is difficult

We focus on minimizing $L(\boldsymbol{L}, \boldsymbol{R}) = \|\boldsymbol{D} - \boldsymbol{L}\boldsymbol{R}\|_F^2$.

- For varying m, n, and r, problem is NP-hard
- When $\text{rank}(\boldsymbol{D}) = 1$ (or $r = 1$), can be solved in polynomial time
  1. Take first non-zero column of $\boldsymbol{D}$ as $\boldsymbol{L}_{m\times 1}$
  2. Determine $\boldsymbol{R}_{1\times n}$ entry by entry (using the fact that $\boldsymbol{d}_j = \boldsymbol{L} r_{1j}$)
- Problem is not convex
  - Local optimum may not correspond to global optimum
  - Generally little hope to find global optimum
- But: Problem is biconvex
  - For fixed $\boldsymbol{R}$, $f(\boldsymbol{L}) = \|\boldsymbol{D} - \boldsymbol{L}\boldsymbol{R}\|_F^2$ is convex

$$
\begin{aligned}
f(\boldsymbol{L}) &= \textstyle\sum_i \|\boldsymbol{d}_i^T - \boldsymbol{l}_i^T \boldsymbol{R}\|_F^2 && \text{(chain rule)} \\
\nabla_{l_{ik}} f(\boldsymbol{L}) &= -2(\boldsymbol{d}_i^T - \boldsymbol{l}_i^T \boldsymbol{R})\boldsymbol{r}_k && \text{(product rule)} \\
\nabla^2_{l_{ik}} f(\boldsymbol{L}) &= 2\boldsymbol{r}_k^T \boldsymbol{r}_k \geq 0 && \text{(convex in } l_{ik}\text{; can show: also in } \boldsymbol{L}\text{)}
\end{aligned}
$$

  - For fixed $\boldsymbol{L}$, $f(\boldsymbol{R}) = \|\boldsymbol{D} - \boldsymbol{L}\boldsymbol{R}\|_F^2$ is convex
  - Allows for efficient algorithms

General framework

- Gradient descent generally slow
- Stochastic gradient descent often also slow
- Key approach: **alternating minimization**

  1: Pick starting point $\boldsymbol{L}_0$ and $\boldsymbol{R}_0$
  2: **while** not converged
  3: Keep $\boldsymbol{R}$ fixed, optimize $\boldsymbol{L}$
  4: Keep $\boldsymbol{L}$ fixed, optimize $\boldsymbol{R}$

- Update steps 3 and 4 easier than full problem
- Also called *alternating projections* or *(block) coordinate descent*
- Starting point
  - Random
  - Multi-start initialization: try multiple random starting points, run a few epochs, continue with best
  - Based on SVD
  - ...

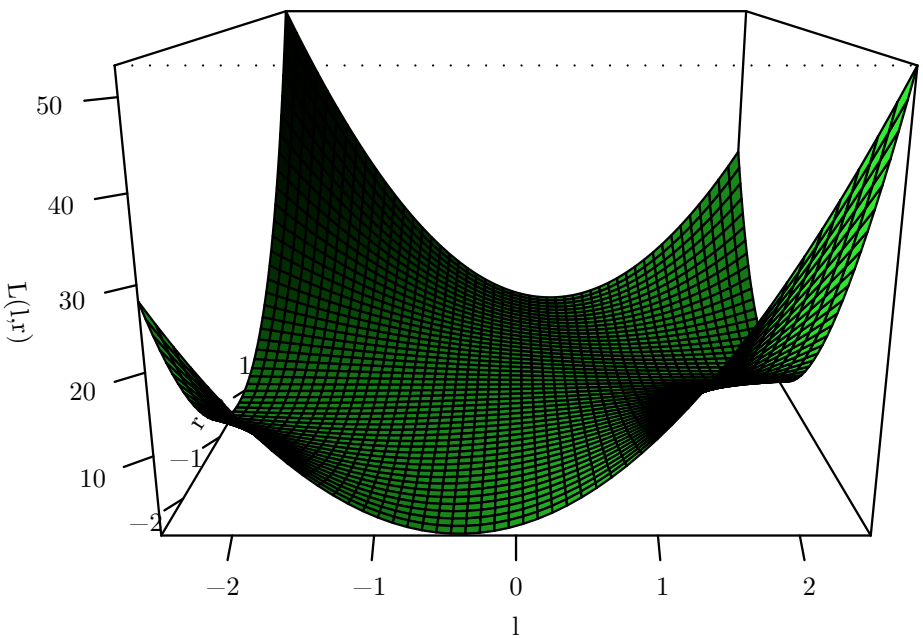
Example

Ignore non-negativity for now. Consider the regularized least-square error:

$$L(\boldsymbol{L}, \boldsymbol{R}) = \|\boldsymbol{D} - \boldsymbol{L}\boldsymbol{R}\|_F^2 + \frac{\lambda}{2}(\|\boldsymbol{L}\|_F^2 + \|\boldsymbol{R}\|_F^2)$$

By setting $m = n = r = 1$, $\boldsymbol{D} = (1)$ and $\lambda = 0.1$, we obtain

$$L(l, r) = (1 - lr)^2 + 0.05(l^2 + r^2)$$



$$\nabla_l L = -2r(1 - lr) + 0.1l$$
$$\nabla_r L = -2l(1 - lr) + 0.1r$$

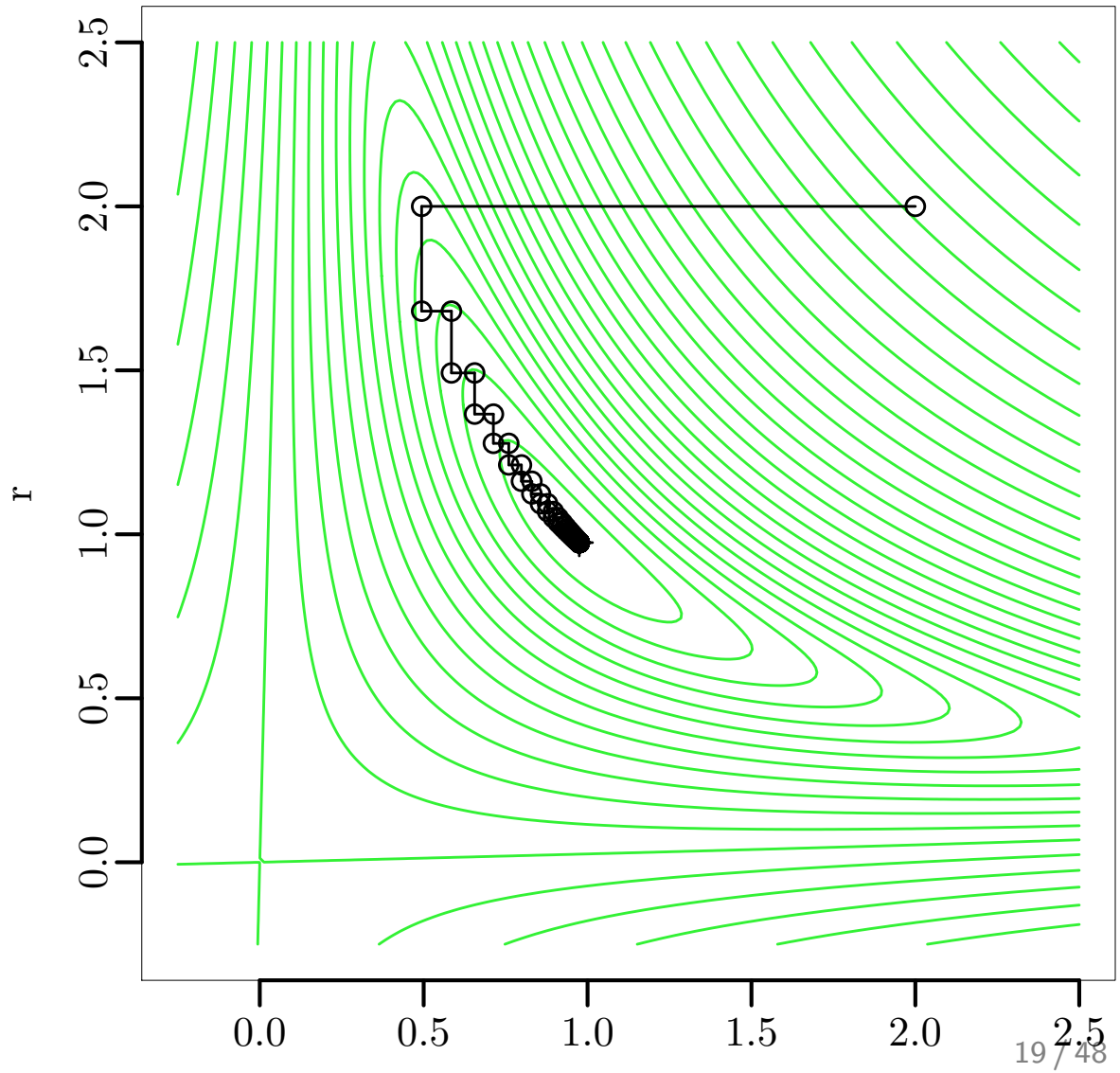
Local optima:

$$\left(\sqrt{\frac{19}{20}}, \sqrt{\frac{19}{20}}\right), \left(-\sqrt{\frac{19}{20}}, -\sqrt{\frac{19}{20}}\right)$$

Stationary point: (0,0)

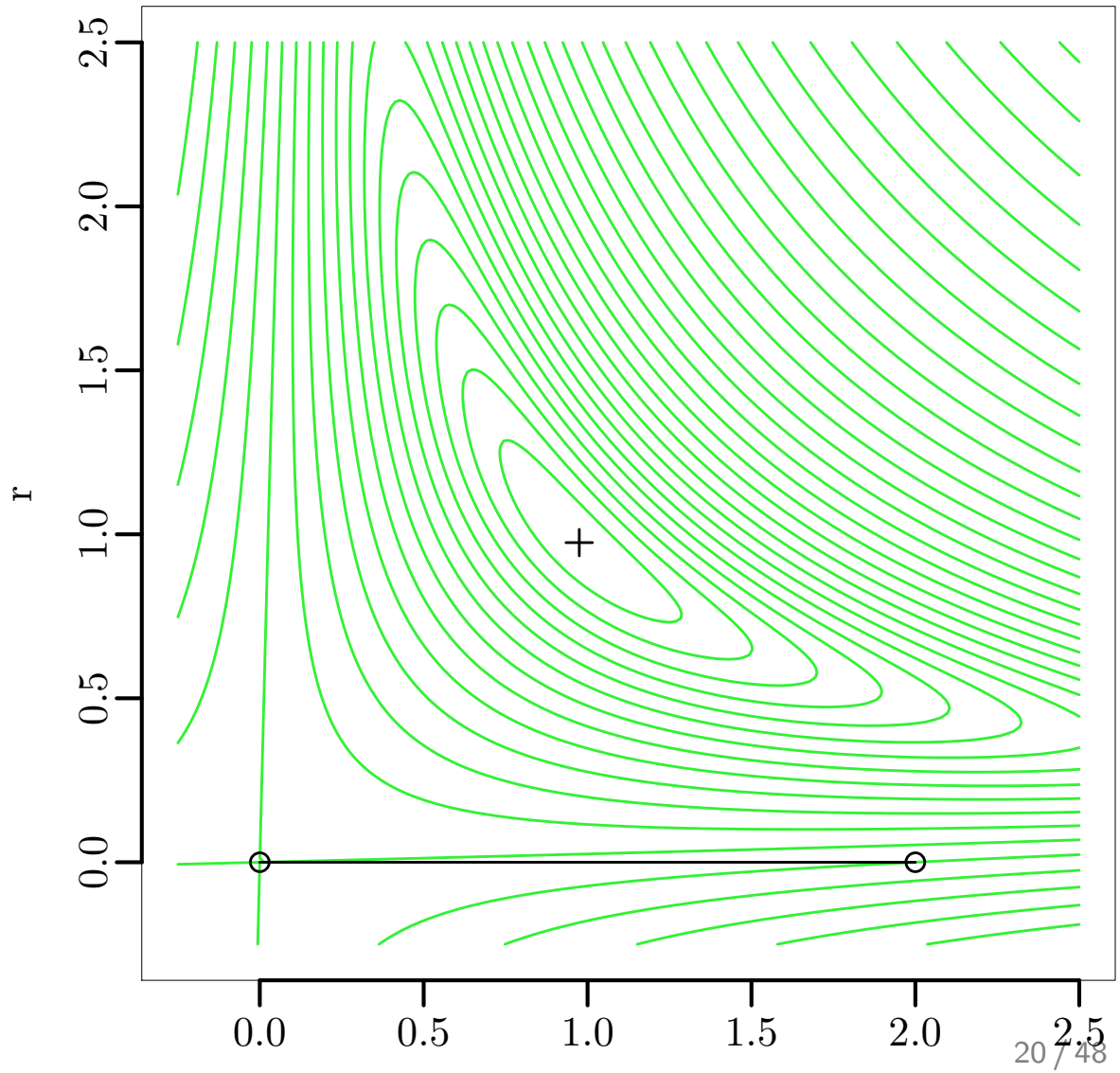
Example (Alternating Least Squares, ALS)

- $L(l, r) = (1 - lr)^2 + 0.05(l^2 + r^2)$
- $l \leftarrow \min_l L = \frac{r}{r^2+0.05}$
- $r \leftarrow \min_r L = \frac{l}{l^2+0.05}$
Step	l	r
0	2	2
1	0.49	2
2	0.49	1.68
3	0.58	1.68
4	0.58	1.49
⋮	⋮	⋮
100	0.97	0.97
- Converges to local minimum



Example (Alternating Least Squares, ALS)

- $L(l, r) = (1 - lr)^2 + 0.05(l^2 + r^2)$
- $l \leftarrow \min_l L = \frac{r}{r^2+0.05}$
- $r \leftarrow \min_r L = \frac{l}{l^2+0.05}$
Step	l	r
0	2	0
1	0	0
2	0	0
⋮	⋮	⋮
- Converges to stationary point



Alternating non-negative least squares (ANLS)

- Uses non-negative least squares approximation of $\boldsymbol{L}$ and $\boldsymbol{R}$:

$$\underset{\boldsymbol{L} \in \mathbb{R}_+^{m \times r}}{\operatorname{argmin}} \|\boldsymbol{D} - \boldsymbol{L}\boldsymbol{R}\|_F^2 \qquad \text{and} \qquad \underset{\boldsymbol{R} \in \mathbb{R}_+^{r \times n}}{\operatorname{argmin}} \|\boldsymbol{D} - \boldsymbol{L}\boldsymbol{R}\|_F^2$$

- Equivalently: find non-negative least squares solution to $\boldsymbol{L}\boldsymbol{R} = \boldsymbol{D}$
- Common (bad!) hack: Solve unconstrained least squares problems and "remove" negative values. E.g., when columns (rows) of $\boldsymbol{L}$ ($\boldsymbol{R}$) are linearly independent, set

$$\boldsymbol{L} = [\boldsymbol{D}\boldsymbol{R}^+]_\epsilon \qquad \text{and} \qquad \boldsymbol{R} = [\boldsymbol{L}^+\boldsymbol{D}]_\epsilon$$

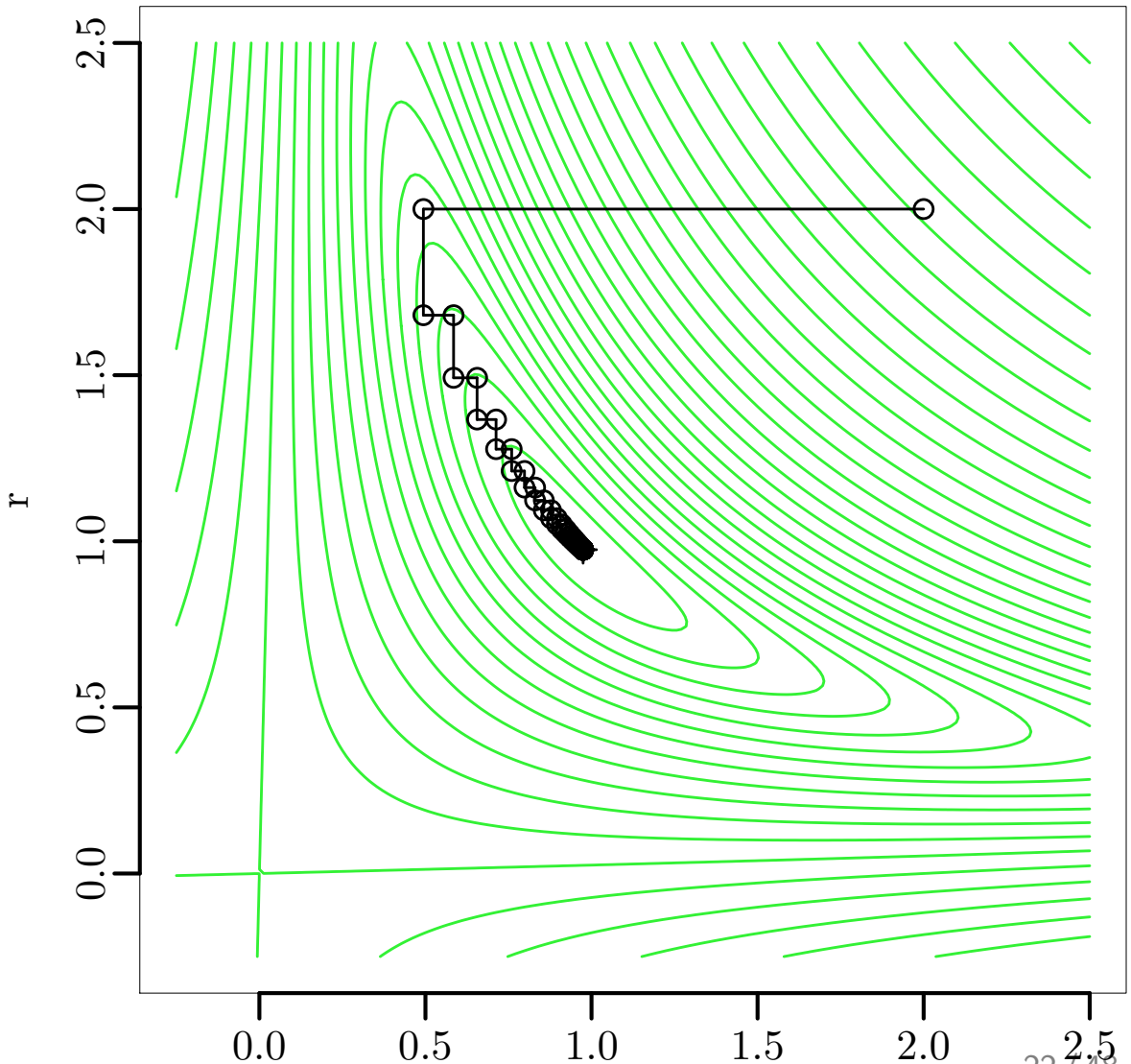
  where
  - $\boldsymbol{R}^+ = \boldsymbol{R}^T(\boldsymbol{R}\boldsymbol{R}^T)^{-1}$ is the pseudo-inverse of $\boldsymbol{R}$
  - $\boldsymbol{L}^+ = (\boldsymbol{L}^T\boldsymbol{L})^{-1}\boldsymbol{L}^T$ is the pseudo-inverse of $\boldsymbol{L}$
  - $[a]_\epsilon = \max\{\epsilon, a\}$ for $\epsilon = 0$ or some small constant (e.g., $\epsilon = 10^{-9}$)
- Difficult to analyze due to non-linear update steps
- Often slow convergence to a "bad" local minimum (better when regularized or good NLS solver used)

Example (ANLS, common bad hack)

- $f(l, r) = (1 - lr)^2 + 0.05(l^2 + r^2)$ and set $\epsilon = 10^{-9}$
- $l \leftarrow \left[\frac{r}{r^2+0.05}\right]_\epsilon$
- $r \leftarrow \left[\frac{l}{l^2+0.05}\right]_\epsilon$
-

Step	l	r
0	2	2
1	0.49	2
2	0.49	1.68
3	0.58	1.68
4	0.58	1.49
⋮	⋮	⋮
100	0.97	0.97

- Converges to local minimum

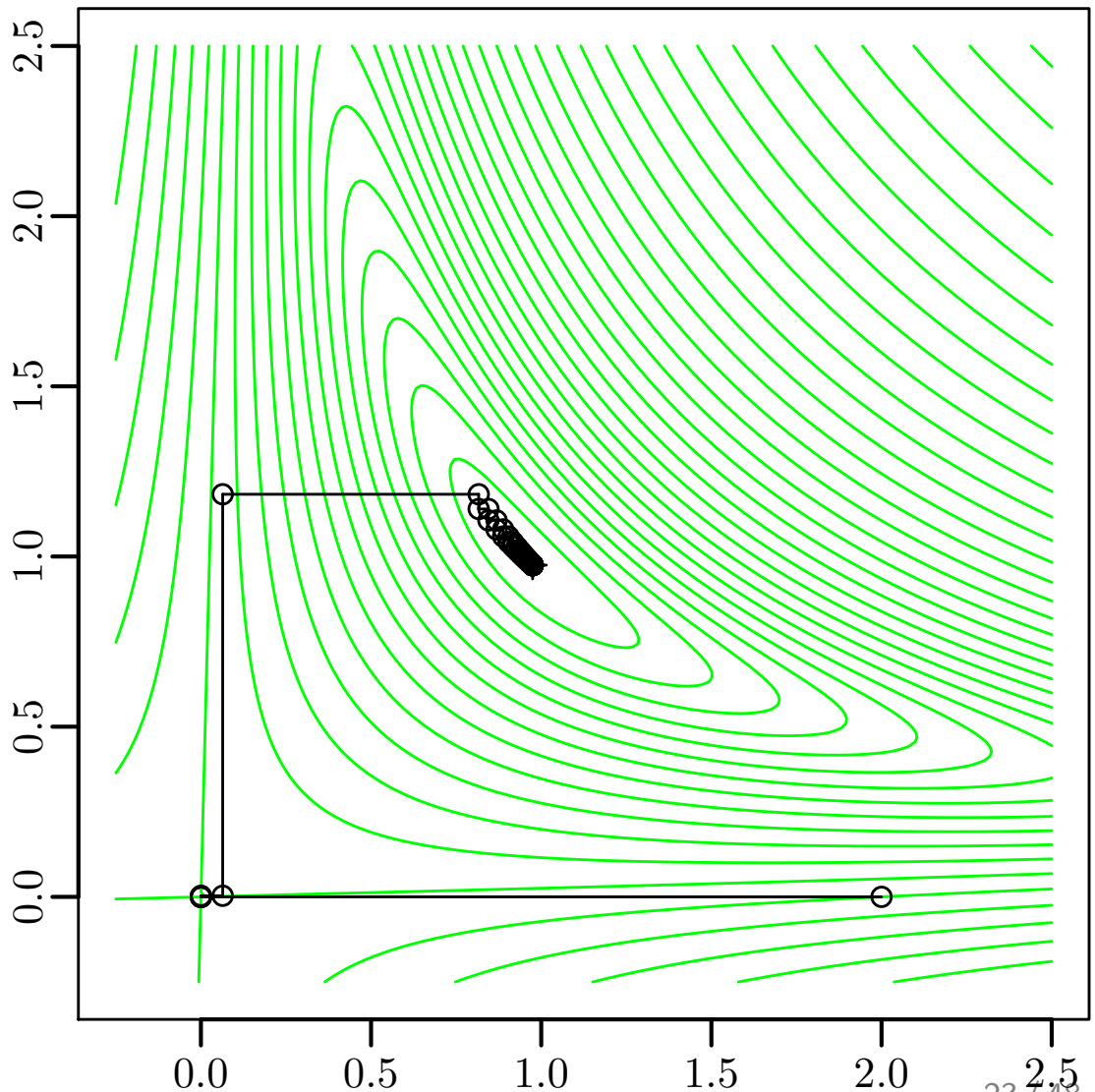


Example (ANLS, common bad hack)

- $L(l, r) = (1 - lr)^2 + 0.05(l^2 + r^2)$ and set $\epsilon = 10^{-9}$
- $l \leftarrow \left[\frac{r}{r^2+0.05}\right]_\epsilon$
- $r \leftarrow \left[\frac{l}{l^2+0.05}\right]_\epsilon$
-

Step	l	r
0	2	0
1	$1 \cdot 10^{-9}$	0
2	$1 \cdot 10^{-9}$	$2 \cdot 10^{-8}$
3	$4 \cdot 10^{-7}$	$2 \cdot 10^{-8}$
4	$4 \cdot 10^{-7}$	$8 \cdot 10^{-6}$
$\vdots$	$\vdots$	$\vdots$
100	0.97	0.97

- Converges to local minimum



Hierarchical alternating least squares (HALS)

- Optimize just one component, then next component, and so on
- Let $\boldsymbol{D}^{(k)}$ be the residual matrix (error) when k-th component is removed:

$$\boldsymbol{D}^{(k)} = \boldsymbol{D} - \boldsymbol{L}\boldsymbol{R} + \boldsymbol{l}_k \boldsymbol{r}_k^T = \boldsymbol{D} - \sum_{k' \neq k} \boldsymbol{l}_{k'} \boldsymbol{r}_{k'}^T$$

- HALS optimizes $\|\boldsymbol{D}^{(k)} - \boldsymbol{l}_k \boldsymbol{r}_k^T\|_F^2$ for $k = 1, 2, \ldots, r, 1, \ldots$ (thus: optimize only k-th component)
- E.g., in each iteration, set (once or multiple times):

$$\boldsymbol{l}_k^T \leftarrow \frac{1}{\|\boldsymbol{r}_k\|_F^2} \left[\boldsymbol{D}^{(k)} \boldsymbol{r}_k\right]_\epsilon \quad \text{and} \quad \boldsymbol{r}_k^T \leftarrow \frac{1}{\|\boldsymbol{l}_k\|_F^2} \left[\boldsymbol{l}_k^T \boldsymbol{D}^{(k)}\right]_\epsilon$$

- $\boldsymbol{D}^{(k)}$ can be incrementally maintained $\rightarrow$ fast implementation

$$\boldsymbol{D}^{(k+1)} = \boldsymbol{D}^{(k)} + \boldsymbol{l}_k \boldsymbol{r}_k^T - \boldsymbol{l}_{k+1} \boldsymbol{r}_{k+1}^T$$

- Good performance in practice
- Converges to stationary point when initialized with positive matrix and sufficiently small ϵ

Multiplicative updates

- Gradient descent step with step size η_{kj}

$$r_{kj} \leftarrow r_{kj} + \eta_{kj}([\boldsymbol{L}^T\boldsymbol{D}]_{kj} - [\boldsymbol{L}^T\boldsymbol{L}\boldsymbol{R}]_{kj})$$

- Setting $\eta_{kj} = \frac{r_{kj}}{[\boldsymbol{L}^T\boldsymbol{L}\boldsymbol{R}]_{kj}}$, we obtain the multiplicative update rules

$$\boldsymbol{L} \leftarrow \boldsymbol{L} \circ \frac{\boldsymbol{D}\boldsymbol{R}^T}{\boldsymbol{L}\boldsymbol{R}\boldsymbol{R}^T} \quad \text{and} \quad \boldsymbol{R} \leftarrow \boldsymbol{R} \circ \frac{\boldsymbol{L}^T\boldsymbol{D}}{\boldsymbol{L}^T\boldsymbol{L}\boldsymbol{R}},$$

  where multiplication ($\circ$) and division are element-wise
- Does not necessarily find optimum $\boldsymbol{L}$ (or $\boldsymbol{R}$), but can be shown to never increase loss
- Faster than ANLS (no NLS problems solved), easy to implement and parallelize
- Zeros in factors are problematic (never changed, division by 0)

$$\boldsymbol{L} \leftarrow \boldsymbol{L} \circ \frac{[\boldsymbol{D}\boldsymbol{R}^T]_\epsilon}{\boldsymbol{L}\boldsymbol{R}\boldsymbol{R}^T + \epsilon} \quad \text{and} \quad \boldsymbol{R} \leftarrow \boldsymbol{R} \circ \frac{[\boldsymbol{L}^T\boldsymbol{D}]_\epsilon}{\boldsymbol{L}^T\boldsymbol{L}\boldsymbol{R} + \epsilon}$$

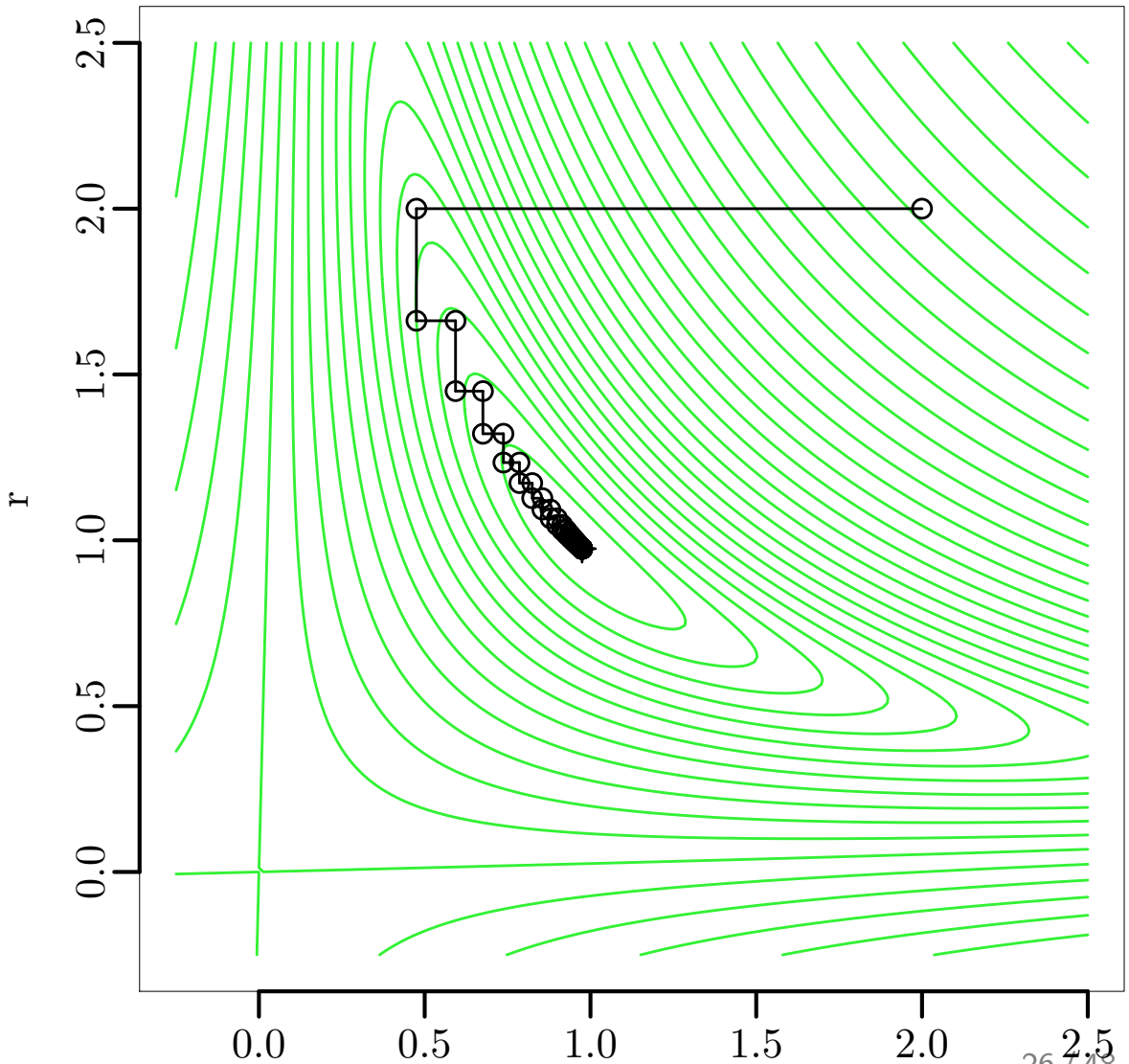
Lee and Seung, 2001. Cichocki et al., 2006.

Example (multiplicative updates)

- $L(l, r) = (1 - lr)^2 + 0.05(l^2 + r^2)$
- $l \leftarrow l \frac{[1r - 0.05l]_\epsilon}{lr^2 + \epsilon}$
- $r \leftarrow r \frac{[l1 - 0.05r]_\epsilon}{l^2 r + \epsilon}$
-

Step	l	r
0	2	2
1	0.48	2
2	0.48	1.66
3	0.59	1.66
4	0.58	1.45
⋮	⋮	⋮
100	0.97	0.97

- Converges to local minimum



Outline

Topic modeling

- Consider a document-word matrix constructed from some corpus

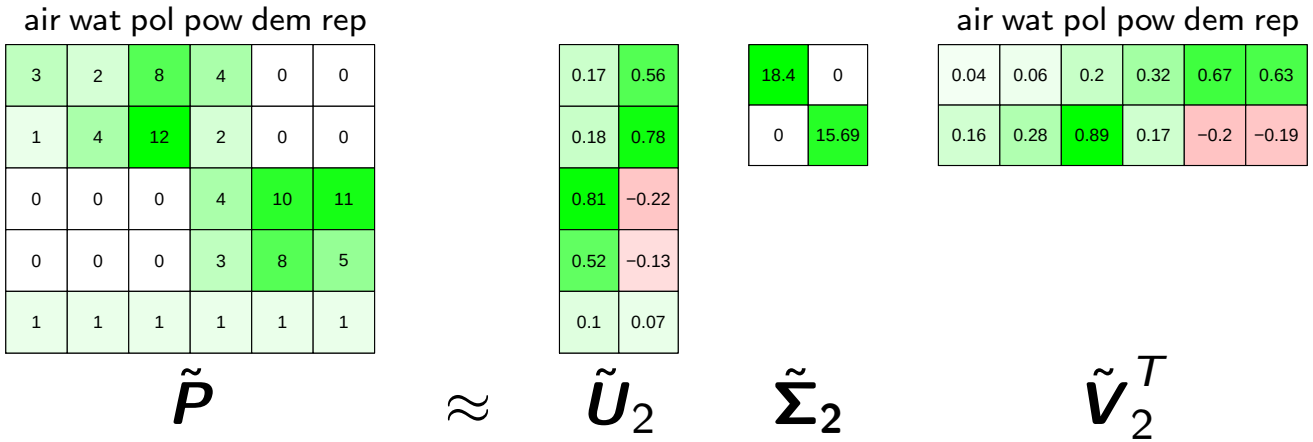
$$\tilde{\boldsymbol{P}} = \begin{array}{c} \\ \text{doc 1} \\ \text{doc 2} \\ \text{doc 3} \\ \text{doc 4} \\ \text{doc 5} \end{array} \begin{array}{c} \begin{array}{cccccc} \text{air} & \text{water} & \text{pollution} & \text{power} & \text{democrat} & \text{republican} \end{array} \\ \begin{pmatrix} 3 & 2 & 8 & 4 & 0 & 0 \\ 1 & 4 & 12 & 2 & 0 & 0 \\ 0 & 0 & 0 & 4 & 10 & 11 \\ 0 & 0 & 0 & 3 & 8 & 5 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \end{array}$$

- Documents seem to talk about two "topics"
  1. Environment (with words air, water, pollution, power)
  2. Congress (with words democrat, republican, power)
- Note: "power" is polysemous (electrical power, political power)

Can we automatically detect topics in documents?

Let's try SVD

- Rank-2 truncated SVD of example matrix

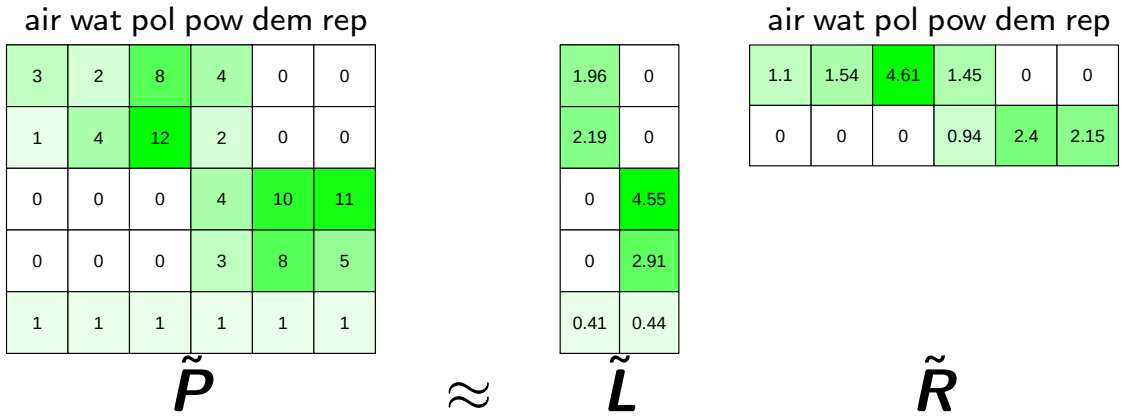


- Can see clear topic-driven differences for documents in $\tilde{u}_1$ and $\tilde{u}_2$
- Similarly for words in $\tilde{v}_1$ and $\tilde{v}_2$
- Hard to interpret
- Called **latent semantic analysis**

Landauer and Dumais, 1997

Let's try NMF

- Rank-2 NMF of example matrix



- Can see clear topic-driven differences in $\tilde{L}$ and $\tilde{R}$
- Easier to interpret: large value $\rightarrow$ relevant for topic
- Decomposition is sparse
- Related to **probabilistic latent semantic analysis** (pLSA)
  - An even more interpretable factorization
  - Assumptions are well-understood
  - Example of a **topic model**
- Polysemy of "power" hinted at by its use in multiple topics
- Topic distributions often much more complex

A probabilistic viewpoint

- Let's normalize $\tilde{\boldsymbol{P}}$ such that the entries sum to unity

air wat pol pow dem rep

3	2	8	4	0	0
1	4	12	2	0	0
0	0	0	4	10	11
0	0	0	3	8	5
1	1	1	1	1	1

$\tilde{\boldsymbol{P}}$

air wat pol pow dem rep

0.04	0.02	0.1	0.05	0	0
0.01	0.05	0.14	0.02	0	0
0	0	0	0.05	0.12	0.13
0	0	0	0.04	0.1	0.06
0.01	0.01	0.01	0.01	0.01	0.01

$\boldsymbol{P}$

- Put all words in an urn, draw, and denote by D and W the result. The probability to draw word w from document d is given by

$$P(D = d, W = w) = p_{dw}$$

- Matrix $\boldsymbol{P}$ can represent *any* such probability distribution
- pLSA tries to find a distribution that is "close" to $\boldsymbol{P}$ but exposes information about topics

Probabilities: Shortcut notation

Let X and Y be discrete random variables with possible values $\mathrm{Val}(X)$ and $\mathrm{Val}(Y)$. Let $x \in \mathrm{Val}(X)$ and $y \in \mathrm{Val}(Y)$.

Expression	Shortcut notation
$P(X = x)$	$P(x)$
$P(X = x, Y = y)$	$P(x, y)$
$P(X = x \mid Y = y)$	$P(x \mid y)$
$\forall x. P(X = x) = f(x)$	$P(X) = f(X)$
$\forall x. \forall y. P(X = x \mid Y = y) = f(x, y)$	$P(X \mid Y) = f(X, Y)$

- $P(x)$, $P(x, y)$, $P(x \mid y)$ are numbers (= probabilities)
- $P(X)$ and $P(X \mid Y)$ are functions (= probability distributions)
- Can be thought of as functions from $\mathrm{Val}(X) \to [0, 1]$ or $\mathrm{Val}(X) \times \mathrm{Val}(Y) \to [0, 1]$, respectively
- $f_y(X) = P(X \mid y)$ is often referred to as **conditional probability distribution** (CPD)

Probabilities: Important properties

$$P(A \cup B) = P(A) + P(B) - P(A \cap B) \qquad \text{(inclusion-exclusion)}$$

$$P(\bar{A}) = 1 - P(A)$$

$$\text{If } B \supseteq A, \quad P(B) = P(A) + P(B \setminus A) \geq P(A)$$

$$P(X, Y) = P(Y \mid X)P(X) \qquad \text{(product rule)}$$

$$P(X) = \sum_y P(X, y) \qquad \text{(sum rule, discrete)}$$

$$P(X) = \int_y P(X, y) \qquad \text{(sum rule, continuous)}$$

$$P(X \mid Y) = \frac{P(Y \mid X)P(X)}{P(Y)} \qquad \text{(Bayes theorem)}$$

$$E[aX + b] = aE[X] + b \qquad \text{(linearity of expectation)}$$
$$E[X + Y] = E[X] + E[Y]$$

$$E_Y[E_X[X \mid Y]] = E[X] \qquad \text{(law of total expectation)}$$

Probabilistic latent semantic analysis (pLSA)

Definition (pLSA, NMF formulation)

Given a rank r, find matrices $\boldsymbol{L}$, $\boldsymbol{\Sigma}$, and $\boldsymbol{R}$ such that

$$\boldsymbol{P} \approx \boldsymbol{L}\boldsymbol{\Sigma}\boldsymbol{R}$$

where

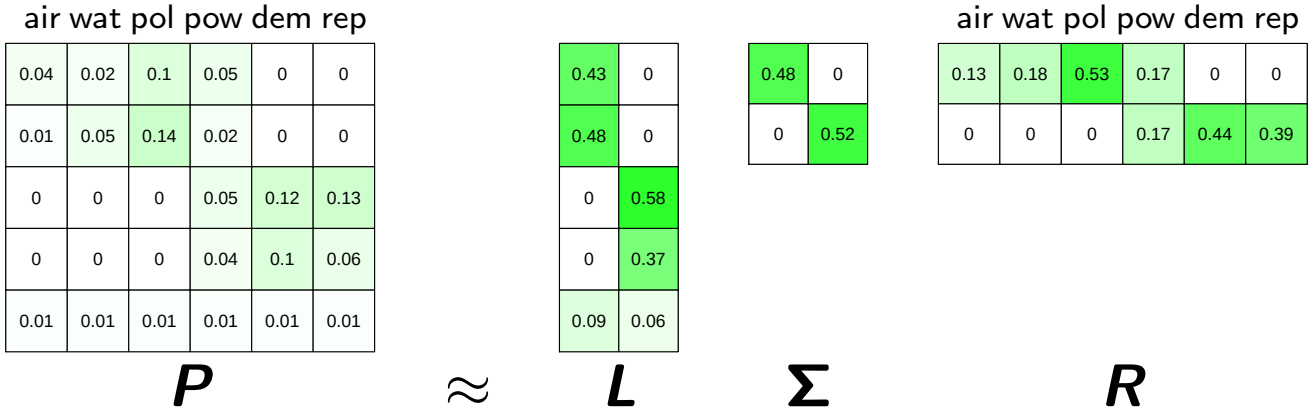
- $\boldsymbol{L}_{m \times r}$ is a column-stochastic matrix,
- $\boldsymbol{\Sigma}_{r \times r}$ is a non-negative, diagonal matrix that sums to unity, and
- $\boldsymbol{R}_{r \times n}$ is a row-stochastic matrix.

- **Column-stochastic** = each column is probability vector = each column is non-negative and sums to 1
- $\approx$ is usually taken to be the (generalized) KL divergence
- Regularization or "tempering" may be necessary to avoid overfitting

Hofmann, 2001.

Example

- pLSA factorization of example matrix



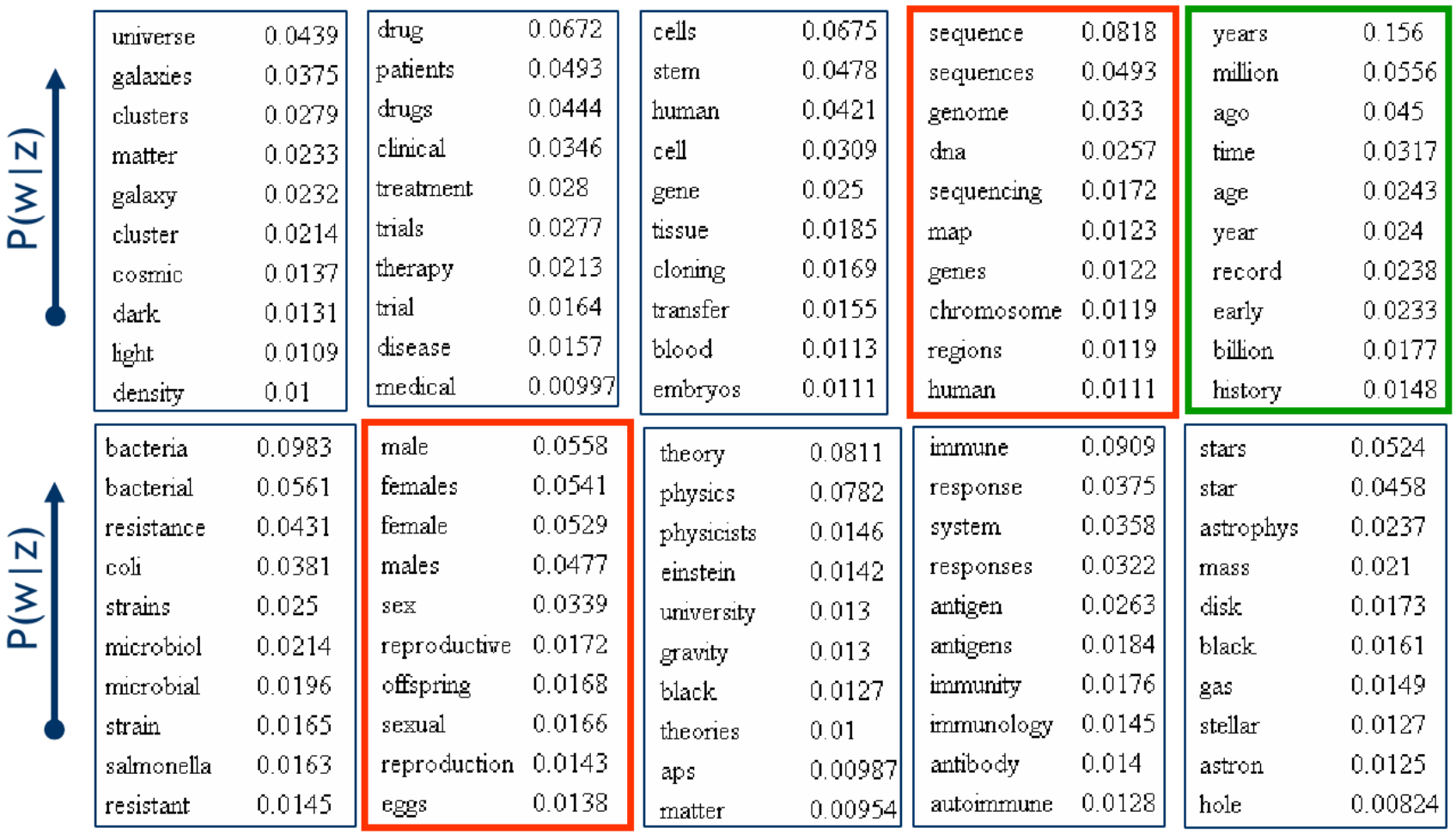
- Rank r corresponds to number of topics
- σ_z corresponds to overall relative frequency of topic z
- l_{dz} corresponds to contribution of document d to topic z
- r_{zw} corresponds to frequency of word w in topic z
- pLSA constraints allow for probabilistic interpretation

$$P(d, w) \approx [\boldsymbol{L\Sigma R}]_{dw} = \sum_z \sigma_z l_{dz} r_{zw} = \sum_z P(z)P(d \mid z)P(w \mid z)$$

- pLSA model assumes conditional independence, i.e., it assumes that words and documents are conditionally independent given a topic $\rightarrow$ restricted space of distributions

Another example

Concepts (10 of 128) extracted from Science Magazine articles (12K)



"Topics" (also: concepts, aspects) described by distributions of words.

Hofmann, 2004.

pLSA geometry

- Rewrite probabilistic formulation

$$P(d, w) = \sum_z P(z)P(d \mid z)P(w \mid z)$$

$$P(w \mid d) = \sum_z P(w \mid z)P(z \mid d)$$

- Generative process
  1. Pick a document according to $P(d)$
  2. Select a topic acc. to $P(z \mid d)$
  3. Select a word acc. to $P(w \mid z)$

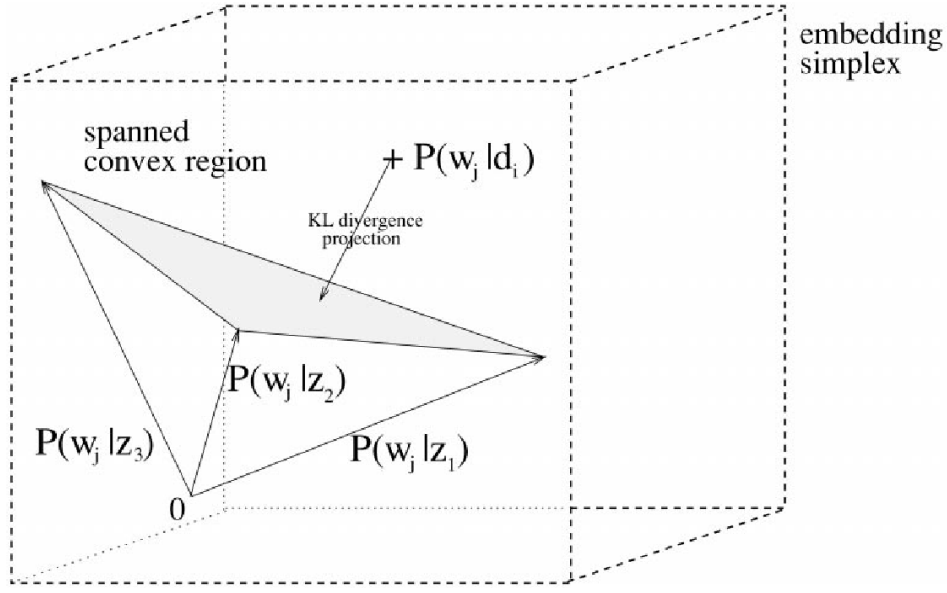
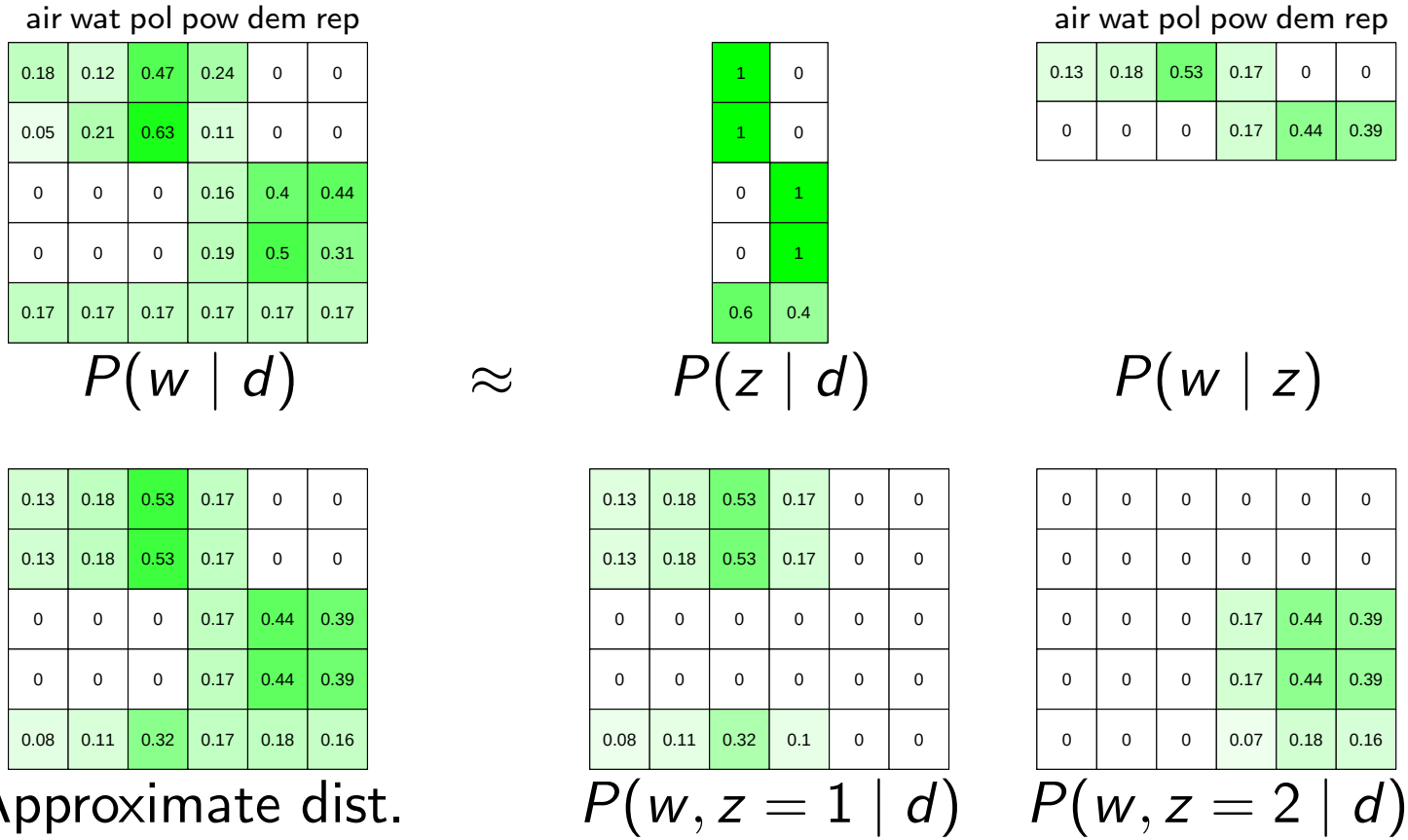


Figure 2. Sketch of the probability simplex and a convex region spanned by class-conditional probabilities in the aspect model.

Hofmann, 2001.

Example

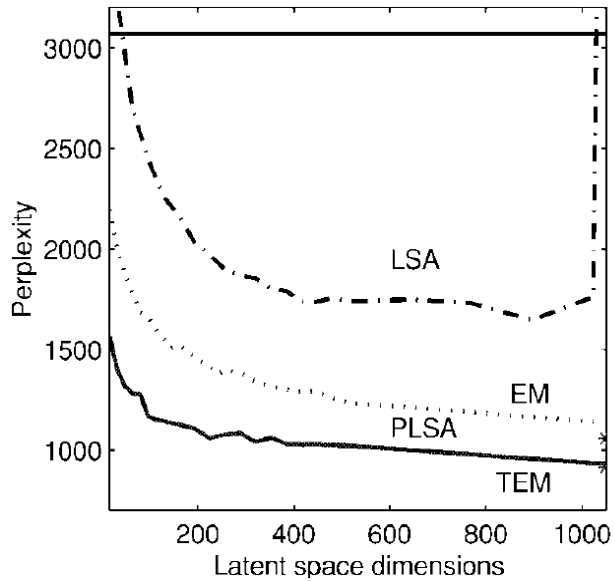
- pLSA factorization of example matrix, rewritten formulation



- This formulation more directly exposes topic distribution of documents (more in assignment)
- Next: Applications & a suitable definition of "≈"

Applications of pLSA

- Topic modeling
- Clustering documents
- Clustering terms
- Information retrieval
  - Treat query q as a "new" document
  - Using the second formulation of the previous slide, determine $P(z \mid q)$, keeping $P(w \mid z)$ fixed (called "fold in" the query)
  - In more detail, find $P(z \mid q)$ such that $P(w \mid q) \approx \sum_z P(w \mid z)P(z \mid q)$
  - Retrieve documents with similar topic mixture ($= P(z \mid d)$)
  - Can deal with synonymy and polysemy
- Better generalization performance than LSA (=SVD), esp. with tempering
- Full Bayesian treatment: Latent Dirichlet Allocation (LDA)



Hofmann, 2001.

Kullback-Leibler divergence (1)

- Let $\tilde{\boldsymbol{P}}$ be the unnormalized word-count data and denote by N total number of words
- Probability of seeing $\tilde{\boldsymbol{P}}$ when drawing N words with replacement

$$P(\tilde{\boldsymbol{P}}) \propto \prod_{d=1}^{m} \prod_{w=1}^{n} P(d,w)^{\tilde{p}_{dw}}$$

- Conditional probability $P(\tilde{\boldsymbol{P}} \mid \boldsymbol{L}, \boldsymbol{\Sigma}, \boldsymbol{R})$ of seeing the data given a pLSA model is as above with $P(d,w)$ given by

$$P(d,w \mid \boldsymbol{L}, \boldsymbol{\Sigma}, \boldsymbol{R}) = [\boldsymbol{L}\boldsymbol{\Sigma}\boldsymbol{R}]_{dw}$$

- pLSA maximizes this probability = **likelihood** $\mathcal{L}(\tilde{\boldsymbol{P}} \mid \boldsymbol{L}, \boldsymbol{\Sigma}, \boldsymbol{R})$ of data given model
- Equivalent to maximizing the **log-likelihood** $\log \mathcal{L}(\tilde{\boldsymbol{P}} \mid \boldsymbol{L}, \boldsymbol{\Sigma}, \boldsymbol{R})$

Kullback-Leibler divergence (2)

Let $\hat{\boldsymbol{P}} = \boldsymbol{L}\boldsymbol{\Sigma}\boldsymbol{R}$.

$$\log \mathcal{L}(\tilde{\boldsymbol{P}} \mid \hat{\boldsymbol{P}}) \propto \sum_{d=1}^{m}\sum_{w=1}^{n} \tilde{p}_{dw} \log \hat{p}_{dw}$$

$$\propto \sum_{d=1}^{m}\sum_{w=1}^{n} p_{dw} \log \hat{p}_{dw}$$

$$= \underbrace{-\sum_{d=1}^{m}\sum_{w=1}^{n} p_{dw} \log \frac{1}{\hat{p}_{dw}}}_{\text{Cross entropy } H(\boldsymbol{P}, \hat{\boldsymbol{P}})}$$

$$= \underbrace{-\sum_{d=1}^{m}\sum_{w=1}^{n} p_{dw} \log \frac{p_{dw}}{\hat{p}_{dw}}}_{\text{Kullback-Leibler divergence } D_{\mathsf{KL}}(\boldsymbol{P} \| \hat{\boldsymbol{P}})} + c_{\boldsymbol{P}}$$

- Takeaway: maximizing the log-likelihood $\equiv$ minimize cross entropy $\equiv$ minimizing the Kullback-Leibler divergence

Gaussier and Goutte, 2005.

Background: Entropy

- Consider any discrete contribution over values $\{ a_1, \ldots, a_n \}$
- We can represent this distribution using a probability vector $\boldsymbol{p} \in \mathbb{R}^n$, where value a_i is associated with probability p_i
- Suppose Anna selects a random value A according to $\boldsymbol{p}$, i.e., $P(A = a_i) = p_i$
- She wants to tell Bob which value she selected
  - Anna and Bob agree upfront on a codeword (bitstring) for each value a_i
  - On average, how many bits does Anna have to send to Bob so that he can determine the value of A?
- Answer: at best $H(\boldsymbol{p})$ bits, where $H(\boldsymbol{p})$ is the **Shannon entropy**

$$H(\boldsymbol{p}) = \sum_{i=1}^{n} p_i \log \frac{1}{p_i}$$

  - p_i = probability that Anna selects value a_i
  - $\log \frac{1}{p_i}$ = number of bits of codeword for a_i

Background: Cross entropy and KL divergence

- Let's modify the game a bit
  - Suppose Anna cheats and gives Bob the wrong distribution $\boldsymbol{q}$
  - They use an optimal code for $\boldsymbol{q}$, i.e., codeword for a_i has $\log \frac{1}{q_i}$ bits
  - But Anna later selects values according to $\boldsymbol{p}$, not $\boldsymbol{q}$
  - How many bits are sent on average?
- Answer: **cross entropy** of $\boldsymbol{p}$ and $\boldsymbol{q}$

$$H(\boldsymbol{p}, \boldsymbol{q}) = \sum_{i=1}^{n} p_i \log \frac{1}{q_i}$$

  - p_i = probability that Anna selects value a_i
  - $\log \frac{1}{q_i}$ = number of bits of codeword for a_i
- On average, how many additional bits are now sent?
- Answer: **Kullback-Leibler divergence**

$$D_{\mathrm{KL}}(\boldsymbol{p} \| \boldsymbol{q}) \stackrel{\mathrm{def}}{=} H(\boldsymbol{p}, \boldsymbol{q}) - H(\boldsymbol{p}) = \sum_{i=1}^{n} p_i \log \frac{p_i}{q_i}$$

Kullback-Leibler divergence (3)

- KL divergence is a measure of "difference" between distributions
- In our setting, we have

$$D_{\mathrm{KL}}(\boldsymbol{P}\|\hat{\boldsymbol{P}}) = \sum_{d,w} p_{dw} \log \frac{p_{dw}}{\hat{p}_{dw}}$$

- Interpretation: expected number of extra bits for encoding a value drawn from $\boldsymbol{P}$ using an optimum code for distribution $\hat{\boldsymbol{P}}$
- $D_{\mathrm{KL}}(\boldsymbol{P}\|\hat{\boldsymbol{P}}) \geq 0$
- $D_{\mathrm{KL}}(\boldsymbol{P}\|\boldsymbol{P}) = 0$
- In general, $D_{\mathrm{KL}}(\boldsymbol{P}\|\hat{\boldsymbol{P}}) \neq D_{\mathrm{KL}}(\hat{\boldsymbol{P}}\|\boldsymbol{P})$
- NMF-based pLSA algorithms often minimize the **generalized KL divergence** (exercise)

$$D_{\mathrm{GKL}}(\tilde{\boldsymbol{P}}\|\tilde{\hat{\boldsymbol{P}}}) = \sum_{d,w} \left(\tilde{p}_{dw} \log \frac{\tilde{p}_{dw}}{\tilde{\hat{p}}_{dw}} - \tilde{p}_{dw} + \tilde{\hat{p}}_{dw} \right),$$

where $\tilde{\hat{\boldsymbol{P}}} = \tilde{\boldsymbol{L}}\tilde{\boldsymbol{R}}$

Multiplicative updates for GKL (w/o tempering)

- We first find a decomposition $\tilde{\boldsymbol{P}} \approx \tilde{\boldsymbol{L}}\tilde{\boldsymbol{R}}$, where $\tilde{\boldsymbol{L}}$ and $\tilde{\boldsymbol{R}}$ are non-negative matrices
  - E.g., using multiplicative update rules

$$\tilde{\boldsymbol{L}} \leftarrow \tilde{\boldsymbol{L}} \circ \frac{\tilde{\boldsymbol{P}}}{\tilde{\boldsymbol{L}}\tilde{\boldsymbol{R}}} \tilde{\boldsymbol{R}}^T \operatorname{diag}(\mathbf{1}_r / (\tilde{\boldsymbol{R}}\mathbf{1}_n))$$

$$\tilde{\boldsymbol{R}} \leftarrow \tilde{\boldsymbol{R}} \circ \operatorname{diag}(\mathbf{1}_r / (\mathbf{1}_m^T \tilde{\boldsymbol{L}})^T) \tilde{\boldsymbol{L}}^T \frac{\tilde{\boldsymbol{P}}}{\tilde{\boldsymbol{L}}\tilde{\boldsymbol{R}}}$$

  - GKL is non-increasing under these update rules
- Normalize by rescaling columns of $\tilde{\boldsymbol{L}}$ and rows of $\tilde{\boldsymbol{R}}$ to obtain

$$\boldsymbol{L} = \tilde{\boldsymbol{L}} \operatorname{diag}(\mathbf{1}_r / (\mathbf{1}_m^T \tilde{\boldsymbol{L}})^T) \qquad \text{(colums of } \boldsymbol{L} \text{ sum to 1)}$$

$$\boldsymbol{R} = \operatorname{diag}(\mathbf{1}_r / (\tilde{\boldsymbol{R}}\mathbf{1}_n)) \tilde{\boldsymbol{R}} \qquad \text{(rows of } \boldsymbol{R} \text{ sum to 1)}$$

$$\tilde{\boldsymbol{\Sigma}} = \operatorname{diag}((\mathbf{1}_m^T \tilde{\boldsymbol{L}})^T \circ (\tilde{\boldsymbol{R}}\mathbf{1}_n)) \qquad \text{(scaling factors)}$$

$$\boldsymbol{\Sigma} = \tilde{\boldsymbol{\Sigma}} / \textstyle\sum_z \tilde{\boldsymbol{\Sigma}}_{zz} \qquad (\boldsymbol{\Sigma} \text{ sums to 1})$$

Lee and Seung, 2001.

Outline

Lessons learned

- Non-negative matrix factorization (NMF) appears natural for non-negative data
- NMF encourages parts-based decomposition, interpretability, and (sometimes) sparseness
- Many variants, many applications
- Usually solved via alternating minimization algorithms
  - Alternating non-negative least squares (ANLS)
  - Projected gradient local hierarchical ALS (HALS)
  - Multiplicative updates
- pLSA is a well-known approach to topic modeling
  - Can be seen as an NMF

Literature

- David Skillicorn
  Understanding Complex Datasets: Data Mining with Matrix Decompositions (Chapter 8)
  Chapman and Hall, 2007
- Andrzej Cichocki, Rafal Zdunek, Anh Huy Phan, and Shun-ichi Amari
  Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-way Data Analysis and Blind Source Separation
  Wiley, 2009
- Yifeng Li and Alioune Ngom
  The NMF MATLAB Toolbox
  http://cs.uwindsor.ca/ li11112c/nmf
- Renaud Gaujoux and Cathal Seoighe
  NMF R package
  http://cran.r-project.org/web/packages/NMF/index.html
- References given at bottom of slides

Data Mining and Matrices

06 – Spectral Clustering

Prof. Dr. Rainer Gemulla

Universität Mannheim

FSS 2018

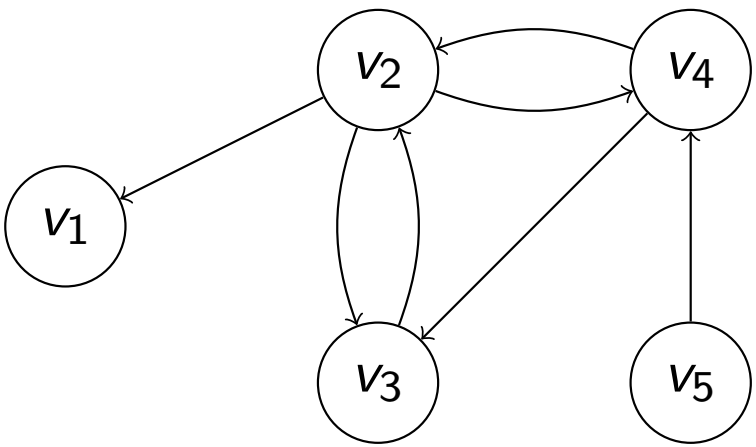
Graph mining

- Graphs everywhere, e.g.
  - Internet
  - World wide web
  - Social networks
  - Protein-protein interactions
  - Similarity graphs
- Goals of graph mining
  - As data mining: classification, clustering, outliers, patterns
  - Output often also one or more graphs
  - Interesting subgraphs (e.g., communities, near-cliques, clusters)
  - Interesting vertices (e.g., influential bloggers, PageRank, outliers)
  - Web mining (e.g., topic predicition, classification)
  - Web usage mining (e.g., frequent subgraphs, patterns)
  - Recommender systems (e.g., movie recommendation)
  - Knowledge bases (e.g., link prediction)
  - ...

Spectral analysis of matrices associated with graphs is an important tool in graph mining. Our focus: spectral clustering and link analysis.

A graph is a matrix is a graph

- Let $G = (V, E)$ be a (weighted) graph
- Vertices $V = \{ v_1, \ldots, v_n \}$
- Edge $(i, j) \in E$ has positive weight w_{ij} (or 1 if graph is unweighted)
- Convention: absent edges $(i, j) \notin E$ have weight $w_{ij} = 0$
- **Adjacency matrix** $\boldsymbol{W}$ is $n \times n$ matrix with entries w_{ij}
- Undirected graph $\implies$ $\boldsymbol{W}$ symmetric ($\boldsymbol{W} = \boldsymbol{W}^T$)
- (Out-)degree of vertex i given by $d_i = \sum_j w_{ij} = \boldsymbol{w}_i^T \mathbf{1}$
- **Degree matrix** $\boldsymbol{D}$ is $n \times n$ diagonal matrix with $d_{ii} = d_i$



G

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

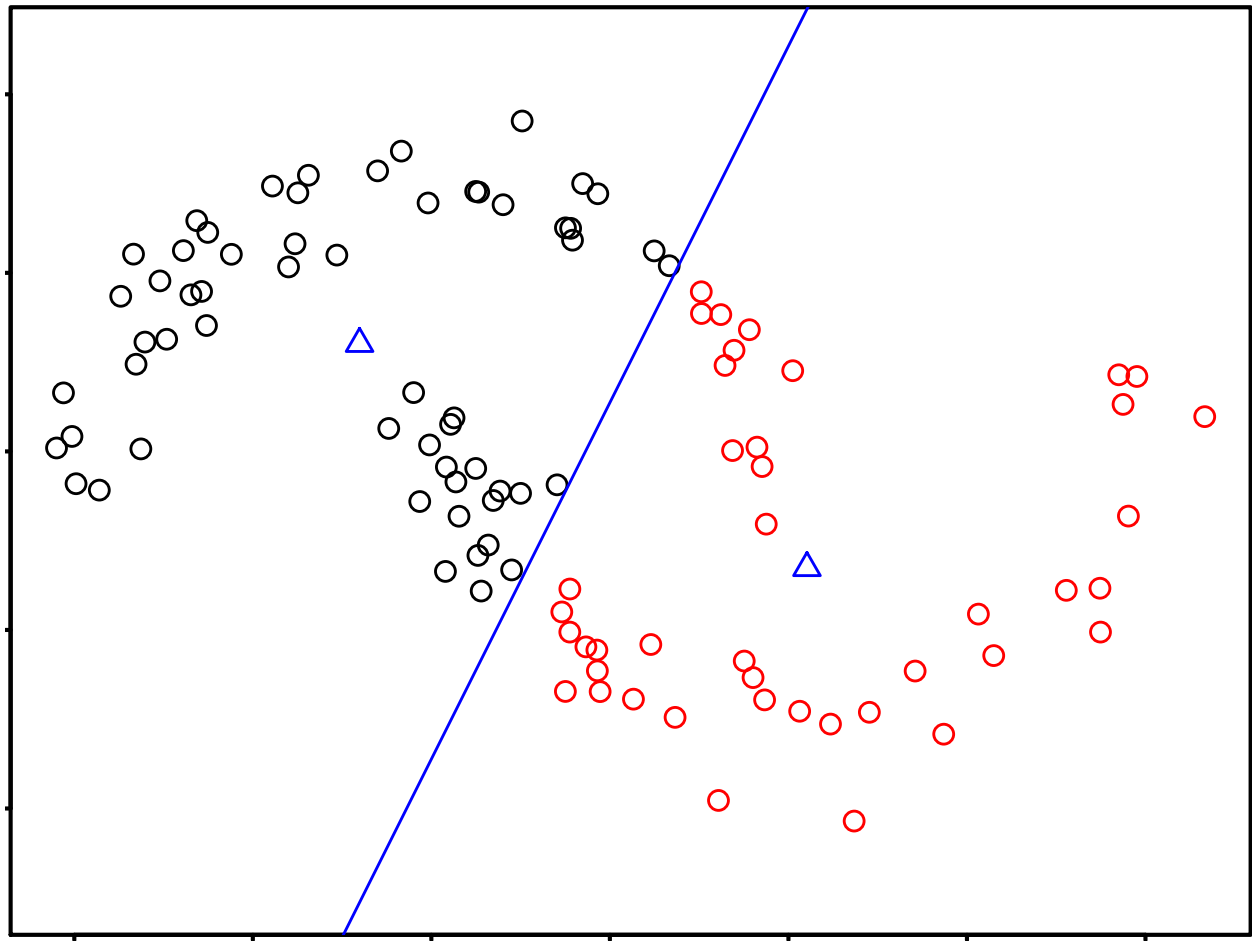
$\boldsymbol{W}$

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$\boldsymbol{D}$

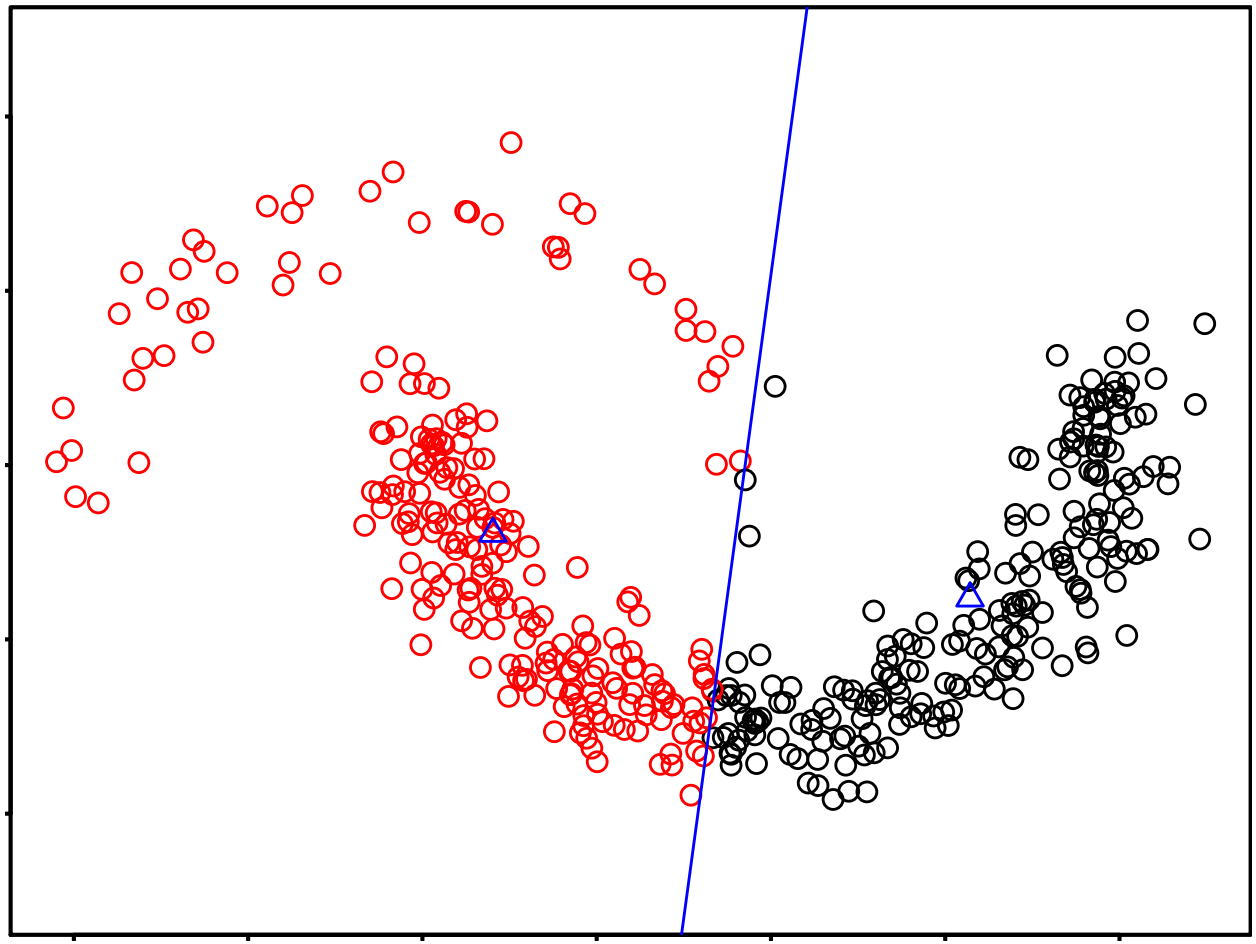
Outline

k-Means example (1)



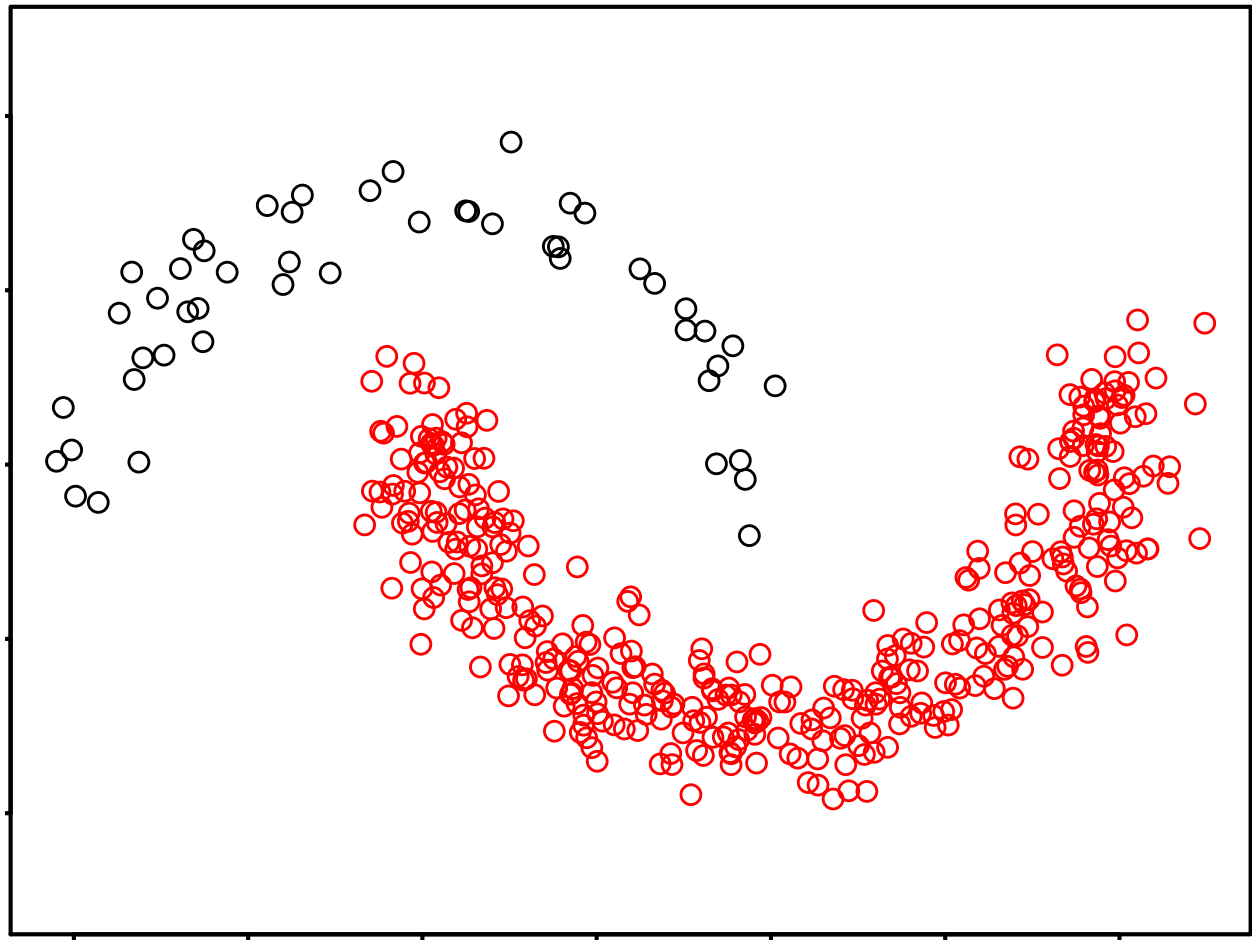
k-Means cannot detect non-convex clusters well.

k-Means example (2)



k-Means is sensitive to skew in cluster sizes.

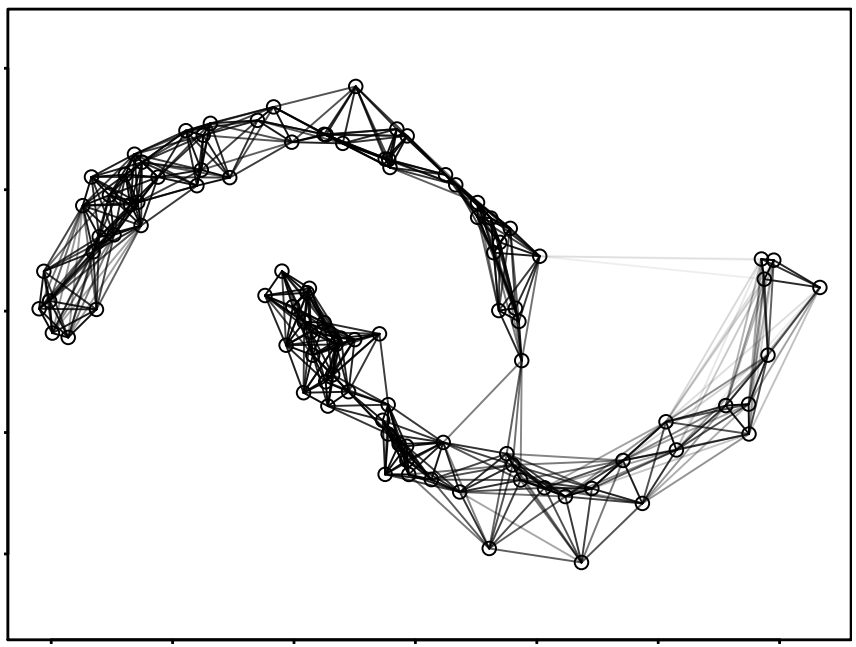
A better clustering



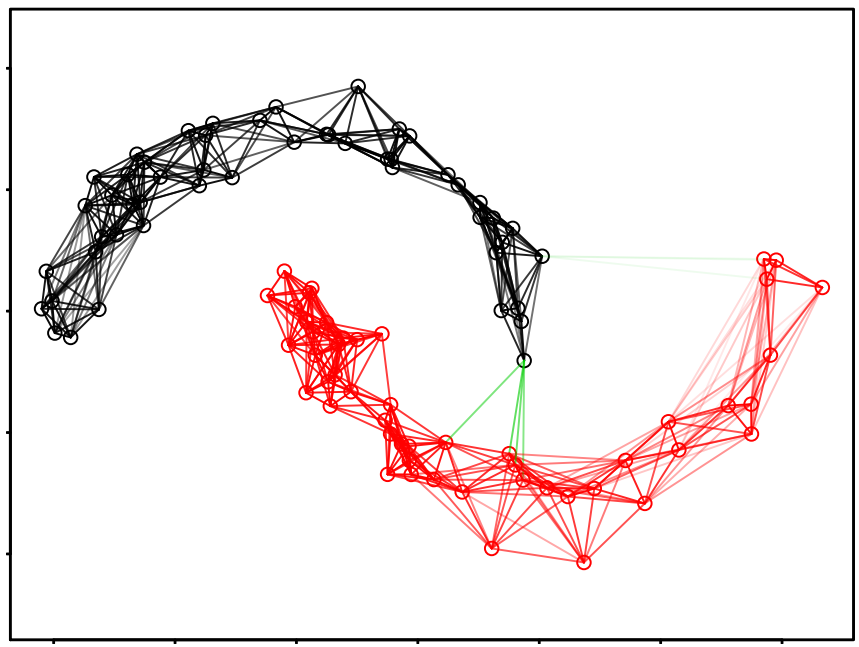
In this clustering, points within a cluster are close to their neighbors, but not necessarily to all the points in the cluster.

Graph-based clustering

1. Given a dataset, construct a *similarity graph* modeling local neighborhood relationships
2. Partition the similarity graph using suitable *graph cuts*



Similarity graph

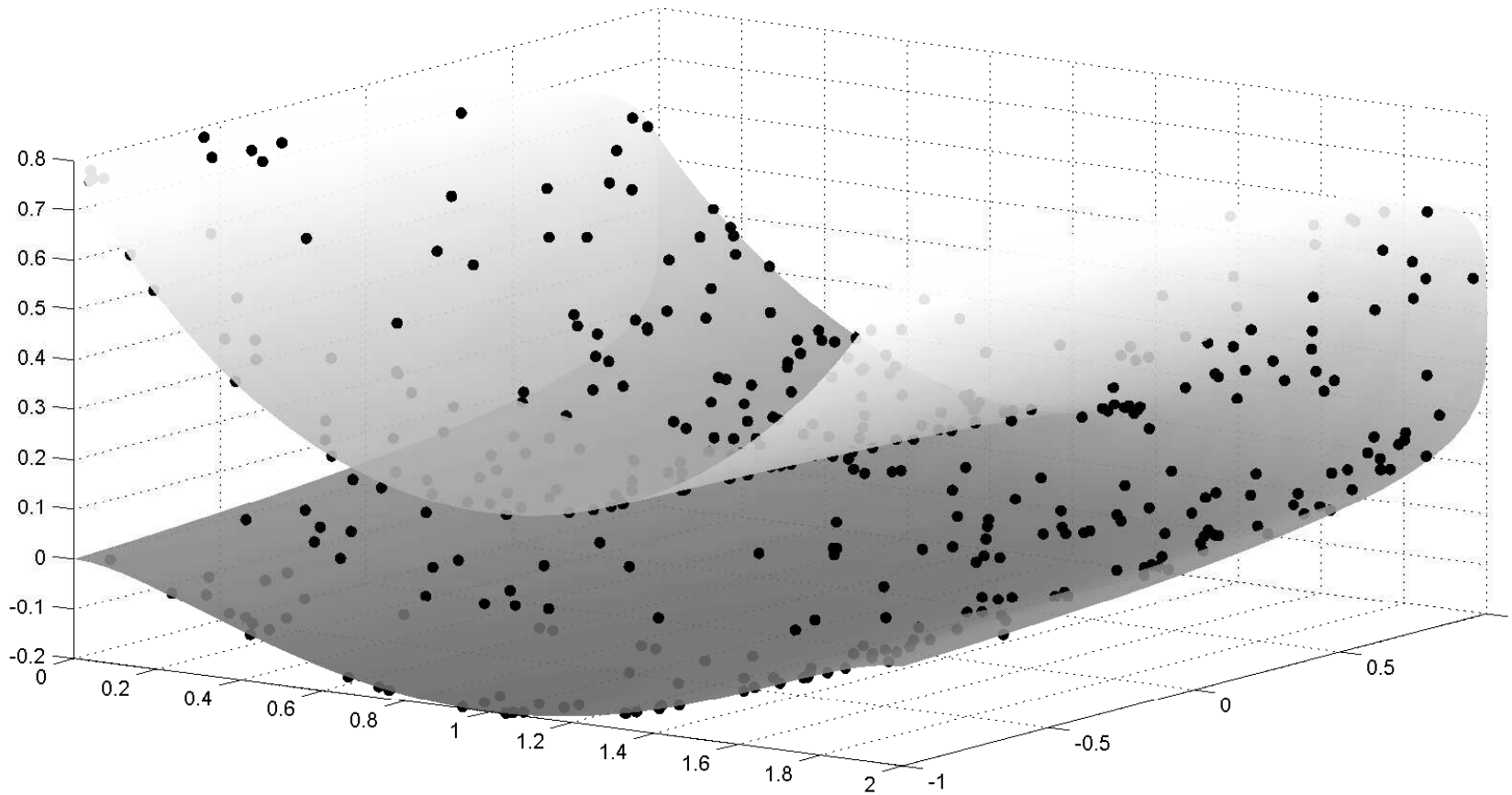


Clustering

Discussion

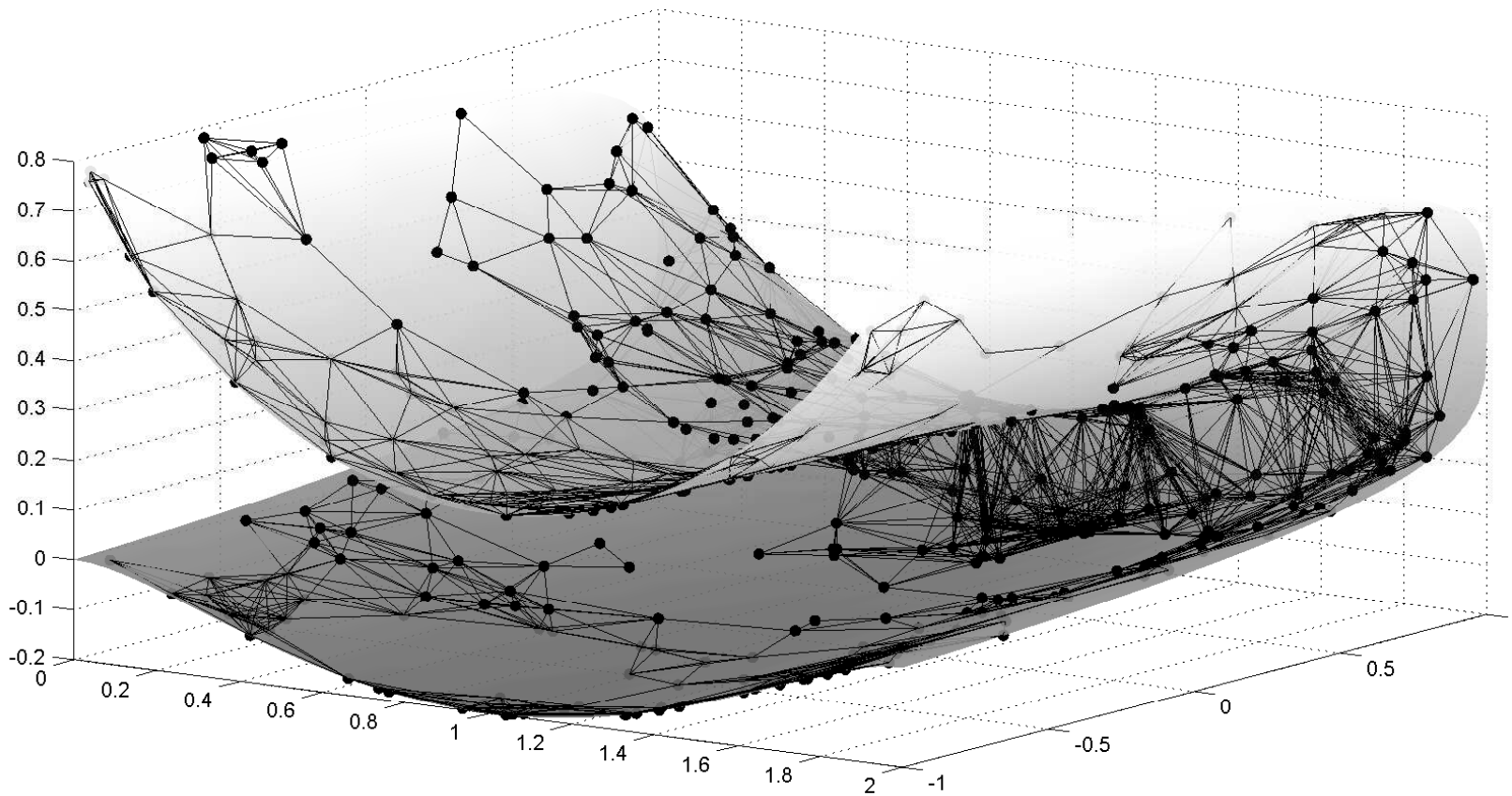
- Clustering
  1. Points within a cluster should be similar
  2. Points in different clusters should be dissimlar
- k-Means is global
  1. *All* points within a cluster should be similar (close)
  2. Points in different clusters should be dissimilar (far apart)
- Graph-based clustering is local
  1. *Neighboring* points within a cluster should be similar (close)
  2. Points in different clusters should be dissimilar (far apart)

Another example (1)



Global distances can be misleading.

Another example (2)



Distances to neighbors may capture structure better.

Which cut? (1)

- $G = (V, E)$: Undirected, weighted similarity graph
- Cut = partitioning of vertices into two partitions $A \subset V$ and $\bar{A} = V \setminus A$
- For us: A and $\bar{A}$ correspond to clusters
- **Minimum cut** is cut that minimizes weight

$$\text{cut}(A, \bar{A}) = \sum_{i \in A, j \in \bar{A}} w_{ij}$$

- Can be solved efficiently (in P)
- Often not useful in practice, e.g., may separate a single vertex
  $\rightarrow$ Need to balance cut weight and cluster sizes

Which cut? (2)

- **Minimum ratio cut** (penalize different sizes w.r.t. vertices)

$$\text{RatioCut}(A, \bar{A}) = \left(\frac{1}{|A|} + \frac{1}{|\bar{A}|} \right) \sum_{i \in A, j \in \bar{A}} w_{ij}$$

- **Minimum normalized cut** (penalize different sizes w.r.t. degrees)

$$\text{Ncut}(A, \bar{A}) = \left(\frac{1}{\text{vol}(A)} + \frac{1}{\text{vol}(\bar{A})} \right) \sum_{i \in A, j \in \bar{A}} w_{ij},$$

  where $\text{vol}(A) = \sum_{i \in A} d_i$
- Both problems are NP-hard

Spectral clustering is a relaxation of RatioCut or Ncut, is simple to implement, and can be solved efficiently.

Which cut? (3)

- Recall clustering objectives
  1. Points in same cluster should be similar (maximize within-cluster similarity)
  2. Points in different clusters should be dissimilar (minimize between-cluster similarity)
- (1) = $\text{vol}(A)$ and $\text{vol}(\bar{A})$ are both large
- (2) = minimize $\text{cut}(A, \bar{A})$
- cut, RatioCut, and Ncut all implement (2)
- Only Ncut additionally implements (1)
- Ncut captures both goals $\rightarrow$ usually good choice

Outline

Similarity graph

- In graph-based clustering, we first need to construct a **similarity graph**
  - Need notion of similarity and a way to construct an appropriate graph based on that similarity
  - How to do this depends on domain
  - Generally difficult
  - Can significantly affect results
- E.g., similarity for documents
  - Similarity of words in documents
  - Similarity of words in documents with tf/idf weighting
  - Similarity of topic distribution of documents
  - ...
- Here:
  - How to obtain a similarites from Euclidean data
  - How to construct a similarity graph from similarities

From distances to similarities

- Sometimes: Need to "convert" distances to similarities
- Large distance $\delta_{ij} \iff$ small similarity w_{ij} (and vice versa)
- Simplest choice: reciprocal (problematic, unbounded)

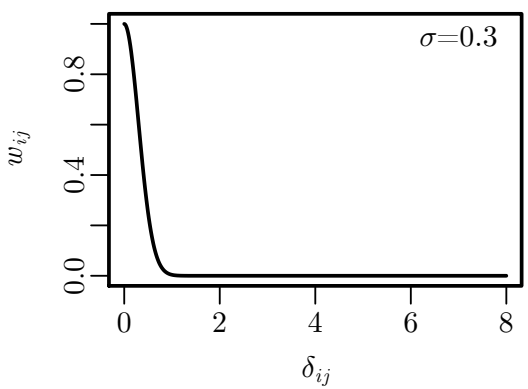
$$w_{ij} = \frac{1}{\delta_{ij}}$$

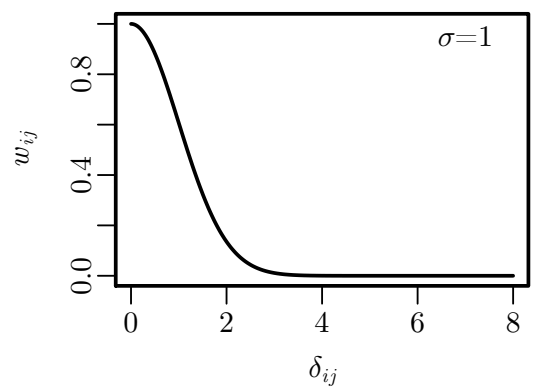
- Common choice for Euclidian data: **Gaussian kernel** (in $[0, 1]$)

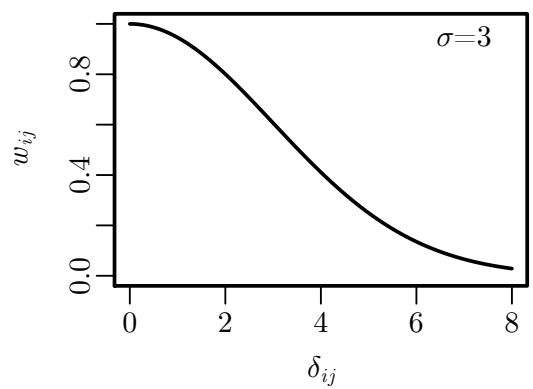
$$w_{ij} = \exp(-\delta_{ij}^2/(2\sigma^2)),$$

where δ_{ij} is Euclidean distance ($\|\boldsymbol{x}_i - \boldsymbol{x}_j\|$)

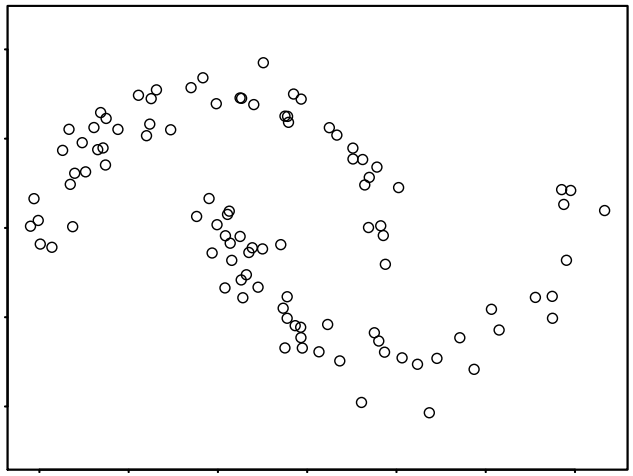
- Parameter σ controls what is considered local (large σ = large neighborhood)

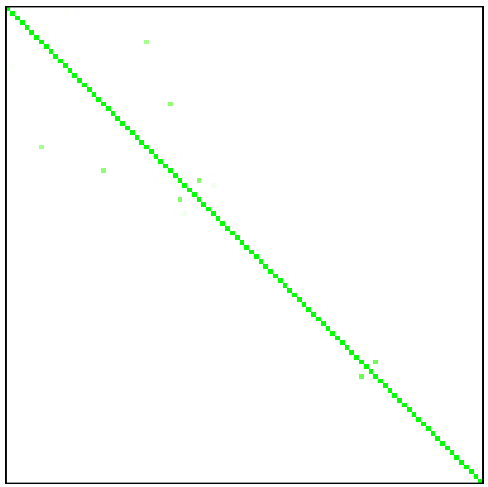




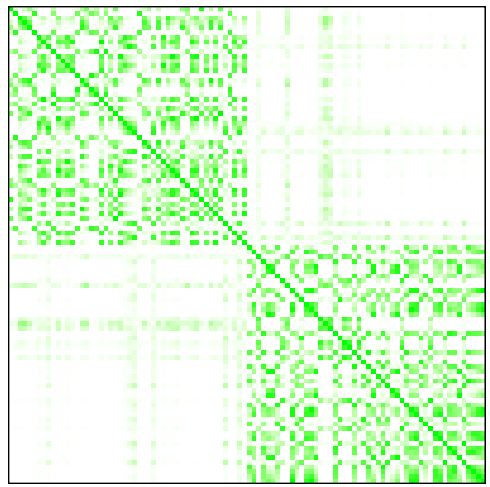


From distances to similarities (examples)

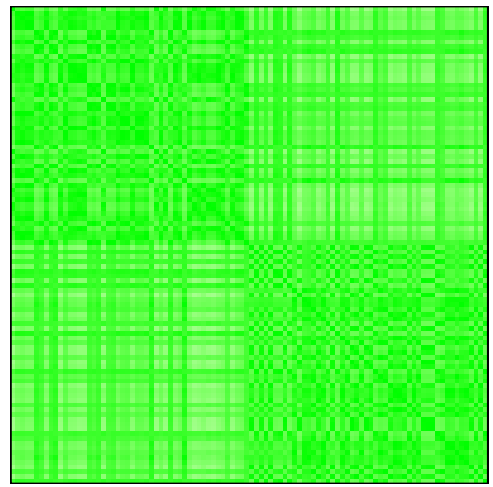




$\sigma = 0.01$
(too small)

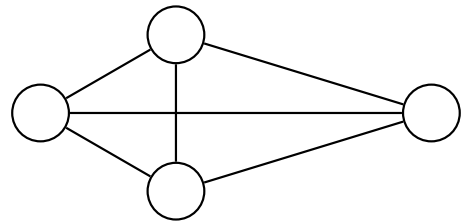


$\sigma = 0.4$
(good)

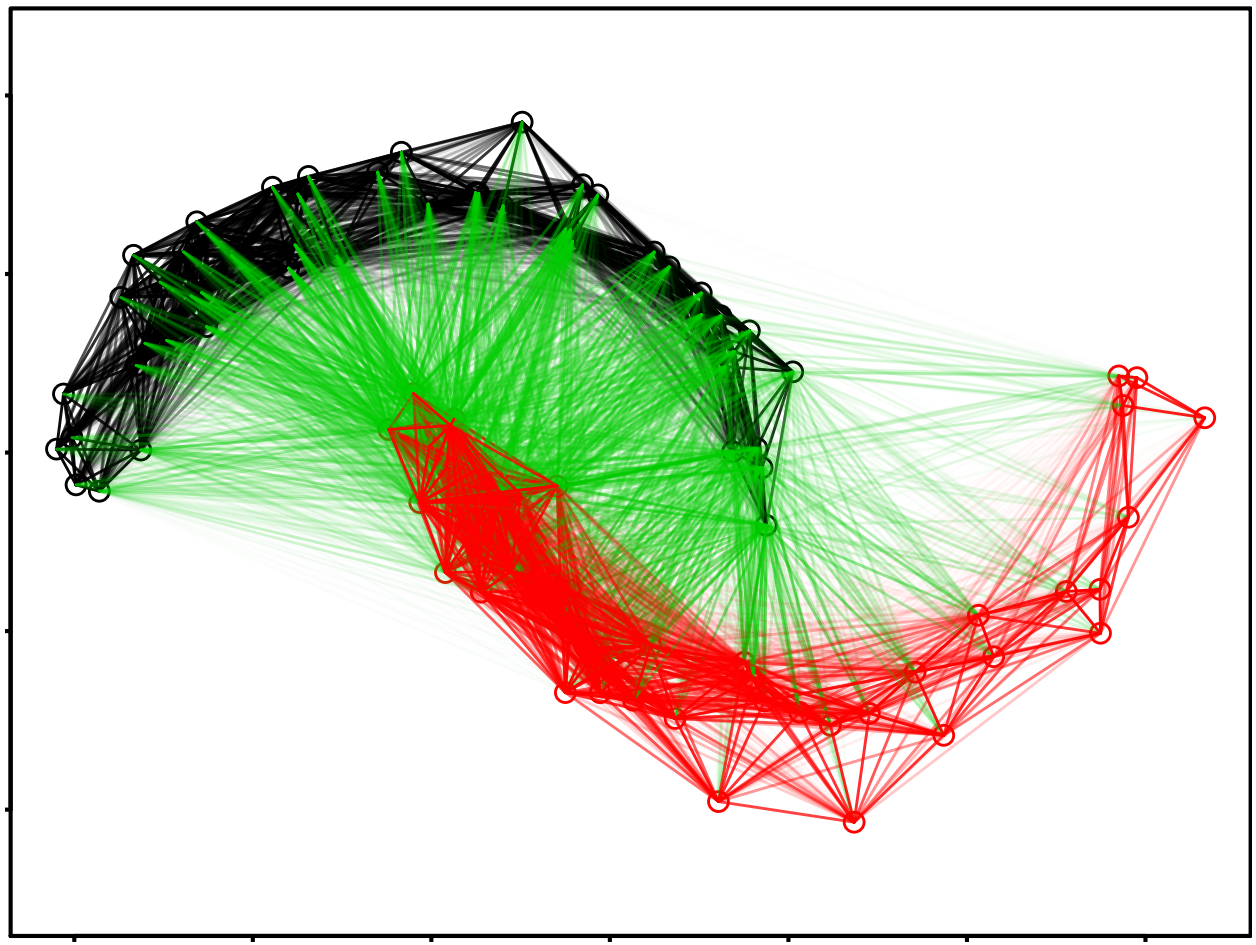


$\sigma = 3$
(too large)

Full graph

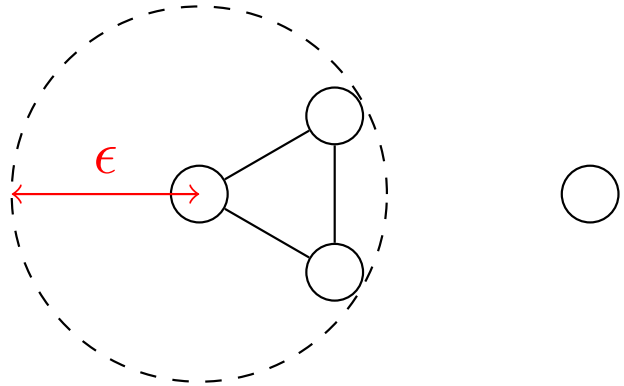


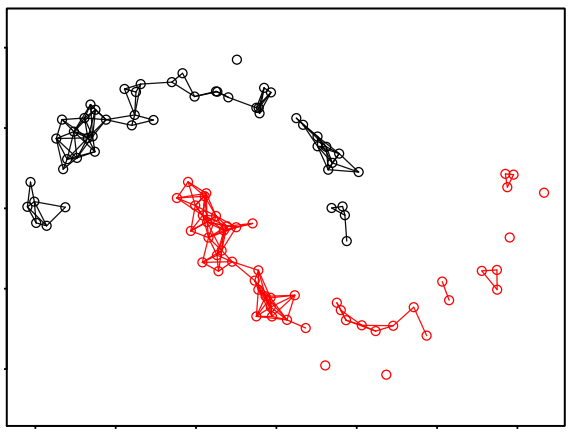
- Connect all pairs of vertices
- Weigh edges by similarity
- Generally expensive, not feasible for large datasets



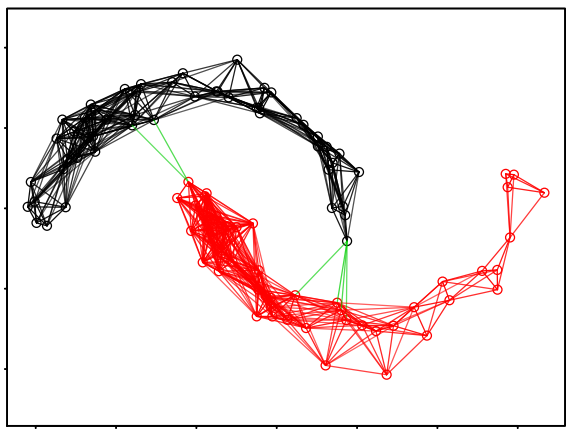
ϵ-Neighborhood graph

- Pick neighborhood size ϵ
- Connect vertices of distance $\leq \epsilon$
- Unweighted or weighted by similarity

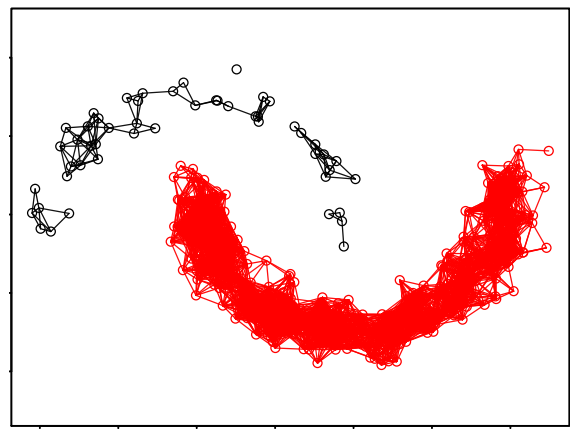




ϵ too small



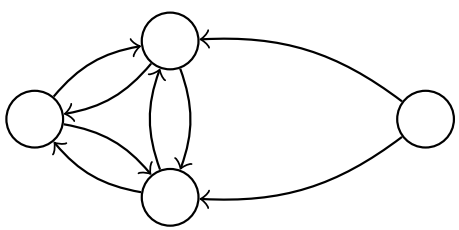
ϵ good



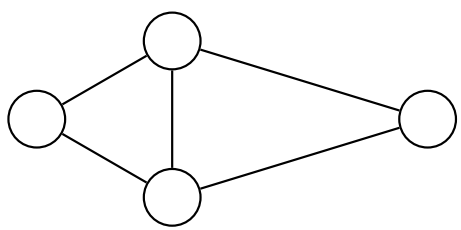
Skewed clusters: ϵ too large for red, too small for black

Nearest neighbor graphs

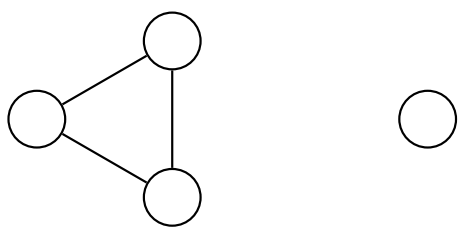
- Pick number k of neighbors
- Directed k-nearest neighbor graph
  - Add directed edge (i,j) if j is among k closest neighbors of i
  - But: need undirected graph for well-defined similarities
- (Symmetric) k-nearest neighbor graph
  - Connect (i, j) if (i, j) or (j, i) in directed k-NN graph (OR)
  - Each node has at least k, but potentially more than k "neighbors"
- Mutual k-nearest neighbor graph
  - Connect (i, j) if (i, j) and (j, i) in directed k-NN graph (AND)
  - Each node has at most k, but potentially less than k "neighbors"
- Weigh edges by similarity



directed 2-NN



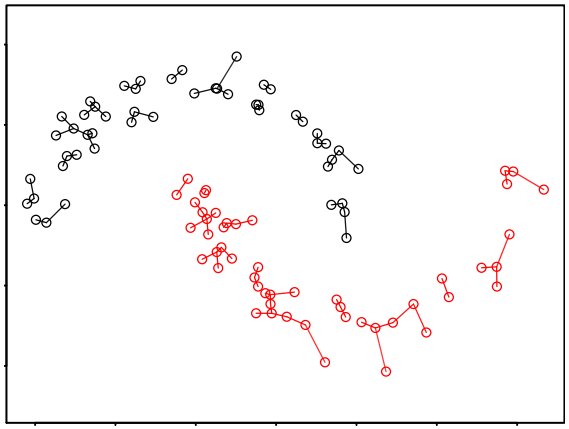
symmetric 2-NN



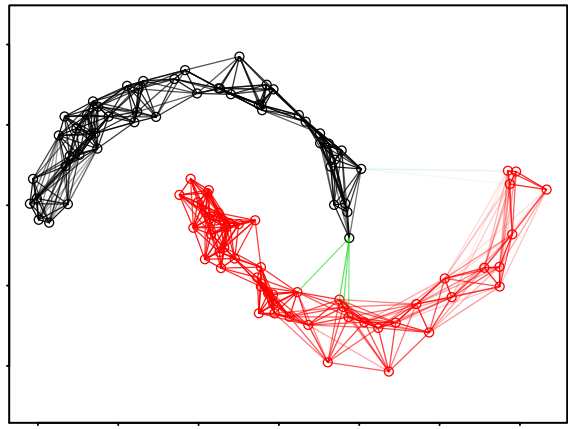
mutual 2-NN

k-Nearest neighbor graph (examples)

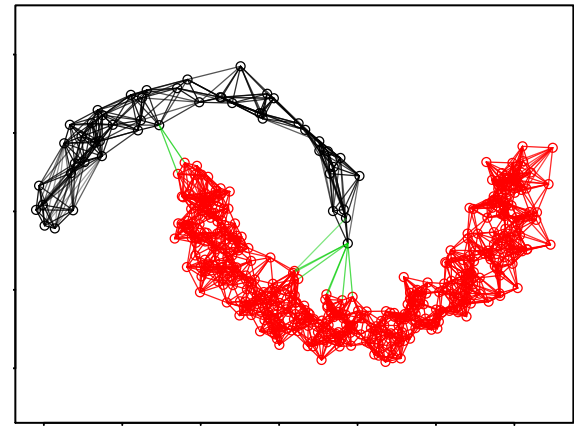
Symmetric k-NN graph



$k = 1$ (too small)

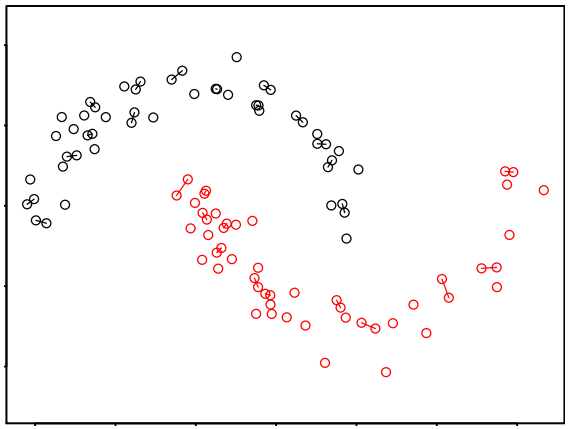


$k = 10$ (good)

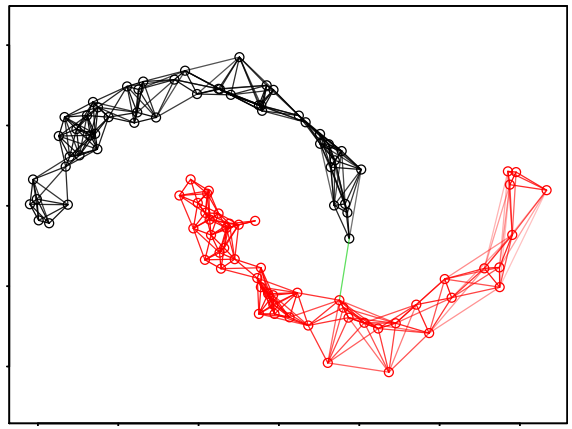


Skewed, $k = 10$ (good)

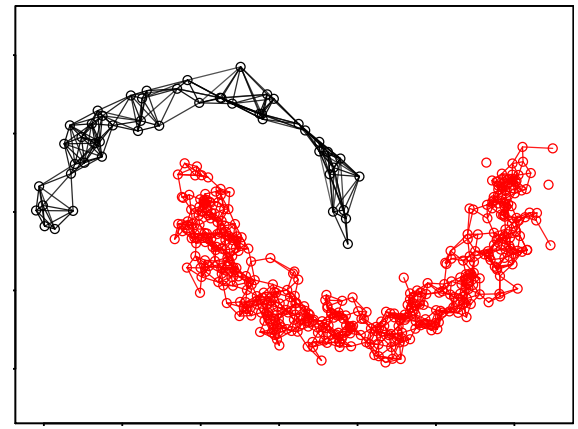
Mutual k-NN graph



$k = 1$ (too small)



$k = 10$ (good)



Skewed, $k = 10$ (too small)

Discussion (1)

- Construction of similarity graph non-trivial and not well-understood
- Clustering results sensitive to choice of graph
- Which similarity function?
  - Should capture similarity of most-similar objects well (other edges pruned by neighborhood graphs anyway)
  - Gaussian kernel common choice for Euclidean data
  - Generally application-dependent
- Which graph?
  - Fully connected graph often too large + requires suitable similarity function, dense graph
  - ϵ-neighborhood graph cannot deal well with clusters of different densities
  - k-NN graph can connect points in regions with different densities $\rightarrow$ Generally recommended choice, sparse graph
  - Mutual k-NN graph is somewhere in between

Discussion (2)

- Which parameters? (ϵ or k, σ)
  - ϵ or k should be small so that graph is sparse
  - But large enough to ensure that similarity graph is connected (or at least has fewer components than desired clusters)
  - Otherwise: clustering sizes arbitrarily unbalanced, sensitive to outliers
  - k-NN: try various values (start with, e.g., $k = O(\log(n))$
  - Mutual k-NN: no good heuristics known
  - ϵ-N: around length of longest edge in minimal spanning tree (problematic with outliers or clusters that are far apart)
  - σ: no. neighbors with similarity significantly larger than 0 "neither too small nor too large" (e.g., mean distance to k-th nearest neighbor, or using minimal spanning tree)

Skilled data miners do not run out of jobs.

Outline

Eigenvectors and eigenvalues

- A non-zero vector $\boldsymbol{v} \in \mathbb{R}^n$ is an **eigenvector** of $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ if

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

- λ is the corresponding **eigenvalue**
- The eigenvalues are the roots of the **characteristic polynomial**

$$p_{\boldsymbol{A}}(\lambda) = \det(\boldsymbol{A} - \lambda \boldsymbol{I})$$

- We can factor $p_{\boldsymbol{A}}(\lambda)$ as

$$p_{\boldsymbol{A}}(\lambda) = (\lambda - \lambda_1)^{n_1} \cdots (\lambda - \lambda_N)^{n_N},$$

  where $\sum_i n_i = n$
  - n_i is called the **algebraic multiplicity** of λ_i
  - There are $1 \leq m_i \leq n_i$ linearly independent eigenvectors associated with eigenvalue λ_i
  - m_i is called the **geometric multiplicity** of λ_i
  - Note: Some eigenvectors can be complex
- Collection of eigenvalues is called **spectrum** of $\boldsymbol{A}$

Eigendecomposition

- The **eigendecomposition** of $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is given by

$$\boldsymbol{A} = \boldsymbol{Q} \boldsymbol{\Lambda} \boldsymbol{Q}^{-1},$$

  where
  - $\boldsymbol{Q}$ is square and has eigenvectors as its columns
  - $\boldsymbol{\Lambda}$ is diagonal and has eigenvalues on its diagonal
- Does not always exist; if it does, $\boldsymbol{A}$ is called **diagonalizable**
- Some properties
  - When $\boldsymbol{A}$ is symmetric, there exists an eigendecomposition where $\boldsymbol{Q}$ is real and orthogonal
  - $\boldsymbol{A} = \boldsymbol{B}\boldsymbol{B}^T = (\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T)(\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T)^T = \boldsymbol{U}\boldsymbol{\Sigma}^2\boldsymbol{U}^T$
  - E.g., $\boldsymbol{A} = \boldsymbol{B}^T\boldsymbol{B} = (\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T)^T(\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T) = \boldsymbol{V}\boldsymbol{\Sigma}^2\boldsymbol{V}^T$
  - E.g., $\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^{-1}$, then $\text{tr}(\boldsymbol{A}) = \text{tr}(\boldsymbol{\Lambda}) = \sum_i \lambda_i$
  - $\text{rank}\,(\boldsymbol{A})$ = number of non-zero eigenvalues (counting multiplicity)

Outline

Graph Laplacian

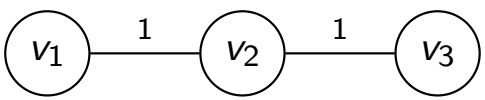
Definition

Let G be an undirected graph with positive edge weights. Denote by $\boldsymbol{W}$ the (weighted) adjacency matrix of G, and by $\boldsymbol{D}$ the degree matrix of G. Then

$$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W}$$

is called the **(unnormalized) graph Laplacian** of G.

Note that self edges ($w_{ii} > 0$) do not affect the graph Laplacian.



G

$$\boldsymbol{D} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad \boldsymbol{W} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \qquad \boldsymbol{L} = \begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix}$$

Graph Laplacians are the main tool for spectral clustering, but they have many other uses too (e.g., label propagation, graph drawing).

Properties of the graph Laplacian (1)

Theorem

For every vector $\boldsymbol{x} \in \mathbb{R}^n$, $f(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{L}\boldsymbol{x} = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}(x_i - x_j)^2$.

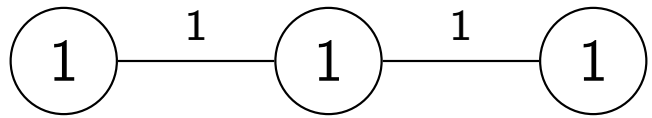
- $\boldsymbol{x}$ assigns a real value to each vertex
- $f(\boldsymbol{x})$ is a **quadratic form** and small when "similar" vertices (which are connected with high-weight edges) take similar values

Proof.

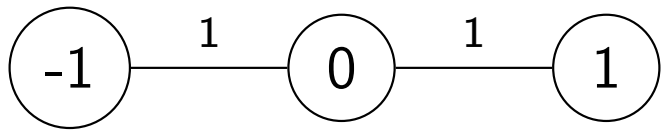
$$\begin{aligned}
\boldsymbol{x}^T \boldsymbol{L}\boldsymbol{x} &= \boldsymbol{x}^T \boldsymbol{D}\boldsymbol{x} - \boldsymbol{x}^T \boldsymbol{W}\boldsymbol{x} = \sum_{i=1}^{n} d_i x_i^2 - \sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} x_i x_j \\
&= \frac{1}{2}\left(\sum_{i=1}^{n} d_i x_i^2 - 2\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} x_i x_j + \sum_{j=1}^{n} d_j x_j^2\right) \\
&= \frac{1}{2}\left(\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}(x_i^2 - 2x_i x_j + x_j^2)\right) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}(x_i - x_j)^2
\end{aligned}$$

Properties of the graph Laplacian (2)

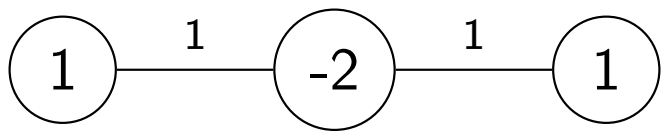
$$\boldsymbol{x}^T \boldsymbol{L}\boldsymbol{x} = \frac{1}{2}\sum_{i,j} w_{ij}(x_i - x_j)^2$$



$$\boldsymbol{x}^T \boldsymbol{L}\boldsymbol{x} = 0$$



$$\boldsymbol{x}^T \boldsymbol{L}\boldsymbol{x} = 2$$



$$\boldsymbol{x}^T \boldsymbol{L}\boldsymbol{x} = 18$$

Properties of the graph Laplacian (3)

A matrix $\boldsymbol{A}^{n \times n}$ is called **positive semi-definite** if $\boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x} \geq 0$ for any $\boldsymbol{x} \in \mathbb{R}^n$.

Theorem

$\boldsymbol{L}$ is symmetric and positive semi-definite.

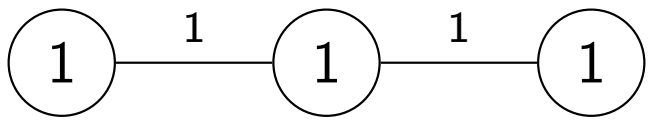
- Implies that $f(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{L} \boldsymbol{x}$ is a convex function
- Implies that $\boldsymbol{L} = \boldsymbol{P} \boldsymbol{P}^T$ for some $\boldsymbol{P}$ (oriented incidence matrix)

Proof. Since $\boldsymbol{D}$ and $\boldsymbol{W}$ are symmetric, so is $\boldsymbol{L}$. Since $\boldsymbol{x}^T \boldsymbol{L} \boldsymbol{x} \geq 0$ (see slide 30) for all $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{L}$ is positive semi-definite.

Properties of the graph Laplacian (4)

Theorem

The smallest eigenvalue of $\boldsymbol{L}$ is zero, the corresponding eigenvector is the constant one vector $\mathbf{1}$.



$$\lambda_3 = 0$$

Proof. The row sums satisfy $\boldsymbol{L}\mathbf{1} = \mathbf{0} = 0\mathbf{1}$ by construction. For smallest, see next slide.

Properties of the graph Laplacian (5)

Theorem

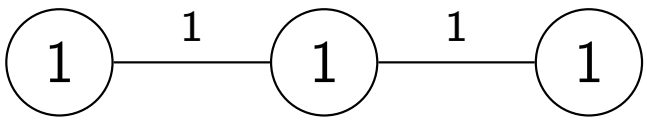
All eigenvalues are non-negative and real-valued, i.e.,
$\lambda_1 \geq \ldots \geq \lambda_{n-1} \geq \lambda_n = 0$.

Proof. All eigenvalues of a symmetric matrix are real. If $\boldsymbol{L}\boldsymbol{v} = \lambda \boldsymbol{v}$, then $\boldsymbol{v}^T \boldsymbol{L}\boldsymbol{v} = \lambda \|\boldsymbol{v}\|^2 \geq 0$ and thus $\lambda \geq 0$.

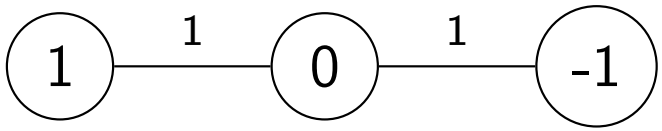
Connected graphs

Theorem

If G is connected, then eigenvalue 0 has multiplicity 1, i.e., $\lambda_{n-1} > 0$.



$\lambda_3 = 0$



$\lambda_2 = 1 > 0$

Proof. Recall that $\mathbf{1}$ is an eigenvector of $\boldsymbol{L}$ with eigenvalue 0. Suppose that $\mathbf{0} \neq \boldsymbol{v} \neq c\mathbf{1}$ is an eigenvector of $\boldsymbol{L}$ with eigenvalue λ. Since G is connected, this implies that there are two neighboring vertices i' and j' such that $v_{i'} \neq v_{j'}$. Now

$$\lambda \|\boldsymbol{v}\|^2 = \boldsymbol{v}^T \boldsymbol{L}\boldsymbol{v} = \frac{1}{2}\sum_{i,j} w_{ij}(v_i - v_j)^2 \geq w_{i'j'}(v_{i'} - v_{j'})^2 > 0$$

so that $\lambda > 0$.

Connected components

Theorem

The multiplicity k of eigenvalue 0 is equal to the number of connected components $G_1, \ldots, G_k$ of G. The corresponding eigenspace is spanned by the indicator vectors $\mathbf{1}_{G_i}$ (value 1 for vertices in G_i, value 0 otherwise).

Proof. Let $\boldsymbol{L}_1, \ldots, \boldsymbol{L}_k$ be the graph Laplacian of the connected components. Order w.l.o.g. the vertices by their component so that

$$\boldsymbol{L} = \begin{pmatrix} \boldsymbol{L}_1 & & & \\ & \boldsymbol{L}_2 & & \\ & & \ddots & \\ & & & \boldsymbol{L}_k \end{pmatrix}.$$

Since $\boldsymbol{L}$ is block-diagonal, the spectrum of $\boldsymbol{L}$ is given by the union of the spectra of the $\boldsymbol{L}_i$. The corresponding eigenvectors are the eigenvectors of $\boldsymbol{L}_i$, filled with 0 at positions of other blocks.

Connected components (example)

$$\boldsymbol{L} = \begin{pmatrix} 1 & -1 & 0 & & & \\ -1 & 2 & -1 & & & \\ 0 & -1 & 1 & & & \\ & & & 1 & -1 & 0 \\ & & & -1 & 2 & -1 \\ & & & 0 & -1 & 1 \end{pmatrix}$$

$\lambda_6 = 0$

$\lambda_5 = 0$

$\lambda_4 = 1$

Outline

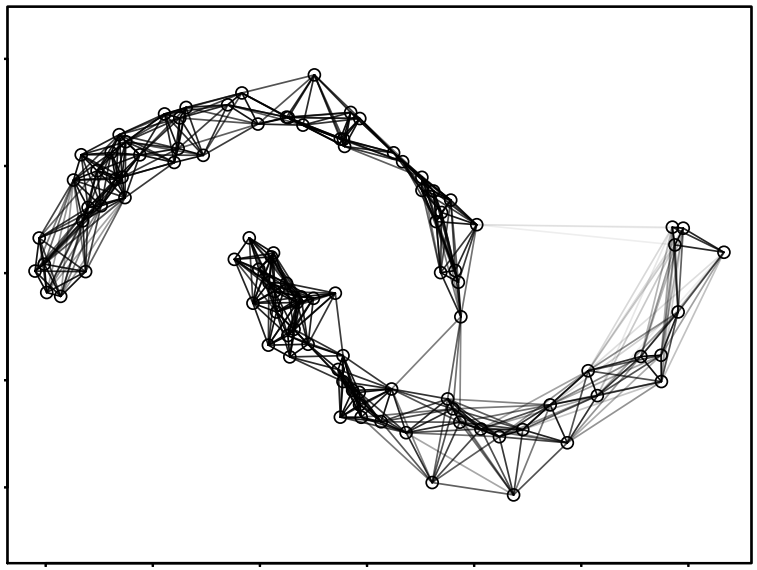
Outline

Algorithm

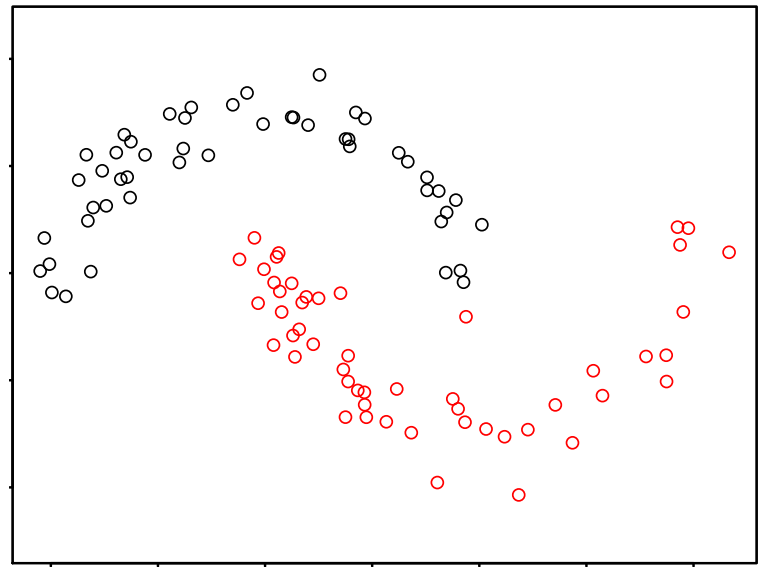
- Algorithm to construct k clusters
  1. Construct similarity graph $\boldsymbol{W}$
  2. Compute its (unnormalized) graph Laplacian $\boldsymbol{L}$
  3. Compute the last k eigenvectors $\boldsymbol{u}_n, \ldots, \boldsymbol{u}_{n-k+1}$ of $\boldsymbol{L}$ (i.e., having k smallest eigenvalues)
  4. Construct $n \times k$ matrix $\boldsymbol{U} = \begin{pmatrix} \boldsymbol{u}_n & \boldsymbol{u}_{n-1} & \cdots & \boldsymbol{u}_{n-k+1} \end{pmatrix}$
  5. Cluster the rows of $\boldsymbol{U}$ using k-means
- Simple, easy to implement
- Main trick: represent (or "embed") each vertex into $\mathbb{R}^k$ (= rows of $\boldsymbol{U}$)
- Change of representation enhances clustering properties in the data

Why does this work? Why are we interested in the smallest eigenvalues?

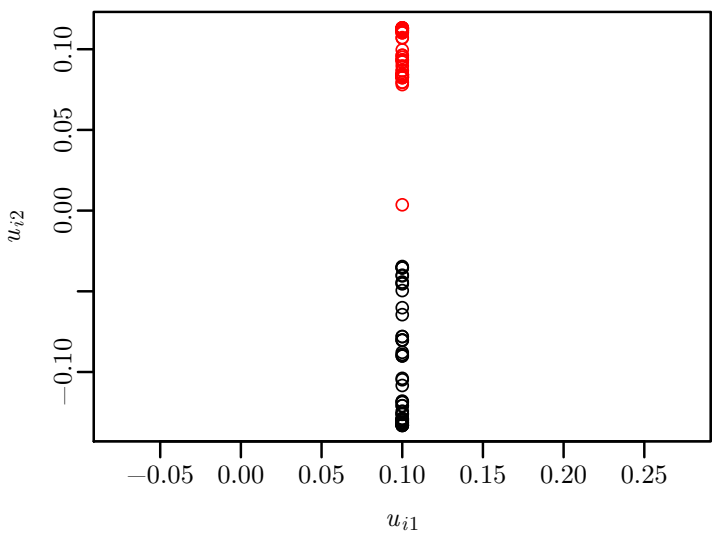
Unnormalized spectral clustering (example, 1)

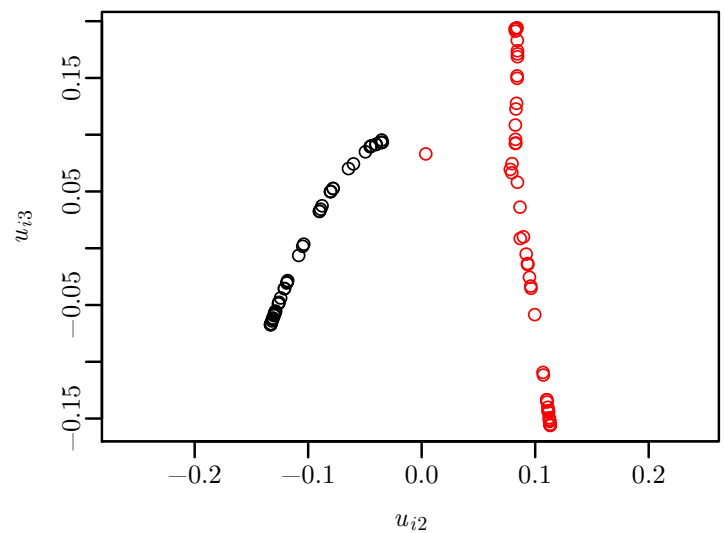


Similarity graph

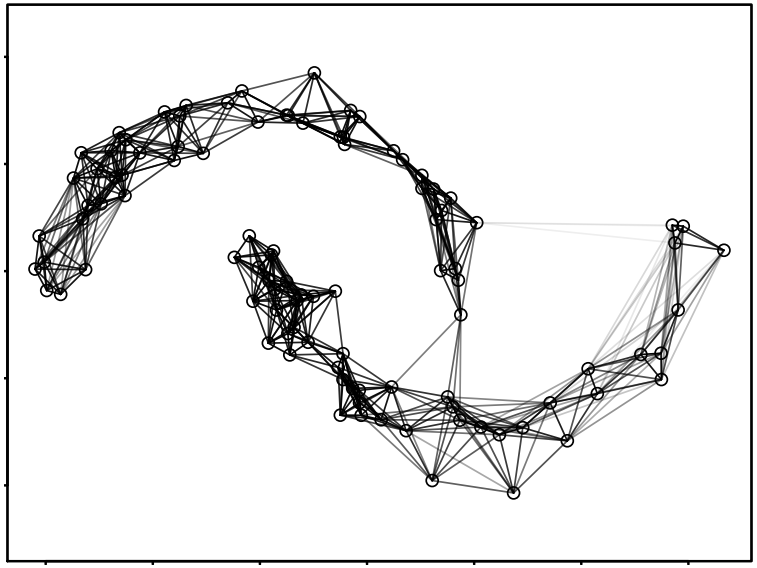


Spectral clustering ($k = 2$)

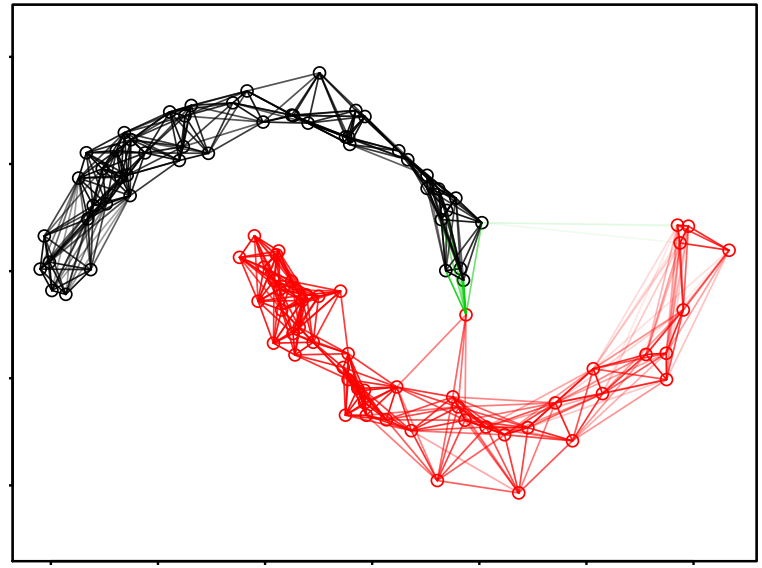




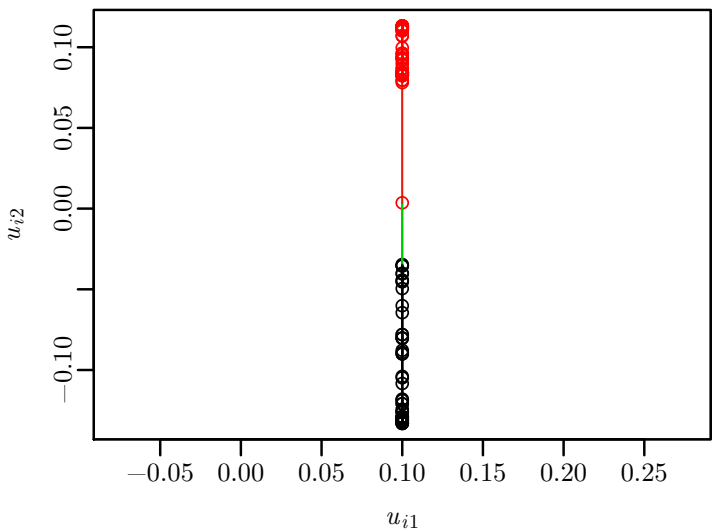
Unnormalized spectral clustering (example, 2)

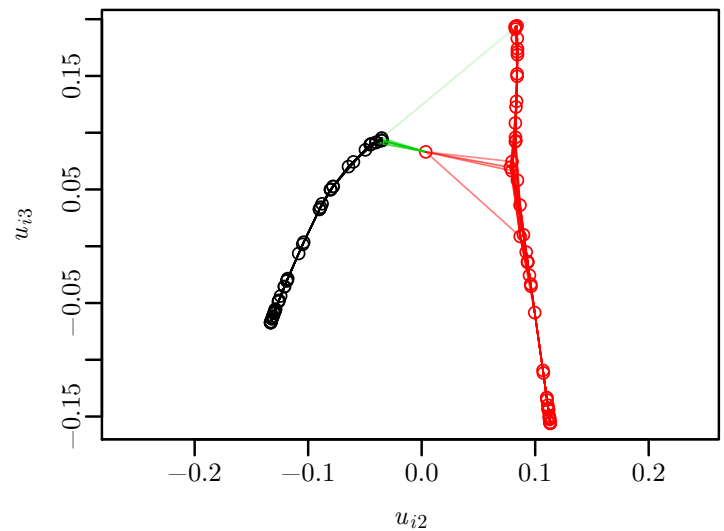


Similarity graph

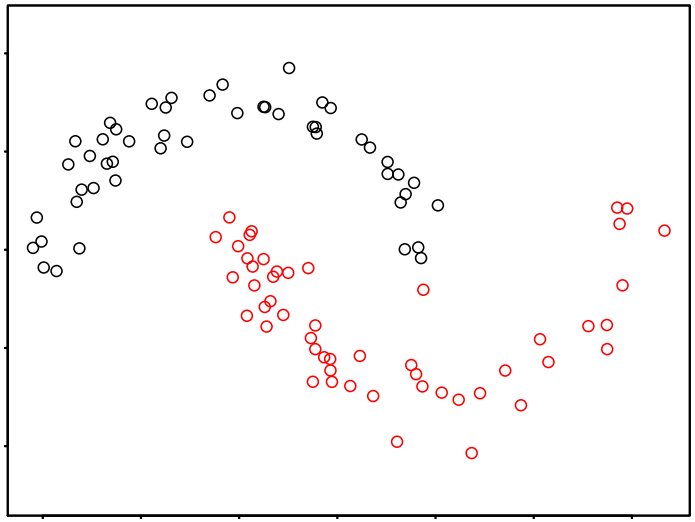


Spectral clustering ($k = 2$)

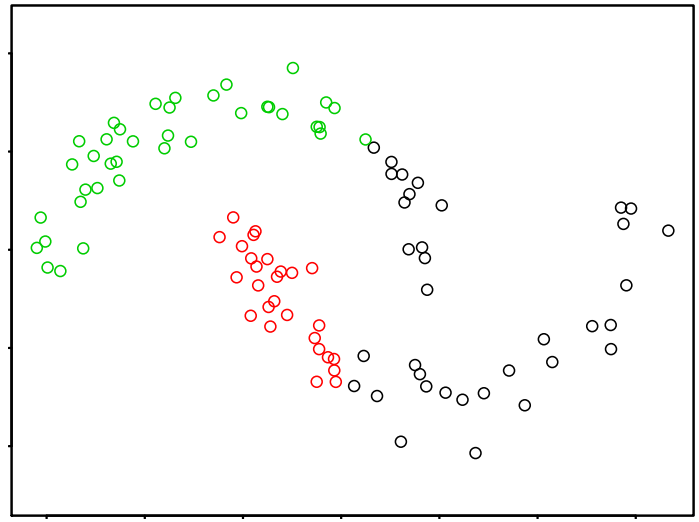




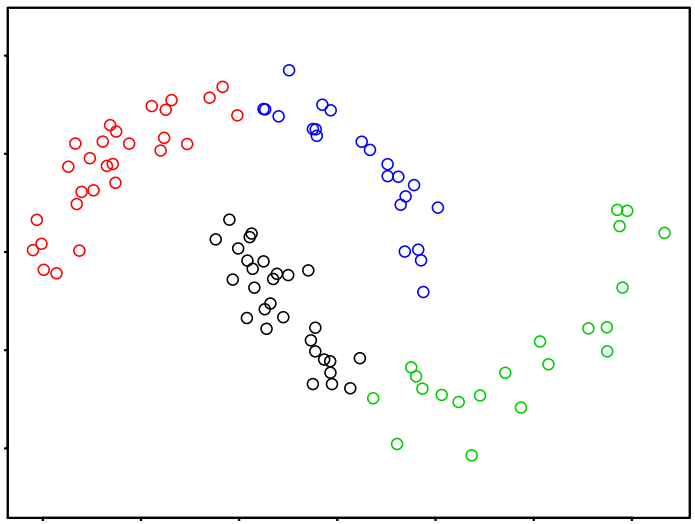
Unnormalized spectral clustering (example)



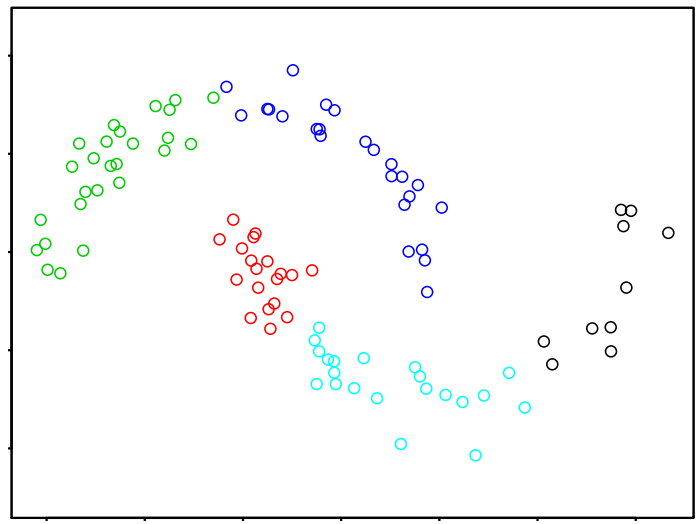
Spectral clustering ($k = 2$)



Spectral clustering ($k = 3$)



Spectral clustering ($k = 4$)



Spectral clustering ($k = 5$)

Why does spectral clustering work? (1)

- Consider the minimum ratio cut problem ($k = 2$)

$$\min_{A \subset V} \text{RatioCut}(A, \bar{A}) = \min_{A \subset V} \sum_{i \in A, j \in \bar{A}} w_{ij} \left(\frac{1}{|A|} + \frac{1}{|\bar{A}|} \right)$$

- Given A, set $\boldsymbol{x} \in \mathbb{R}^n$ such that

$$x_i = \begin{cases} \sqrt{|\bar{A}|/|A|} & \text{if } v_i \in A \\ -\sqrt{|A|/|\bar{A}|} & \text{if } v_i \in \bar{A} \end{cases}$$

- Easy to show
  1. $\boldsymbol{x}^T \boldsymbol{L} \boldsymbol{x} = n \cdot \text{RatioCut}(A, \bar{A})$
  2. $\sum_{i=1}^{n} x_i = 0$ so that $\boldsymbol{x} \perp \mathbf{1}$
  3. $\|\boldsymbol{x}\|^2 = n$

Why does spectral clustering work? (2)

- Minimum ratio cut can be rewritten as

$$\begin{aligned} \text{minimize} \quad & \boldsymbol{x}^T \boldsymbol{L} \boldsymbol{x} \\ \text{subject to} \quad & \boldsymbol{x} \perp \mathbf{1} \\ & \|\boldsymbol{x}\| = \sqrt{n} \\ & \boldsymbol{x} \text{ takes form defined in previous slide} \end{aligned}$$

- Still NP-hard; relax by dropping discreteness constraint

$$\begin{aligned} \text{minimize} \quad & \boldsymbol{x}^T \boldsymbol{L} \boldsymbol{x} \\ \text{subject to} \quad & \boldsymbol{x} \perp \mathbf{1} \\ & \|\boldsymbol{x}\| = \sqrt{n} \end{aligned}$$

- By *Rayleigh-Ritz theorem*: solution is eigenvector corresponding to second-smallest eigenvalue (appropriately normalized)
- $\boldsymbol{u}_{n-1}^T \boldsymbol{L} \boldsymbol{u}_{n-1} = \boldsymbol{u}_{n-1}^T \lambda_{n-1} \boldsymbol{u}_{n-1} = n\lambda_{n-1}$
- Thus: $\lambda_{n-1} \leq \min_{A \subset V} \text{RatioCut}(A, \bar{A})$
- Similar arguments for $k > 2$: solutions of relaxation = eigenvectors $\boldsymbol{u}_{n-1}, \ldots, \boldsymbol{u}_{n-k+1}$ (see exercise)

Why does spectral clustering work? (3)

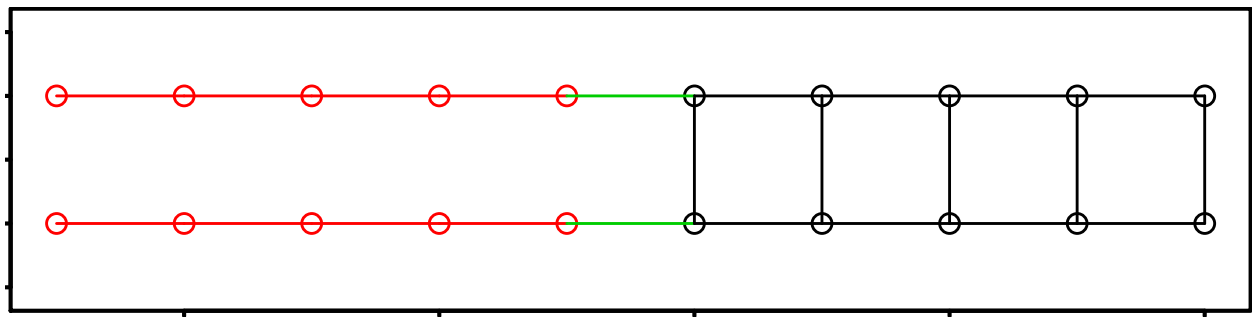
- Recall

$$x_i = \begin{cases} \sqrt{|\bar{A}|/|A|} & \text{if } v_i \in A \\ -\sqrt{|A|/|\bar{A}|} & \text{if } v_i \in \bar{A} \end{cases}$$

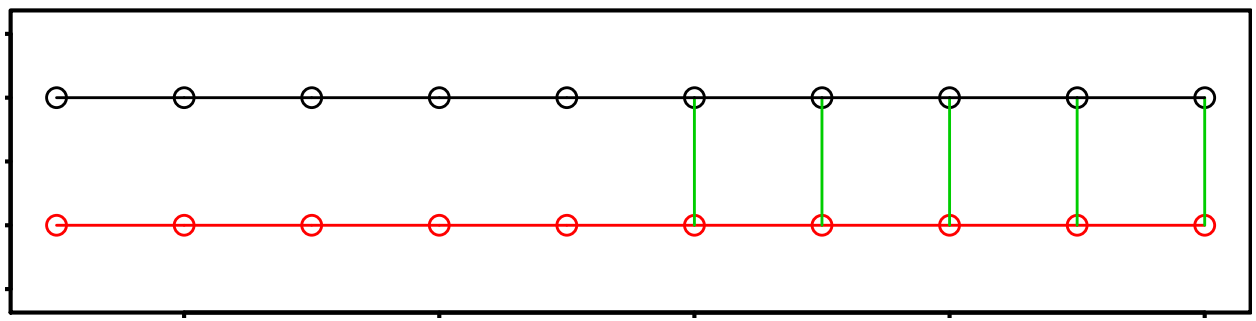
  in the constrained solution
- Need to obtain clustering from $\boldsymbol{u}_{n-1}$
  - Generally does not satisfy the discreteness constraints
  - Simple heuristic: use sign as cluster indicator (threshold=0)
  - Optimal: pick threshold that minimizes RatioCut (*optimal thresholding*)
  - k-means often used in practice (also works well for $k > 2$)
- In general, no guarantees to obtain good solution
- But: popular because simple, standard linear algebra problem that tends to work well in practice
- Approximation of balanced graph cuts (up to constant factor) still hard

Cockroach graph

- Example where spectral clustering performs particularly bad
- Minimum ratio cut $8/n$
- Spectral clustering ratio cut: 1
- Spectral clustering is $O(n)$ times worse



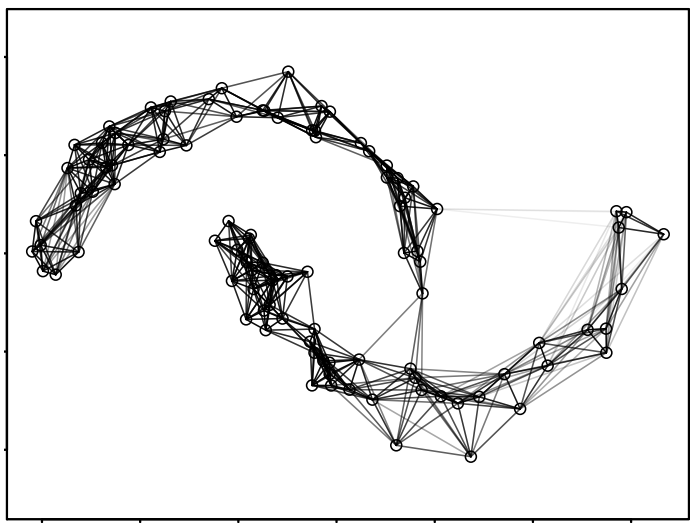
Minimum ratio cut

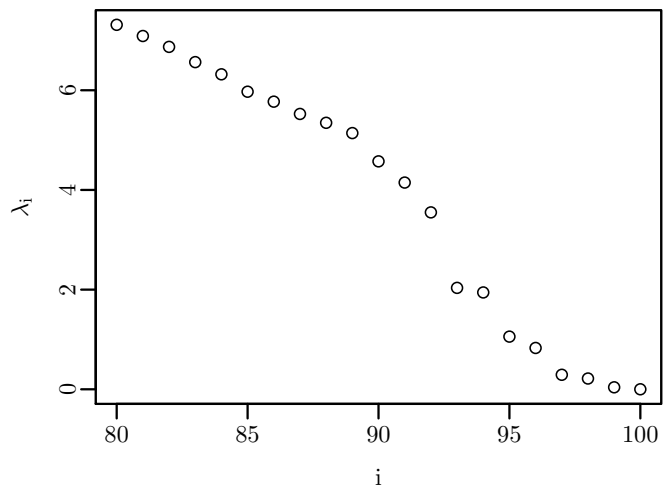


Ratio cut with spectral clustering and sign heuristic

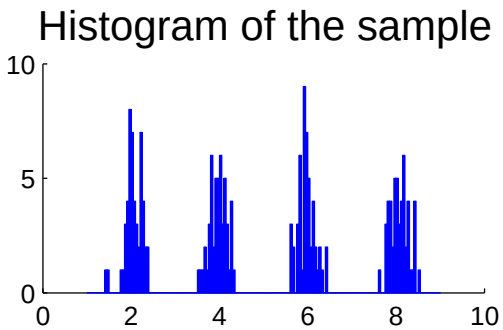
Discussion (1)

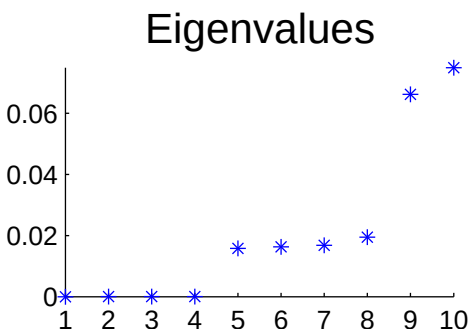
- Computation of eigenvectors
  - Graph can be very large
  - But Laplacian is sparse
  - Efficient algorithms for finding the eigendecomposition of such matrices exist
- Number k of clusters
  - Difficult problem
  - Standard approaches can be used
  - **Eigengap heuristic**: choose k such that eigenvalue $\lambda_1, \ldots, \lambda_{n-k}$ large, eigenvalues $\lambda_{n-k+1}, \ldots, \lambda_n$ small

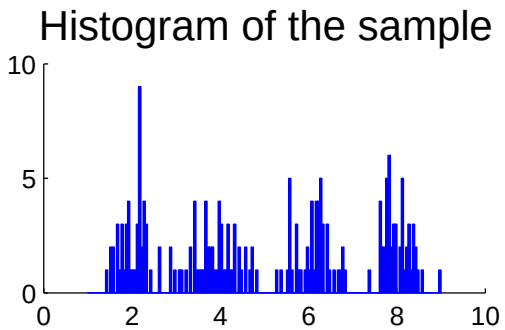


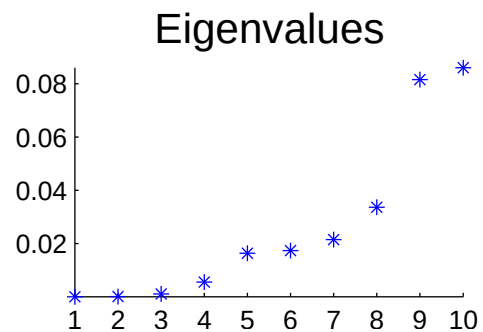


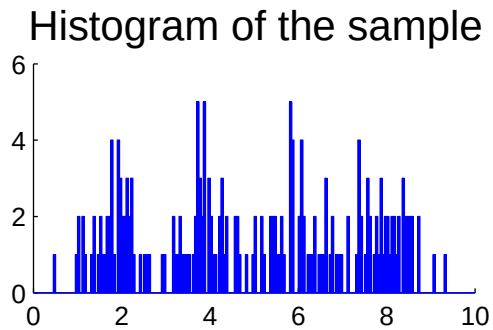
Discussion (2)

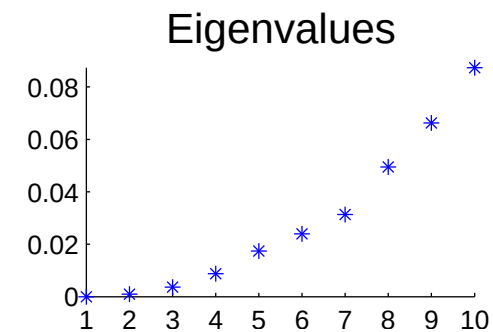












Outline

Normalized graph Laplacians

Definition

There are two common normalizations of the graph Laplacian:

$$\boldsymbol{L}_{\text{sym}} = \boldsymbol{D}^{-1/2}\boldsymbol{L}\boldsymbol{D}^{-1/2} = \boldsymbol{I} - \boldsymbol{D}^{-1/2}\boldsymbol{W}\boldsymbol{D}^{-1/2}$$
$$\boldsymbol{L}_{\text{rw}} = \boldsymbol{D}^{-1}\boldsymbol{L} = \boldsymbol{I} - \boldsymbol{D}^{-1}\boldsymbol{W}$$

- Normalization is performed w.r.t. degree
- $\boldsymbol{L}_{\text{sym}}$ is symmetric, $\boldsymbol{L}_{\text{rw}}$ is not

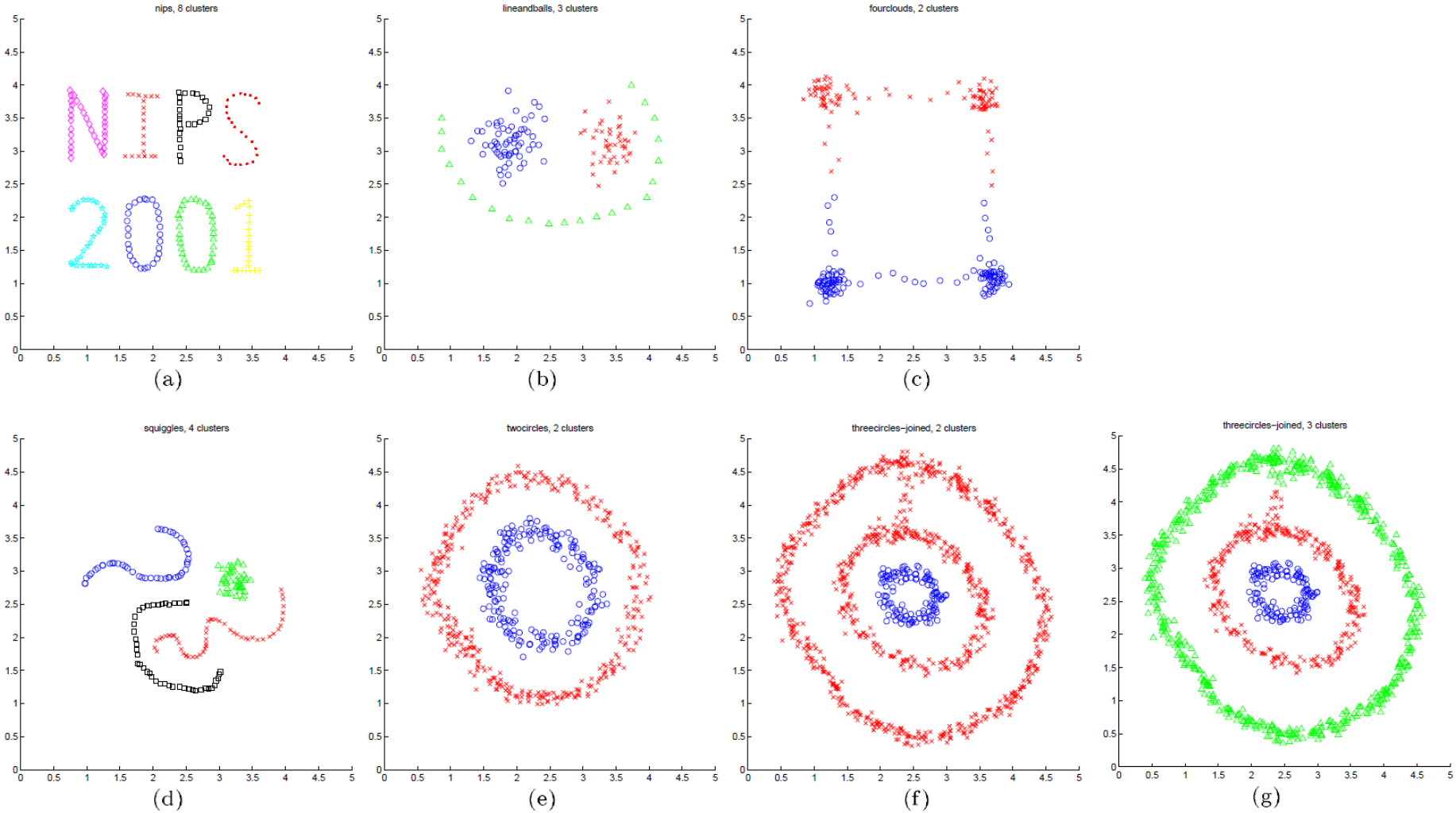
$$\underset{\boldsymbol{L}}{\begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix}} \quad \underset{\boldsymbol{L}_{\text{sym}}}{\begin{pmatrix} 1 & -1/\sqrt{2} & 0 \\ -1/\sqrt{2} & 1 & -1/\sqrt{2} \\ 0 & -1/\sqrt{2} & 1 \end{pmatrix}} \quad \underset{\boldsymbol{L}_{\text{rw}}}{\begin{pmatrix} 1 & -1 & 0 \\ -0.5 & 1 & -0.5 \\ 0 & -1 & 1 \end{pmatrix}}$$

Normalized spectral clustering

- Normalized graph Laplacians have similar spectral properties
- Normalized spectral clustering (using $\boldsymbol{L}_{\text{rw}}$)
  1. Construct similarity graph $\boldsymbol{W}$
  2. Compute its normalized graph Laplacian $\boldsymbol{L}_{\text{rw}}$
  3. Compute the last k eigenvectors $\boldsymbol{u}_n, \ldots, \boldsymbol{u}_{n-k+1}$ of $\boldsymbol{L}_{\text{rw}}$ (i.e., having k smallest eigenvalues)
  4. Construct $n \times k$ matrix $\boldsymbol{U} = \begin{pmatrix} \boldsymbol{u}_n & \boldsymbol{u}_{n-1} & \cdots & \boldsymbol{u}_{n-k+1} \end{pmatrix}$
  5. Cluster the rows of $\boldsymbol{U}$ using k-means
- Normalized spectral clustering is a relaxation of Ncut
- Better behaved from statistical point of view

The normalized spectral clustering algorithm above is often method of choice.

Some examples



Ng et al., 2001

Outline

Some variants of spectral clustering

- p-spectral clustering
  - Use $\sum_{i,j} w_{ij} |x_i - x_j|^p$
  - Standard spectral clustering is obtained for $p = 2$
  - As $p \to 1$, we obtain provably better (Cheeger) cuts
- Constrained spectral clustering
  - Way to incorporate domain knowledge and side information
  - Intuitively, obtained clustering must satisfy some application-defined constraints
  - E.g., must-link and cannot-link constraints (i.e., provide vertex pairs that must or must not end up in same cluster)
  - E.g., partial labels (i.e., provide labels for some vertexes) $\to$ graph-based semi-supervised learning (tutorial)
- Active area of research

Outline

Lessons learned

- Graphs can be represented by matrices (and vice versa)
  - Adjacency matrix
  - Degree matrix
  - Walk matrix
  - Graph Laplacian

- Spectral properties of these matrices relate to properties of the graph

- Spectral clustering
  - Find non-convex clusters in similarity graphs
  - Good clustering $\approx$ good graph cut (RatioCut or Ncut)
  - Related to smallest eigenvectors of graph Laplacian
  - Many useful variants

Literature

- David Skillicorn
  Understanding Complex Datasets: Data Mining with Matrix Decompositions (Chapter 4)
  Chapman and Hall, 2007

- Ulrike von Luxburg
  A Tutorial on Spectral Clustering
  Statistics and Computing, 17(4), 2007
  http://www.kyb.mpg.de/publications/attachments/Luxburg07_tutorial_4488%5B0%5D.pdf

Data Mining and Matrices

08 – Tensors

Prof. Dr. Rainer Gemulla

Universität Mannheim

FSS 2018

Outline

What is a tensor?

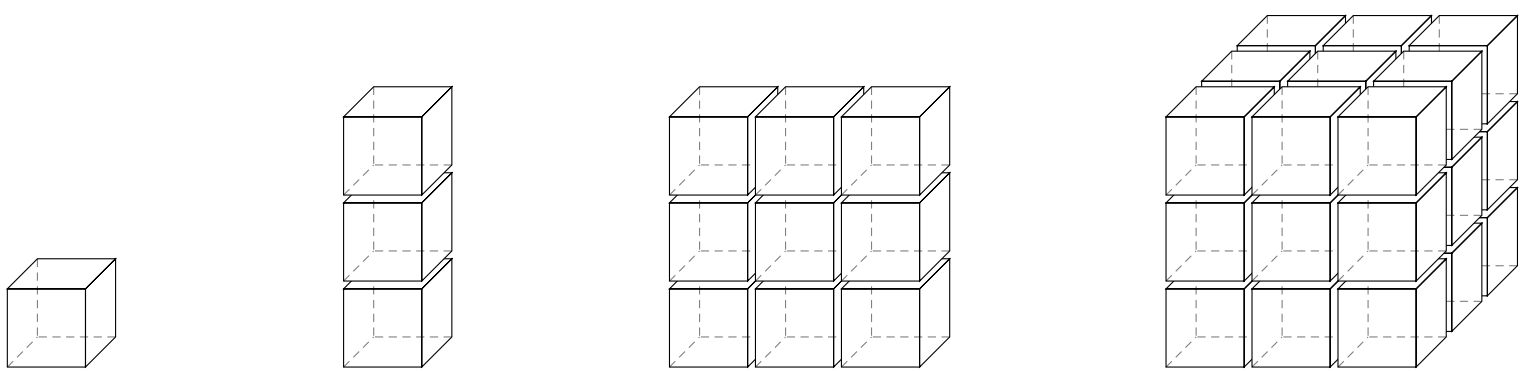
A **tensor** is a multi-way extension of a matrix.

- A multi-dimensional array
- A multi-linear map

Scalars *Vectors* *Matrices* *Stacked matrices* $\cdots$

$$1 \qquad \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \qquad \begin{pmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{pmatrix}$$

$$\left(\begin{pmatrix} 1 & 2 & 5 \\ 2 & 3 & 6 \\ 3 & 4 & 7 \end{pmatrix} \begin{pmatrix} 9 & 9 & 9 \\ 9 & 9 & 9 \\ 9 & 9 & 9 \end{pmatrix}\right)$$

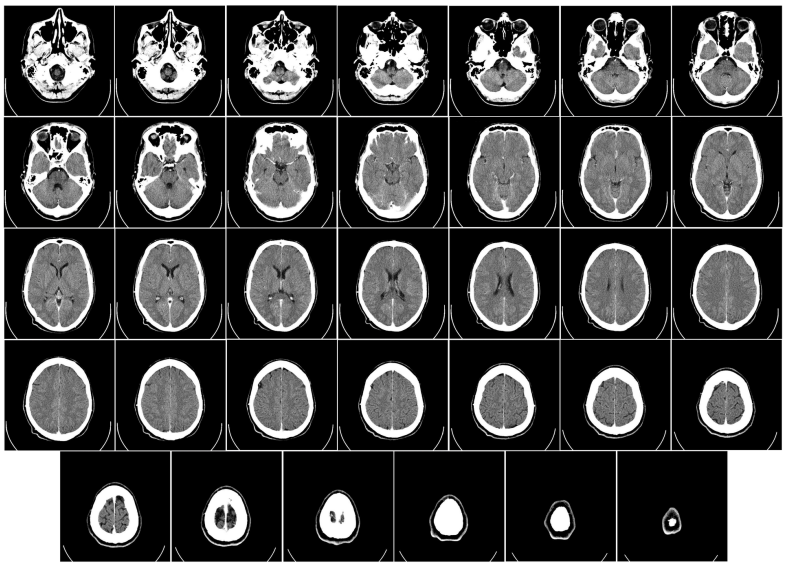


Why tensors?

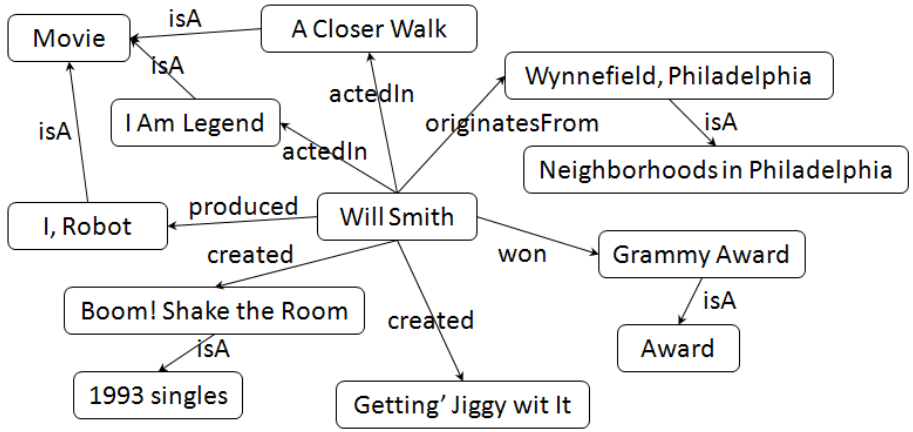
- Tensors can be used when matrices are not enough
- A matrix can represent a binary relation (or function)
  - Entities and attributes
  - Vertices and edges
- A tensor can represent an n-ary relation (or function)
  - A relationship between n sets of "things"
  - E.g. Subject-predicate-object data
- A tensor can represent a set of binary relations (or functions)
  - Vertices and typed edges ("colored")
  - E.g., images (rows and columns) for a set of persons
- A tensor can represent a sequence of binary (or n-ary) relations (or functions)
  - Objects and their attributes over time
  - But: using tensors for time series should be approached with care

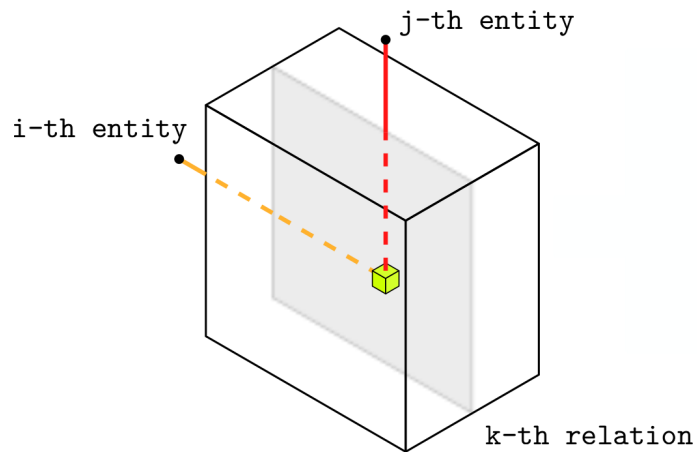
Two examples

CT scans



Knowledge bases





Outline

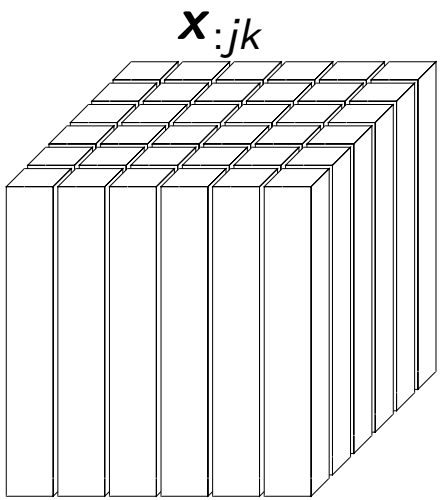
Terminology

- We say a tensor is an **N-way** array
  - E.g., a matrix is a 2-way array
- In the literature, one also finds
  - N-dimensional tensor
    (confusing: is a 3-dimensional vector a 1-dimensional tensor?)
  - rank-N tensor
    (but: we have a different use for the word *rank*)
- A 3-way tensor
  - Has three **modes**: rows, columns, and tubes
  - Can be **I-by-J-by-K-dimensional**
  - Also used: M-by-N-by-K or I_1-by-I_2-by-I_3
- Notation
  - Tensors: $\mathcal{X}$, $\mathcal{Y}$, $\mathcal{Z}$, ...
  - Sizes: I, J, K, ...
  - Indexes: i, j, k, ...

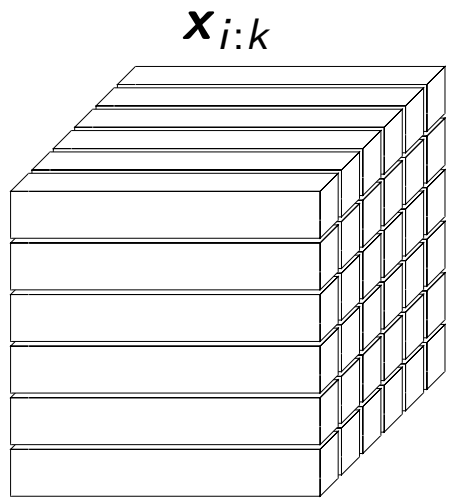
Outline

Fibers

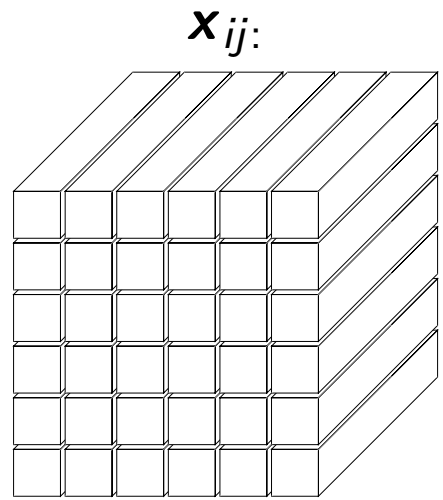
- We can extract subarrays by suitable indexing operations
- If we fix all dimensions, we obtain a scalar (e.g., x_{ijk})
- If we fix all but one dimension, we obtain a **fiber**
  - Convention: fibers are extracted as column vectors



Mode-1 fibers (columns)



Mode-2 fibers (rows)

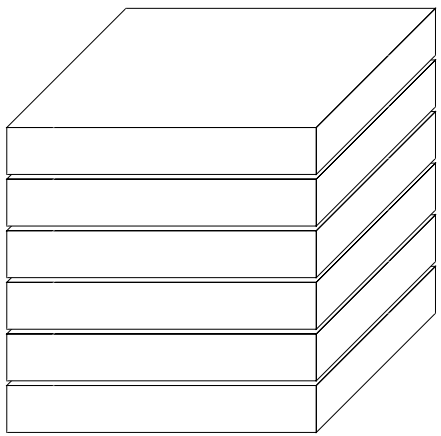


Mode-3 fibers (tubes)

Slices

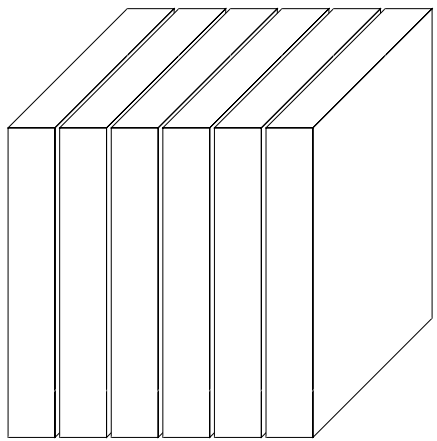
- If we fix all but two dimensions, we obtain a **slice**
  - Convention: slices are extracted as matrices
  - Rows correspond to first omitted dimension, columns to second

$\boldsymbol{X}_{i::}$



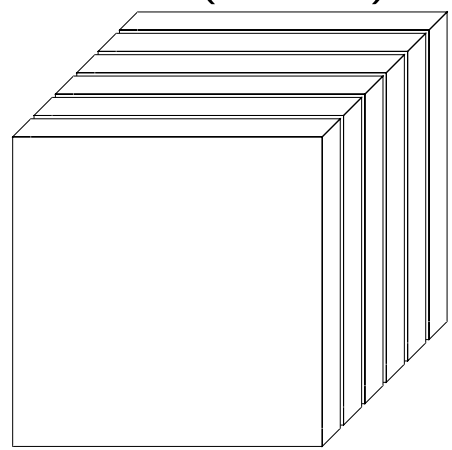
Horizontal slices

$\boldsymbol{X}_{:j:}$



Lateral slices

$\boldsymbol{X}_{::k}$ (or $\boldsymbol{X}_k$)



Frontal slices

Example

$$\mathcal{X} = \begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix} \begin{pmatrix} 5 & 7 \\ 6 & 8 \end{pmatrix} \quad \rightarrow \quad \begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix} \quad \begin{pmatrix} 5 & 7 \\ 6 & 8 \end{pmatrix}$$

- Column $\boldsymbol{x}_{:11} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$, row $\boldsymbol{x}_{1:1} = \begin{pmatrix} 1 \\ 3 \end{pmatrix}$, tube $\boldsymbol{x}_{11:} = \begin{pmatrix} 1 \\ 5 \end{pmatrix}$
- Frontal slice $\boldsymbol{X}_{::1} = \begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix}$
- Lateral slice $\boldsymbol{X}_{:1:} = \begin{pmatrix} 1 & 5 \\ 2 & 6 \end{pmatrix}$
- Horizontal slice $\boldsymbol{X}_{1::} = \begin{pmatrix} 1 & 5 \\ 3 & 7 \end{pmatrix}$

Outline

Matricization

- We can often express tensor operations using suitable matrices
  - Allows use of standard linear algebra packages
  - We'll repeatedly need this, so we discuss it first
  - In particular: matricization, Kroenecker product, Khatri-Rao product, Hadamard product

- Tensor matricization (*flattening*, *unfolding*) unfolds an N-way tensor into a matrix
  - **Mode-n matricization** $\boldsymbol{X}_{(n)}$ arranges the mode-n fibers as columns of a matrix
  - As many rows as size of n-th mode
  - As many columns as the product of the sizes of the other modes
  - Cumbersome to express formally: If $\boldsymbol{\mathcal{X}}$ is an $I_1 \times I_2 \times \ldots \times I_n$ tensor, then $\boldsymbol{X}_{(n)}$ contains $x_{i_1 i_2 \ldots i_N}$ at position (i_n, j), where

$$j = 1 + \sum_{k=1}^{N} (i_k - 1) J_k [k \neq n] \quad \text{with} \quad J_k = \prod_{m=1}^{k-1} I_m [m \neq n]$$

  - Different authors may use different orders (!)

Example

$$\mathcal{X} = \left(\begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix} \begin{pmatrix} 5 & 7 \\ 6 & 8 \end{pmatrix}\right) \quad \rightarrow \quad \underbrace{\begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix}}_{\boldsymbol{X}_1} \quad \underbrace{\begin{pmatrix} 5 & 7 \\ 6 & 8 \end{pmatrix}}_{\boldsymbol{X}_2}$$

- Column $\boldsymbol{x}_{:11} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$, row $\boldsymbol{x}_{1:1} = \begin{pmatrix} 1 \\ 3 \end{pmatrix}$, tube $\boldsymbol{x}_{:11} = \begin{pmatrix} 1 \\ 5 \end{pmatrix}$
- $\boldsymbol{X}_{(1)} = \begin{pmatrix} 1 & 3 & 5 & 7 \\ 2 & 4 & 6 & 8 \end{pmatrix} = (\boldsymbol{X}_1 \quad \boldsymbol{X}_2)$
- $\boldsymbol{X}_{(2)} = \begin{pmatrix} 1 & 2 & 5 & 6 \\ 3 & 4 & 7 & 8 \end{pmatrix}$
- $\boldsymbol{X}_{(3)} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{pmatrix}$

Kronecker matrix product

- Rescales one matrix with each element of another matrix
- $m_1 \times n_1$ and $m_2 \times n_2$ matrices give $m_1 m_2 \times n_1 n_2$ matrix

$$\boldsymbol{A} \otimes \boldsymbol{B} = \begin{pmatrix} a_{11}\boldsymbol{B} & a_{12}\boldsymbol{B} & \cdots & a_{1n_1}\boldsymbol{B} \\ a_{21}\boldsymbol{B} & a_{22}\boldsymbol{B} & \cdots & a_{2n_1}\boldsymbol{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m_1 1}\boldsymbol{B} & a_{m_1 2}\boldsymbol{B} & \cdots & a_{m_1 n_1}\boldsymbol{B} \end{pmatrix}$$

- For example,

$$\begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix} \otimes \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 3 & 3 & 3 \\ 1 & 1 & 1 & 3 & 3 & 3 \\ 2 & 2 & 2 & 4 & 4 & 4 \\ 2 & 2 & 2 & 4 & 4 & 4 \end{pmatrix}$$

Khatri-Rao matrix product

- Column-wise Kroenecker product
  - Rescales the columns of one matrix with each element in the respective column of another matrix
  - Number of columns must match
- $m_1 \times n$ and $m_2 \times n$ matrices give $m_1 m_2 \times n$ matrix

$$\begin{aligned}\boldsymbol{A} \odot \boldsymbol{B} &= \begin{pmatrix} \boldsymbol{a}_1 \otimes \boldsymbol{b}_1 & \boldsymbol{a}_2 \otimes \boldsymbol{b}_2 & \cdots & \boldsymbol{a}_n \otimes \boldsymbol{b}_n \end{pmatrix} \\ &= \begin{pmatrix} a_{11}\boldsymbol{b}_1 & a_{12}\boldsymbol{b}_2 & \cdots & a_{1n}\boldsymbol{b}_n \\ a_{21}\boldsymbol{b}_1 & a_{22}\boldsymbol{b}_2 & \cdots & a_{2n}\boldsymbol{b}_n \\ \vdots & \vdots & \ddots & \vdots \\ a_{m_1 1}\boldsymbol{b}_1 & a_{m_1 2}\boldsymbol{b}_2 & \cdots & a_{m_1 n}\boldsymbol{b}_n \end{pmatrix}\end{aligned}$$

- For example,

$$\begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix} \odot \begin{pmatrix} 1 & 100 \\ 10 & 1000 \end{pmatrix} = \begin{pmatrix} 1 & 300 \\ 10 & 3000 \\ 2 & 400 \\ 20 & 4000 \end{pmatrix}$$

Hadamard matrix product

- Element-wise product
  - Rescales every entry of one matrix with the corresponding entry of another matrix
  - Number of rows and columns must match
- Two $m \times n$ matrices give $m \times n$ matrix

$$\boldsymbol{A} * \boldsymbol{B} = \begin{pmatrix} a_{11} b_{11} & a_{12} b_{12} & \cdots & a_{1n} b_{1n} \\ a_{21} b_{21} & a_{22} b_{22} & \cdots & a_{2n} b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} b_{m1} & a_{m2} b_{m2} & \cdots & a_{mn} b_{mn} \end{pmatrix}$$

- For example,

$$\begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix} * \begin{pmatrix} 1 & 100 \\ 10 & 1000 \end{pmatrix} = \begin{pmatrix} 1 & 300 \\ 20 & 4000 \end{pmatrix}$$

Some identities

$$
\begin{aligned}
(\boldsymbol{A} \otimes \boldsymbol{B})(\boldsymbol{C} \otimes \boldsymbol{D}) &= \boldsymbol{AC} \otimes \boldsymbol{BD} \\
(\boldsymbol{A} \otimes \boldsymbol{B})^\dagger &= \boldsymbol{A}^\dagger \otimes \boldsymbol{B}^\dagger \\
\boldsymbol{A} \odot \boldsymbol{B} \odot \boldsymbol{C} &= (\boldsymbol{A} \odot \boldsymbol{B}) \odot \boldsymbol{C} = \boldsymbol{A} \odot (\boldsymbol{B} \odot \boldsymbol{C}) \\
(\boldsymbol{A} \odot \boldsymbol{B})^T (\boldsymbol{A} \odot \boldsymbol{B}) &= \boldsymbol{A}^T \boldsymbol{A} * \boldsymbol{B}^T \boldsymbol{B} \\
(\boldsymbol{A} \odot \boldsymbol{B})^\dagger &= (\boldsymbol{A}^T \boldsymbol{A} * \boldsymbol{B}^T \boldsymbol{B})^\dagger (\boldsymbol{A} \odot \boldsymbol{B})^T
\end{aligned}
$$

- Here $\boldsymbol{A}^\dagger$ denotes the pseudo-inverse of $\boldsymbol{A}$
- Using these equations may save considerable space and time; e.g.
  - Let $\boldsymbol{A}, \boldsymbol{B}$ be $n \times k$ matrices with $n \gg k$
  - $\boldsymbol{A} \odot \boldsymbol{B}$ is an $n^2 \times k$ matrix
  - $(\boldsymbol{A} \odot \boldsymbol{B})^T (\boldsymbol{A} \odot \boldsymbol{B})$, $\boldsymbol{A}^T \boldsymbol{A}$, $\boldsymbol{B}^T \boldsymbol{B}$ and $\boldsymbol{A}^T \boldsymbol{A} * \boldsymbol{B}^T \boldsymbol{B}$ are $k \times k$ matrices

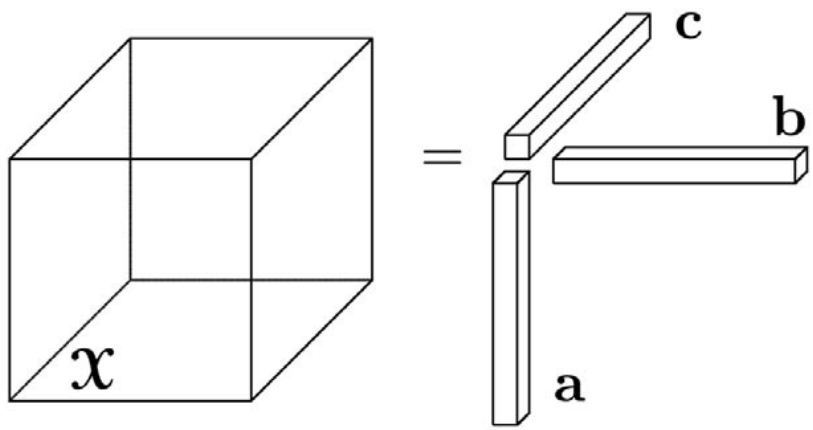
Outline

Outline

Outer product

- Tensors require extensions to the standard linear algebra operations for matrices
- We'll discuss those in turn whenever needed
- A **multi-way vector outer product** is a tensor where each element is the product of corresponding elements in the vectors

$$\boldsymbol{\mathcal{X}} = \boldsymbol{a} \circ \boldsymbol{b} \circ \boldsymbol{c} \qquad \Longleftrightarrow \qquad x_{ijk} = a_i b_j c_k$$



- Observe: $\boldsymbol{a} \circ \boldsymbol{b} = \boldsymbol{a}\boldsymbol{b}^T$

Example

- $\boldsymbol{a} = \begin{pmatrix} 1 & 2 & 3 \end{pmatrix}^T$
- $\boldsymbol{b} = \begin{pmatrix} 1 & 2 & 4 \end{pmatrix}^T$
- $\boldsymbol{c} = \begin{pmatrix} 1 & 10 \end{pmatrix}^T$
- $\boldsymbol{\mathcal{X}} = \boldsymbol{a} \circ \boldsymbol{b} \circ \boldsymbol{c}$ is a $3 \times 3 \times 2$ tensor
- The frontal slices are then

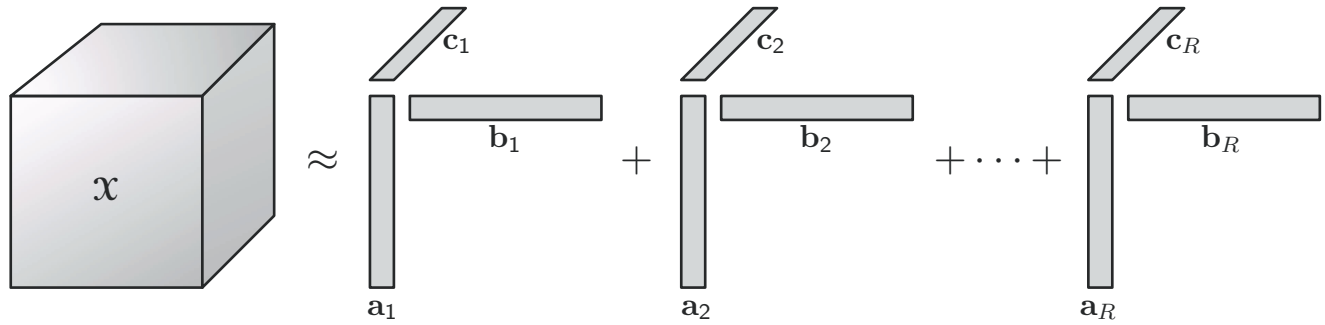
$$\boldsymbol{X}_1 = (\boldsymbol{a} \circ \boldsymbol{b})c_1 = (\boldsymbol{a}\boldsymbol{b}^T)c_1 = \begin{pmatrix} 1 & 2 & 4 \\ 2 & 4 & 8 \\ 3 & 6 & 12 \end{pmatrix}$$

$$\boldsymbol{X}_2 = (\boldsymbol{a} \circ \boldsymbol{b})c_2 = (\boldsymbol{a}\boldsymbol{b}^T)c_2 = \begin{pmatrix} 10 & 20 & 40 \\ 20 & 40 & 80 \\ 30 & 60 & 120 \end{pmatrix}$$

Rank-1 tensors

- A matrix decomposition represents the given matrix as the product of two (or more) **factor matrices**
- The *rank* of a matrix $\boldsymbol{M}$ is the
  - Number of linearly independent rows (*row rank*)
  - Number of linearly independent columns (*column rank*)
  - Minimum number of rank-1 matrices needed to be summed up to get $\boldsymbol{M}$ (*Schein rank*)
  - All definitions are equivalent
- Let's first look at rank-1 tensors
  - A rank-1 matrix is an outer product of two vectors
  - We define: An N-way tensor is a **rank-1 tensor** if it can be written as an outer product of N vectors
  - E.g., the tensor on the previous slide

The CP decomposition



- The **CP decomposition** approximates $\mathcal{X}$ by summing up R rank-1 tensors
- R is called the **size** of the CP decomposition
- Can also be written using N factor matrices: $\mathcal{X} = [\![\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!]$
  - For each mode, create a matrix with the corresponding vectors of each rank-1 component as columns
  - E.g., $\boldsymbol{A} = \begin{pmatrix} \boldsymbol{a}_1 & \boldsymbol{a}_2 & \cdots & \boldsymbol{a}_R \end{pmatrix}$
  - We have $x_{ijk} \approx \sum_{r=1}^{R} a_{ir} b_{jr} c_{kr}$

The many names of the CP decomposition

Name	Proposed by
Polyadic Form of a Tensor	Hitchcock, 1927
PARAFAC (Parallel Factors)	Harshman, 1970
CANDECOMP or CAND (Canonical decomposition)	Carroll and Chang, 1970
Topographic Components Model	Möcks, 1988
CP (CANDECOMP/PARAFAC)	Kiers, 2000

- Chemometrics, psychometrics, phonetics, sensor data processing, telecommunications, neuroscience, medical data, text mining, social network analysis, image compression, computer vision, ...

Example: TOPHITS

- Builds a web pages-by-web pages-by-anchor text tensor to study the link structure and link topics of web pages
  - w_{ijk} is the number of times page i links to page j using term k
- The CP decomposition of this tensor behaves akin to HITS
  - In size-1 CP, $\boldsymbol{a}$ gives hub scores, $\boldsymbol{b}$ gives authority scores, and $\boldsymbol{c}$ gives term weights
  - In size-R CP, the data is split up into multiple topics, each with its own hubs, authorities, and term weights
  - Helps to avoid topic drift, not restricted to focused subgraphs
- Used a higher-order power method to compute the CP

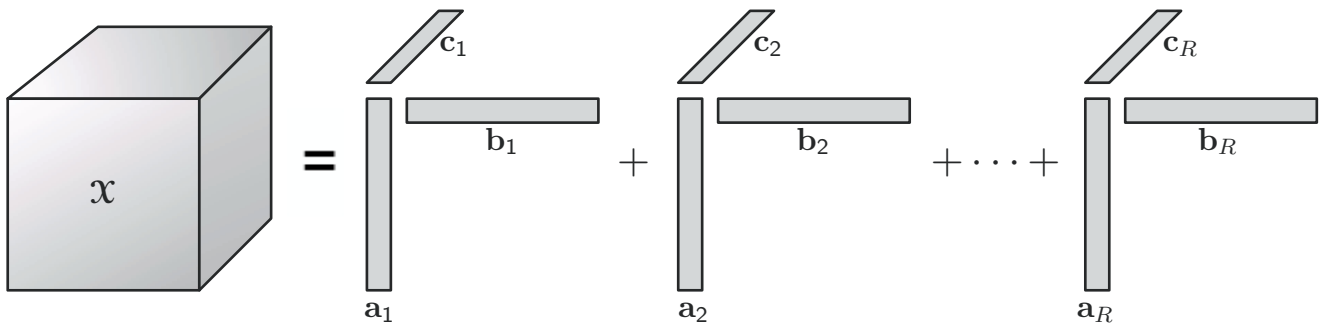
Topics		**Authorities**	
SCORE	TERM	SCORE	HOST
1st Principal Factor			
0.23	java	0.86	`java.sun.com`
0.18	sun	0.38	`developers.sun.com`
0.17	platform	0.16	`docs.sun.com`
0.16	solaris	0.14	`see.sun.com`
0.16	developer	0.14	`www.sun.com`
0.15	edition	0.09	`www.samag.com`
0.15	download	0.07	`developer.sun.com`
0.14	info	0.06	`sunsolve.sun.com`
0.12	software	0.05	`access1.sun.com`
		0.05	`iforce.sun.com`

Kolda et al., 2005

Outline

Tensor rank

- The **rank** of a tensor is the minimum number of rank-1 tensors needed to represent the tensor exactly
  - Given by the exact CP decomposition of size $R = \text{rank}(\mathcal{X})$, also called **rank decomposition**
  - Generalizes the notion of Schein rank for matrices



- Tensors behave differently than matrices
  - Much more complicated on the one hand
  - (Some) nicer properties on the other hand

Tensor rank oddities (1)

- The rank of a (real-valued) tensor can be different over reals and over complex numbers.

Tensor	*Over* $\mathbb{R}$	*Over* $\mathbb{C}$
$\boldsymbol{X}_1 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$	$\boldsymbol{A} = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & -1 \end{pmatrix}$	$\boldsymbol{A} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ -i & i \end{pmatrix}$
$\boldsymbol{X}_2 = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$	$\boldsymbol{B} = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}$	$\boldsymbol{B} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ i & -i \end{pmatrix}$
	$\boldsymbol{C} = \begin{pmatrix} 1 & 1 & 0 \\ -1 & 1 & 1 \end{pmatrix}$	$\boldsymbol{C} = \begin{pmatrix} 1 & 1 \\ -i & i \end{pmatrix}$

- Determining the rank is NP-hard
  - No "straightforward" algorithm known
- Over reals, the rank can be **larger** than the largest dimension
  - $\operatorname{rank}(\boldsymbol{\mathcal{X}}) \leq \min\{IJ, IK, JK\}$ ofr $I \times J \times K$ tensor
  - Better bounds known only for few special cases

Tensor rank oddities (2)

- For matrices, the Eckart-Young theorem tells us that
  - We can determine a best low-rank approximation
  - Factors of best rank-k approximation are part of best higher-rank approximations
- There are tensors of rank R that can be approximated *arbitrarily well* with tensors of rank $R' < R$
  - Such tensors are called **degenerate**
  - There are no *best* low-rank approximations for degenerate tensors
  - The smallest such R' is called the **border rank** of the tensor
- There are tensors for which the factors of the best rank-one approximation are not part of the best rank-2 approximation
  - Cannot find factors sequentially (but must find them simultaneously)

Tensor rank oddities (3)

- The rank decomposition is often *essentially unique*
  - Finally some good news!
  - Not true for matrices (SVD is essentially unique only due to its additional constraints)
- Essentially unique = only scaling and permutations allowed
- Holds under "mild conditions"

Outline

How to compute the CP decomposition?

- Here we discuss the alternating least squares (ALS) approach
  - Popular method
  - Simple to understand and implement
  - Not guaranteed to converge to minimum or stationary point
  - Objective merely decreases over iterations (until converged)
  - Can take many iterations until convergence
  - Final solution heavily dependent on starting point
- Many alternative methods have been proposed (and are being proposed)

ALS for CP

- Our goal is to optimize $\min_{\boldsymbol{A},\boldsymbol{B},\boldsymbol{C}} \|\boldsymbol{\mathcal{X}} - [\![\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!]\|_F$
- Method is as before
  1. Fix $\boldsymbol{B}$ and $\boldsymbol{C}$ and solve for $\boldsymbol{A}$
  2. Solve for $\boldsymbol{B}$, then solve for $\boldsymbol{C}$
  3. Repeat until convergence
- To implement this method, we rewrite the CP decomposition

$$\boldsymbol{X}_{(1)} \approx \boldsymbol{A}(\boldsymbol{C} \odot \boldsymbol{B})^T$$
$$\boldsymbol{X}_{(2)} \approx \boldsymbol{B}(\boldsymbol{C} \odot \boldsymbol{A})^T$$
$$\boldsymbol{X}_{(3)} \approx \boldsymbol{C}(\boldsymbol{B} \odot \boldsymbol{A})^T$$

- Now we can "read off" the solution; e.g.

$$\min_{\boldsymbol{A}} \|\boldsymbol{X}_{(1)} - \boldsymbol{A}(\boldsymbol{C} \odot \boldsymbol{B})^T\|_F$$
$$\boldsymbol{A} = \boldsymbol{X}_{(1)}(\underbrace{(\boldsymbol{C} \odot \boldsymbol{B})^T}_{R \times JK})^\dagger$$
$$\boldsymbol{A} = \boldsymbol{X}_{(1)}(\boldsymbol{C} \odot \boldsymbol{B})(\underbrace{\boldsymbol{C}^T\boldsymbol{C} * \boldsymbol{B}^T\boldsymbol{B}}_{R \times R})^\dagger$$

Outline

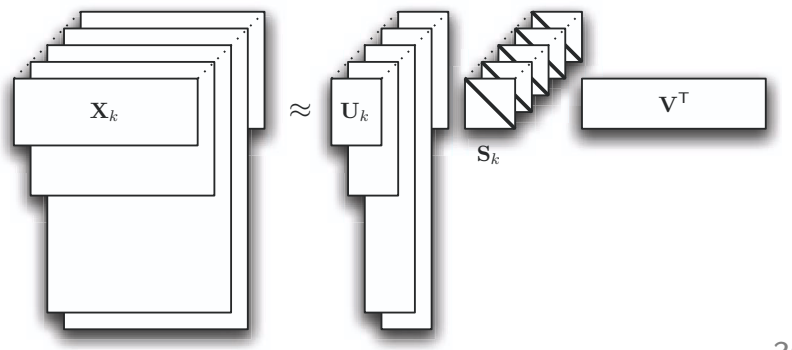
Variants of the CP decomposition

Many variants of the CP decomposition have been proposed; e.g.,

- Explicit modeling of component weights; e.g.,

$$\boldsymbol{\mathcal{X}} \approx [\![\boldsymbol{\lambda}; \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!] = \sum_{r=1}^{R} \lambda_r \, \boldsymbol{a}_r \circ \boldsymbol{b}_r \circ \boldsymbol{c}_r,$$

  where $\boldsymbol{\lambda} \in \mathbb{R}^R$ and the columns of $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$ have unit norm
- Non-negative CP decomposition (NNCP)
- Tensor completion (to handle missing entries)
- PARAFAC2 decomposition: jointly factor K matrices such that $\boldsymbol{X}_k \approx \boldsymbol{U}_k \boldsymbol{S}_k \boldsymbol{V}^T$, where $\boldsymbol{V}^T$ is shared for all matrices (*shared subspace learning*)
- The INDSCAL decomposition
- ...



The INDSCAL decomposition (definition)

- Recall: CP decomposition decomposes a 3-way tensor $\boldsymbol{\mathcal{X}}$ using three factor matrices $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$
  - $\boldsymbol{\mathcal{X}} \approx [\![\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!]$
  - Or element-wise: $x_{ijk} \approx \sum_{r=1}^{R} a_{ir} b_{jr} c_{kr}$
- The **INDSCAL decomposition** decomposes a 3-way tensor $\boldsymbol{\mathcal{X}}$ into two factor matrices $\boldsymbol{A}$ and $\boldsymbol{C}$
  - Matrix $\boldsymbol{A}$ is used for first two modes
  - $\boldsymbol{\mathcal{X}} \approx [\![\boldsymbol{A}, \boldsymbol{A}, \boldsymbol{C}]\!]$
  - Or element-wise: $x_{ijk} \approx \sum_{r=1}^{R} a_{ir} a_{jr} c_{kr}$
  - $\boldsymbol{\mathcal{X}}$ must be an $I \times I \times K$ tensor, i.e., the first two modes must have the same dimensionality
- Observe that reconstructed frontal slices are symmetric (i.e., $[\![\boldsymbol{A}, \boldsymbol{A}, \boldsymbol{C}]\!]_{ijk} = [\![\boldsymbol{A}, \boldsymbol{A}, \boldsymbol{C}]\!]_{jik}$)
  - If we know that the frontal slices of $\boldsymbol{\mathcal{X}}$ are symmetric, INDSCAL won't destroy this structure

Carroll & Chan, 1970

The INDSCAL decomposition (why?)

- INDSCAL stands for "Individual Differences in Scaling"
  - Useful to analyze multiple symmetric matrices referring to the same objects
  - Distance, similarity, covariance, Laplacian matrices, ...
  - Recently also applied to knowledge bases (under the name DistMult)
- Assume K subjects each ranked the similarity of I objects
  - This gives us K similarity matrices, each $I \times I$
  - INDSCAL assumes that similarity decisions for each subject depends on a set of latent factors ($\boldsymbol{A}$), and that those factors are weighted differently by different subjects ($\boldsymbol{C}$)
- Best method for computing INDSCAL still open; common hack:
  - Compute normal CP and hope that $\boldsymbol{A}$ and $\boldsymbol{B}$ converge
  - Then force $\boldsymbol{A}$ and $\boldsymbol{B}$ equal and update $\boldsymbol{C}$ one more time

Psychological data: "How similar are these countries?"

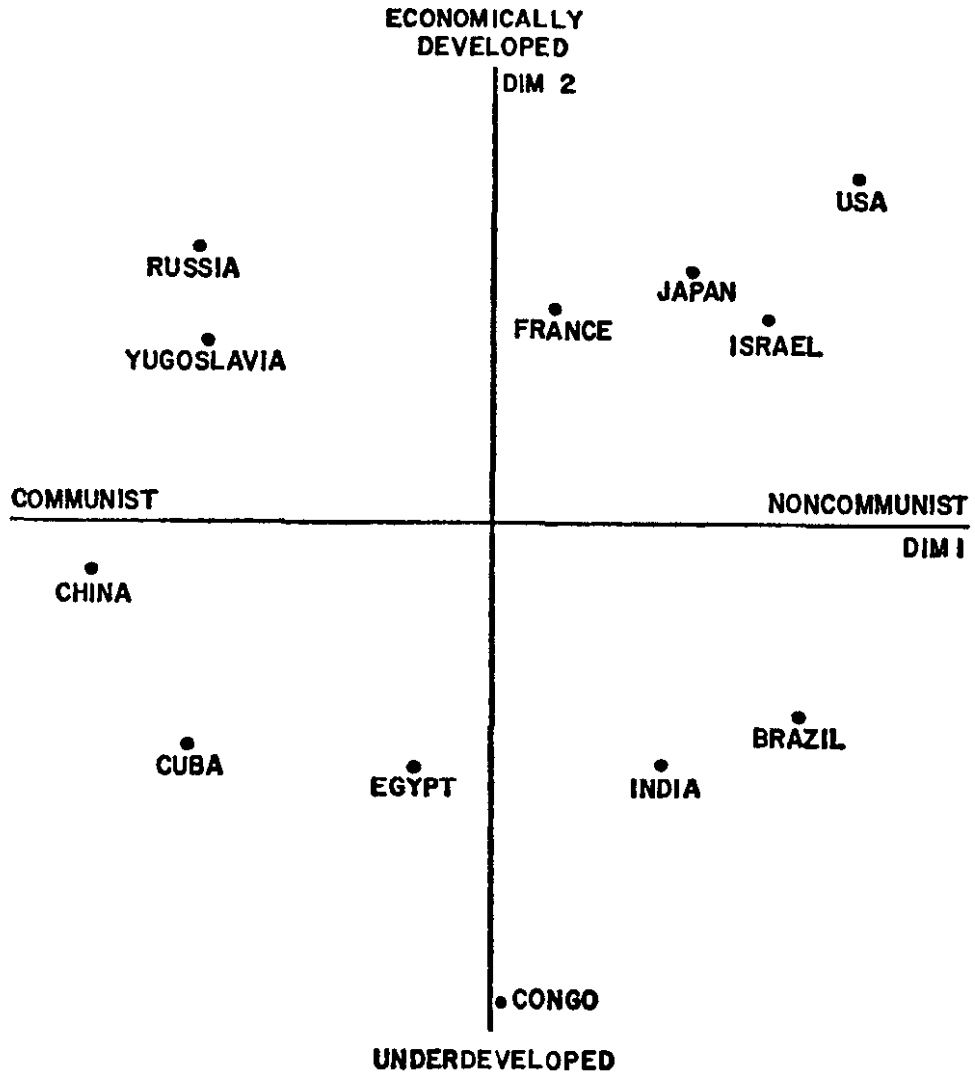


FIGURE 11

Carroll & Chan, 1970

Psychological data: "How similar are these countries?"

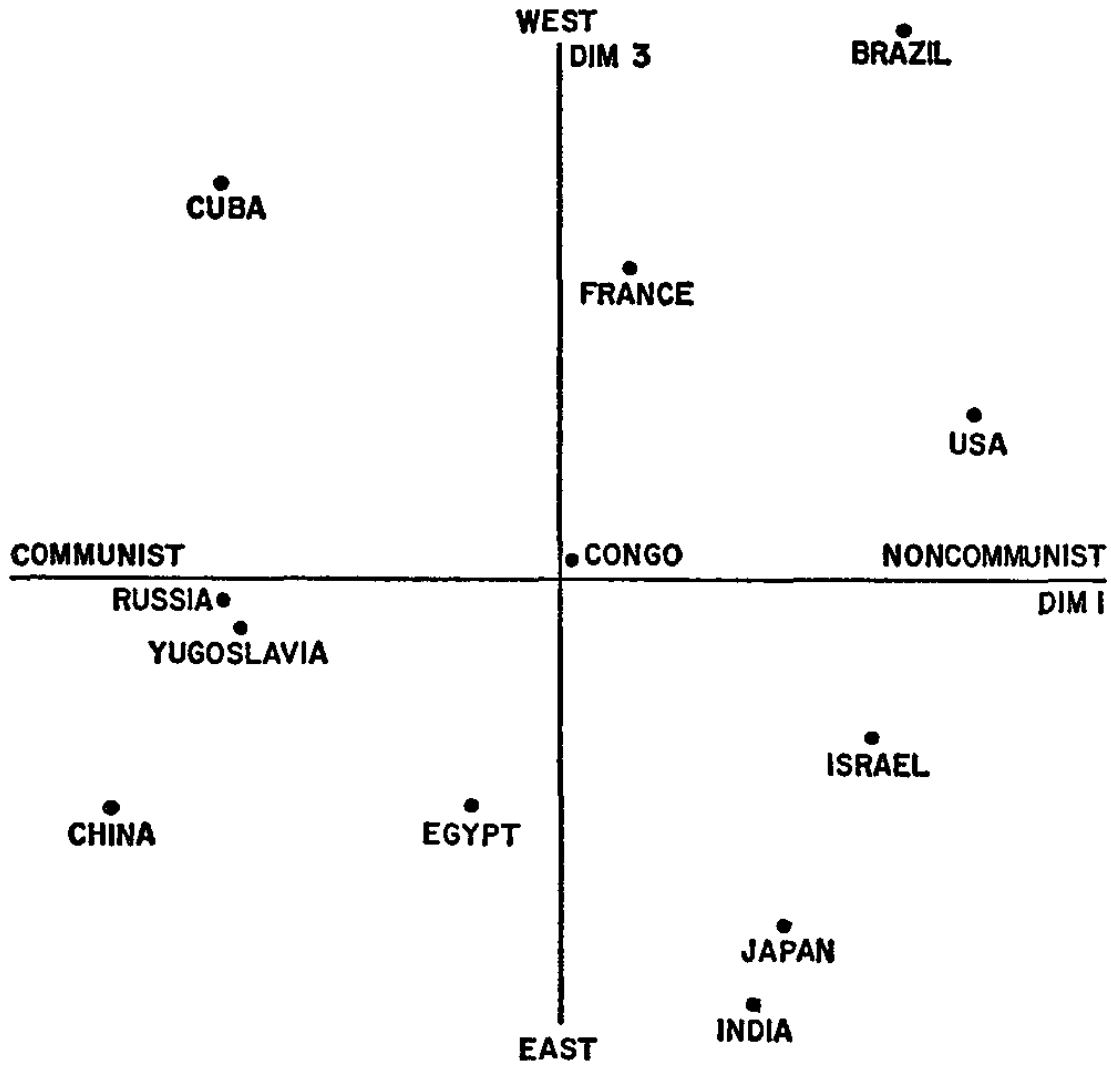


FIGURE 12

Psychological data: "How similar are these countries?"

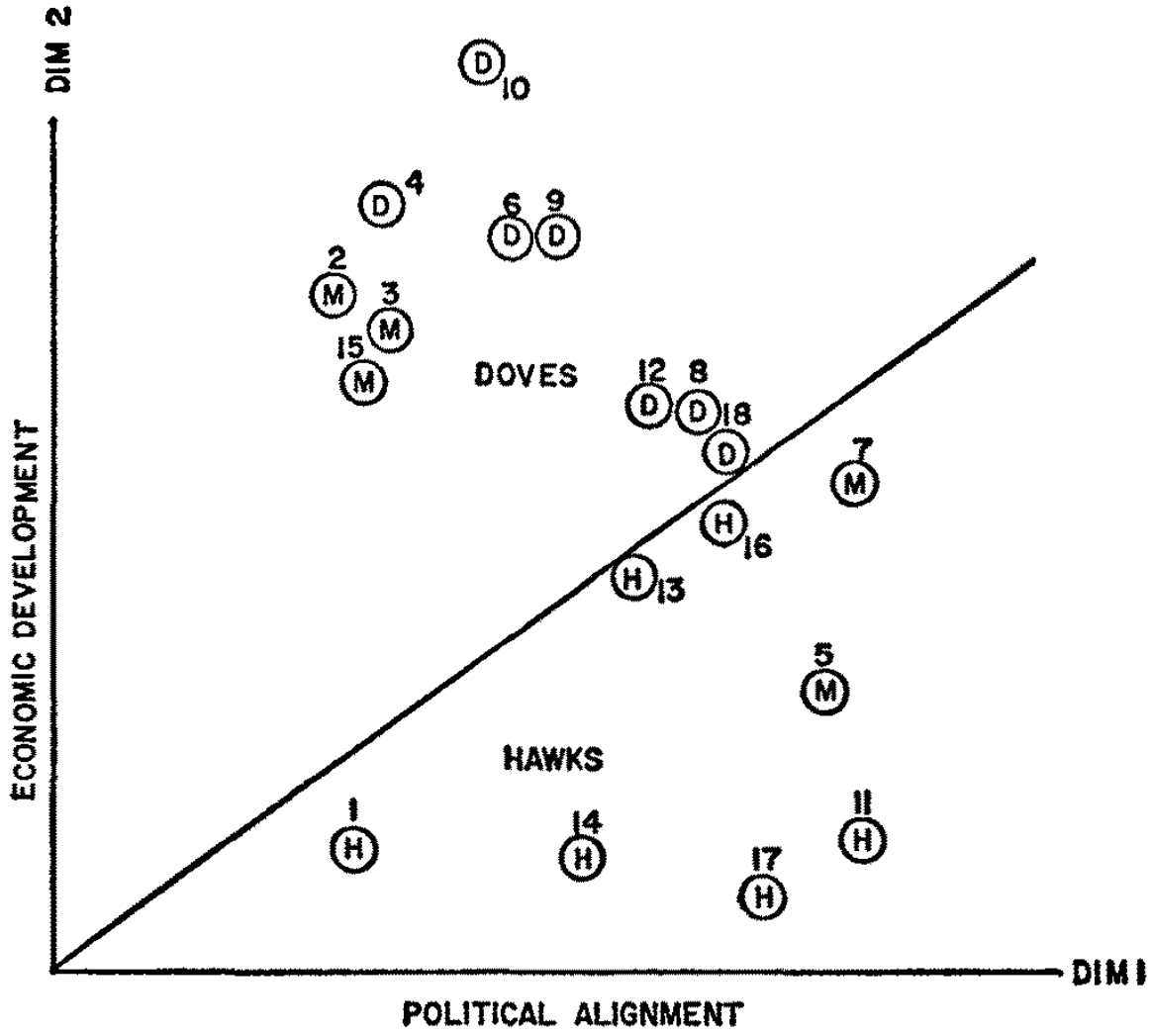


FIGURE 13

Carroll & Chan, 1970

Outline

Outline

Tensor n-mode product

- Let $\mathcal{X}$ be an N-way tensor of size $I_1 \times I_2 \times \cdots \times I_N$, $\boldsymbol{U}$ a matrix of size $J \times I_n$, and $\boldsymbol{v}$ vector of size I_n
- The **n-mode vector product** $\mathcal{X} \times_n \boldsymbol{v}$ of $\mathcal{X}$ with $\boldsymbol{v}$ computes the inner product of each mode-n fiber with $\boldsymbol{v}$
  - Result is tensor of order $N-1$; size $I_1 \times \cdots \times I_{n-1} \times I_{n+1} \times \cdots \times I_N$
  - Element-wise: $[\mathcal{X} \times_n \boldsymbol{v}]_{i_1 \cdots i_{n-1} i_{n+1} \cdots i_N} = \sum_{i_n=1}^{I_n} x_{i_1 i_2 \cdots i_N} v_{i_n}$
- The **n-mode (matrix) product** $\mathcal{X} \times_n \boldsymbol{U}$ of $\mathcal{X}$ with $\boldsymbol{U}$ multiplies each mode-n fiber with $\boldsymbol{U}$ (from the left)
  - Result is tensor of order N; size $I_1 \times \cdots \times I_{n-1} \times J \times I_{n+1} \times \cdots \times I_N$
  - In terms of unfolding: $\mathcal{Y} = \mathcal{X} \times_n \boldsymbol{U} \iff \boldsymbol{Y}_{(n)} = \boldsymbol{U}\boldsymbol{X}_{(n)}$
  - Or element-wise: $[\mathcal{X} \times_n \boldsymbol{U}]_{i_1 \cdots i_{n-1} j i_{n+1} \cdots i_N} = \sum_{i_n=1}^{I_n} x_{i_1 i_2 \cdots i_N} u_{j i_n}$

Example

$$\mathcal{X} = \begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix} \begin{pmatrix} 5 & 7 \\ 6 & 8 \end{pmatrix} \quad \rightarrow \quad \underbrace{\begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix}}_{\boldsymbol{X}_1} \quad \underbrace{\begin{pmatrix} 5 & 7 \\ 6 & 8 \end{pmatrix}}_{\boldsymbol{X}_2}$$

- $\mathcal{X} \times_3 \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 6 & 10 \\ 8 & 12 \end{pmatrix}$ $\quad \mathcal{X} \times_3 \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 \begin{pmatrix} 5 & 7 \\ 6 & 8 \end{pmatrix} \begin{pmatrix} 6 & 10 \\ 8 & 12 \end{pmatrix} \\ 2 \begin{pmatrix} 1 & 3 \\ 6 & 8 \end{pmatrix} \end{pmatrix}$
- $\mathcal{X} \times_1 \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 3 & 11 \\ 7 & 15 \end{pmatrix}$ $\quad \mathcal{X} \times_1 \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} \begin{pmatrix} 5 & 7 \\ 6 & 8 \end{pmatrix} \\ 2 \begin{pmatrix} 1 & 3 \\ 11 & 15 \end{pmatrix} \\ 3 \quad 7 \end{pmatrix}$

Rewriting the CP decomposition

- Useful identities
  - $\boldsymbol{\mathcal{X}} \times_n \boldsymbol{I}_{I_n} = \boldsymbol{\mathcal{X}}$
  - $\boldsymbol{\mathcal{X}} \times_m \boldsymbol{A} \times_n \boldsymbol{B} = \boldsymbol{\mathcal{X}} \times_n \boldsymbol{B} \times_m \boldsymbol{A}$ for $m \neq n$
  - $\boldsymbol{\mathcal{X}} \times_n \boldsymbol{A} \times_n \boldsymbol{B} = \boldsymbol{\mathcal{X}} \times_n (\boldsymbol{B}\boldsymbol{A})$
- We can rewrite a size-R CP decomposition as

$$[\![\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!] = \mathcal{I}_R \times_1 \boldsymbol{A} \times_2 \boldsymbol{B} \times_3 \boldsymbol{C}$$

  - $\mathcal{I}_R$ is the $R \times R \times R$ **identity tensor** (superdiagonal, 1s on the superdiagonal)
- What happens if we use another tensor instead of $\mathcal{I}_R$?

Outline

The Tucker decomposition

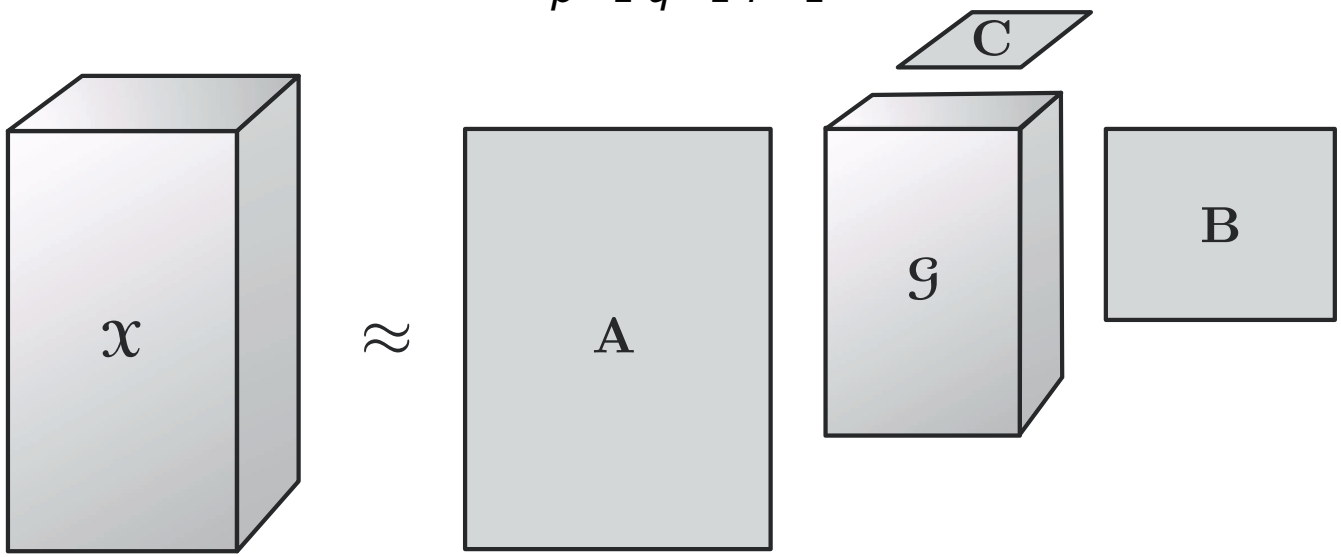
- The **Tucker3 decomposition** decomposes a 3-way tensor as

$$\mathcal{X} \approx \mathcal{G} \times_1 \boldsymbol{A} \times_2 \boldsymbol{B} \times_3 \boldsymbol{C}$$

  - $\mathcal{X}$ is $I \times J \times K$
  - $\mathcal{G}$ is a $P \times Q \times R$ tensor, called the **core tensor**
  - $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$ are $I \times P$, $J \times Q$, and $K \times R$, resp.

- We write $[\![\mathcal{G}; \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!]$ for short and obtain

$$\mathcal{X} \approx [\![\mathcal{G}; \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!] = \sum_{p=1}^{P} \sum_{q=1}^{Q} \sum_{r=1}^{R} g_{pqr}\, \boldsymbol{a}_p \circ \boldsymbol{b}_q \circ \boldsymbol{c}_r$$



Tucker, 1966

Some remarks

- Core tensor can be thought of a compressed version of $\boldsymbol{\mathcal{X}}$
- Many degrees of freedom
  - Parameters P, Q, and R
  - Properties of $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$ (often taken to be orthogonal)
  - CP decomposition is special case ($\boldsymbol{\mathcal{G}}$ superdiagonal, $P = Q = R$)
- Not unique

$$[\![\boldsymbol{\mathcal{G}}; \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!] = [\![\boldsymbol{\mathcal{G}} \times_1 \boldsymbol{U} \times_2 \boldsymbol{V} \times_3 \boldsymbol{W}; \boldsymbol{A}\boldsymbol{U}^{-1}, \boldsymbol{B}\boldsymbol{V}^{-1}, \boldsymbol{C}\boldsymbol{W}^{-1}]\!]$$

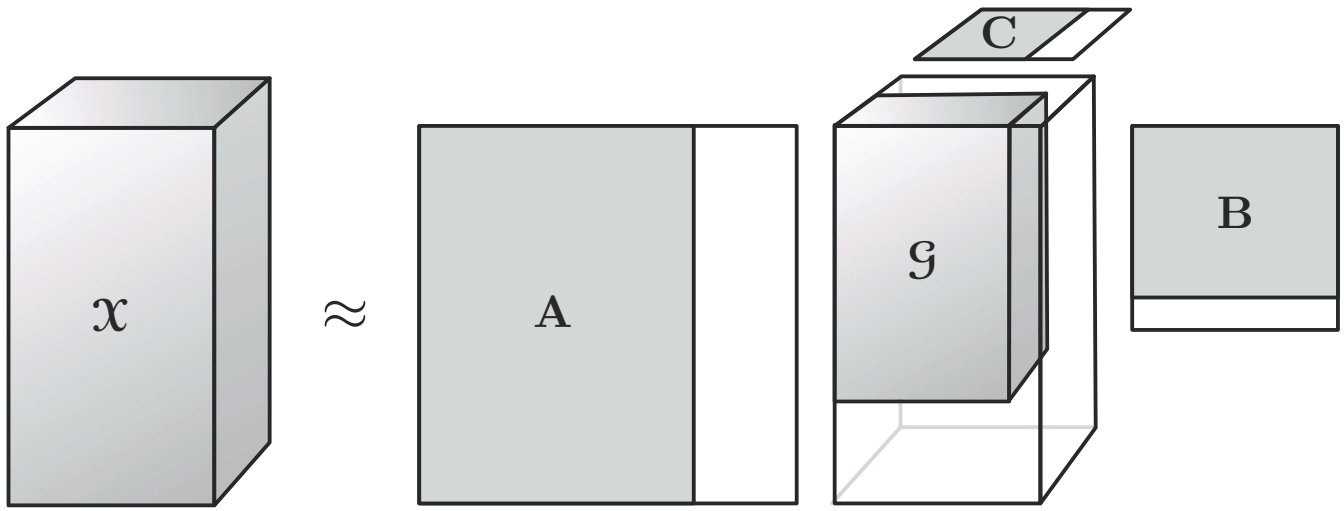
  - Used to "simplify" core (e.g., many zeros, all-orthogonal, ...)
- Can be generalized to N-way tensors
- Can be computed using ALS with matricized forms

$$\boldsymbol{X}_{(1)} \approx \boldsymbol{A}\boldsymbol{G}_{(1)}(\boldsymbol{C} \otimes \boldsymbol{B})^T$$
$$\boldsymbol{X}_{(2)} \approx \boldsymbol{B}\boldsymbol{G}_{(2)}(\boldsymbol{C} \otimes \boldsymbol{A})^T$$
$$\boldsymbol{X}_{(3)} \approx \boldsymbol{C}\boldsymbol{G}_{(3)}(\boldsymbol{B} \otimes \boldsymbol{B})^T$$

  - If factor matrices are column-orthogonal, update $\boldsymbol{\mathcal{G}} \leftarrow \boldsymbol{\mathcal{X}} \times_1 \boldsymbol{A}^T \times_2 \boldsymbol{B}^T \times_3 \boldsymbol{C}^T$

Higher-order SVD

- Tucker's "Method I" for computing the decomposition
  - Idea is to first find components that capture variation in each mode (independently of the other modes)
  - Set $\boldsymbol{A} = \boldsymbol{U}_1 =$ leading P left singular vectors of $\boldsymbol{X}_{(1)}$
  - Set $\boldsymbol{B} = \boldsymbol{U}_2 =$ leading Q left singular vectors of $\boldsymbol{X}_{(2)}$
  - Set $\boldsymbol{C} = \boldsymbol{U}_3 =$ leading R left singular vectors of $\boldsymbol{X}_{(3)}$
  - Set core tensor $\boldsymbol{\mathcal{G}} = \boldsymbol{\mathcal{X}} \times_1 \boldsymbol{U}_1^T \times_2 \boldsymbol{U}_2^T \times_3 \boldsymbol{U}_3^T$
- Today known as higher-order SVD (HOSVD)
- If P, Q, and R are smaller than rank of their corresponding matricized forms, we obtain a **truncated HOSVD**



Discussion

- SVD of $\boldsymbol{X}_{(n)}$ gives ***n*-mode singular vectors** ($\boldsymbol{U}_n$) and ***n*-mode singular values**
- Indeed, many useful properties of the SVD transfer over
  - Factors and slices of core tensor are ordered
    (Frobenius norm of slice = corresponding n-mode singular value)
  - Essentially unique
  - Core tensor is all-orthogonal
    (inner product between pairs of slices = 0)
- Core tensor cannot be made superdiagonal in general
- Truncated HOSVD is not optimal w.r.t. reconstruction error
  - But it holds

$$\|\mathcal{X} - \hat{\mathcal{X}}\|^2 \leq \text{sum of squares of truncated singular values}$$

  - Good starting point for ALS (or other methods)

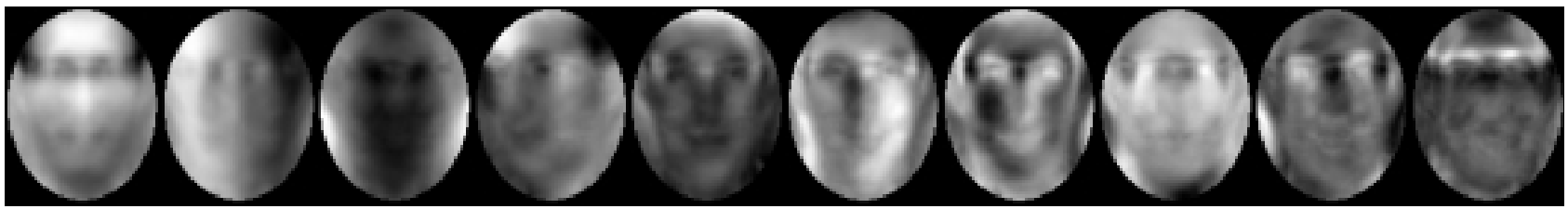
Example: TensorFaces

- Data: 7943-pixel B&W photographs of 28 people in 5 poses under 3 illumination setups performing 3 different expressions
  - $28 \times 5 \times 3 \times 3 \times 7943$ tensor (10M elements)

Vasilescu & Terzopoulos, 2002

Example: TensorFaces

- Data: 7943-pixel B&W photographs of 28 people in 5 poses under 3 illumination setups performing 3 different expressions
  - $28 \times 5 \times 3 \times 3 \times 7943$ tensor (10M elements)

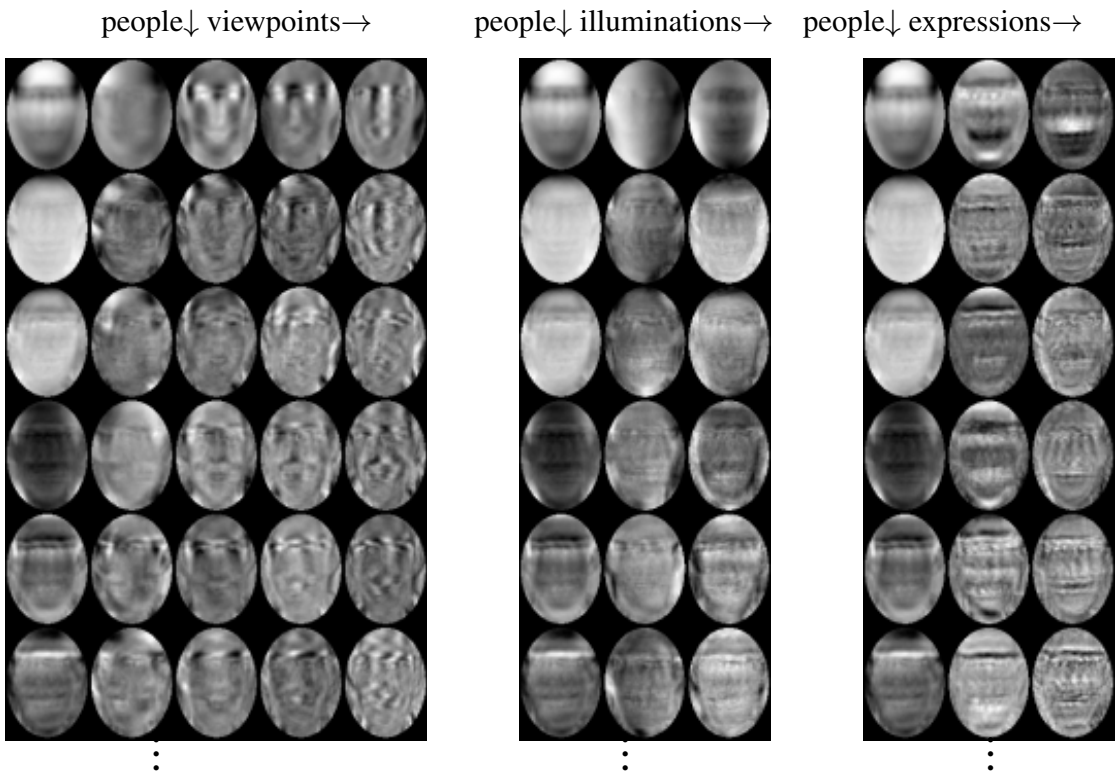


$\boldsymbol{U}_{\text{pix}}$ $(=\boldsymbol{U}_5)$ contains the eigenfaces
(SVD of pixels-by-picture matrix)

Vasilescu & Terzopoulos, 2002

Example: TensorFaces

- Data: 7943-pixel B&W photographs of 28 people in 5 poses under 3 illumination setups performing 3 different expressions
  - $28 \times 5 \times 3 \times 3 \times 7943$ tensor (10M elements)

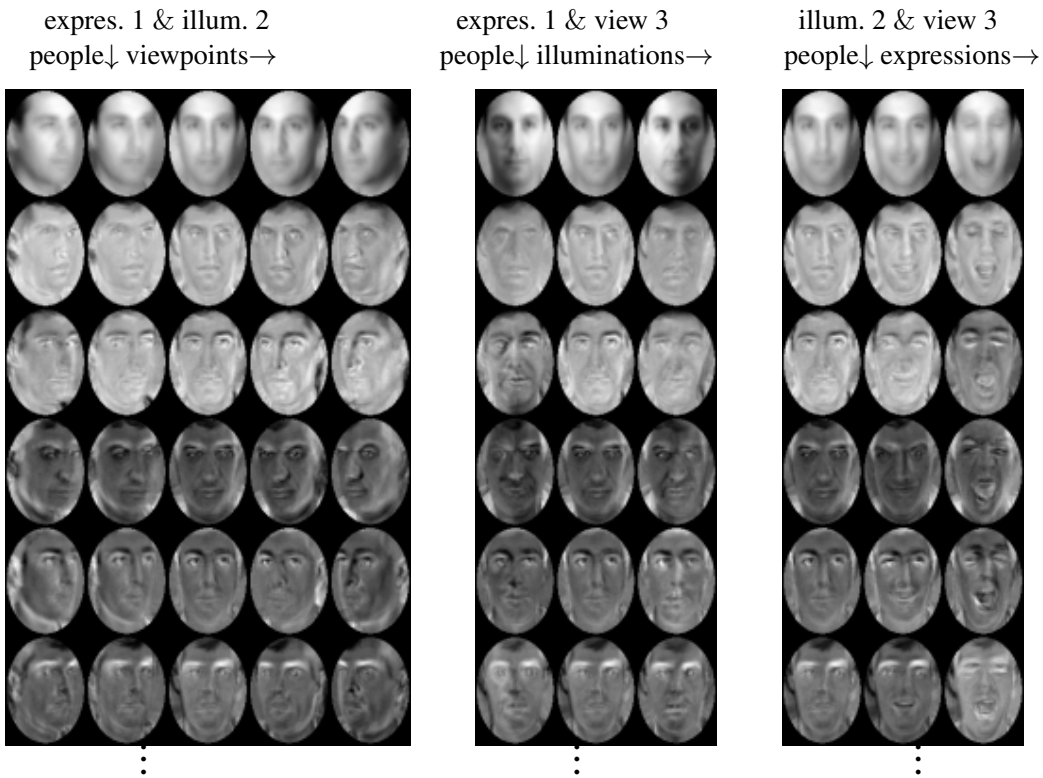


Some visualizations of $\mathcal{G} \times_5 \boldsymbol{U}_{\text{pix}}$
(decreasing "importance" from top to bottom / left to right)

Vasilescu & Terzopoulos, 2002

Example: TensorFaces

- Data: 7943-pixel B&W photographs of 28 people in 5 poses under 3 illumination setups performing 3 different expressions
  - $28 \times 5 \times 3 \times 3 \times 7943$ tensor (10M elements)



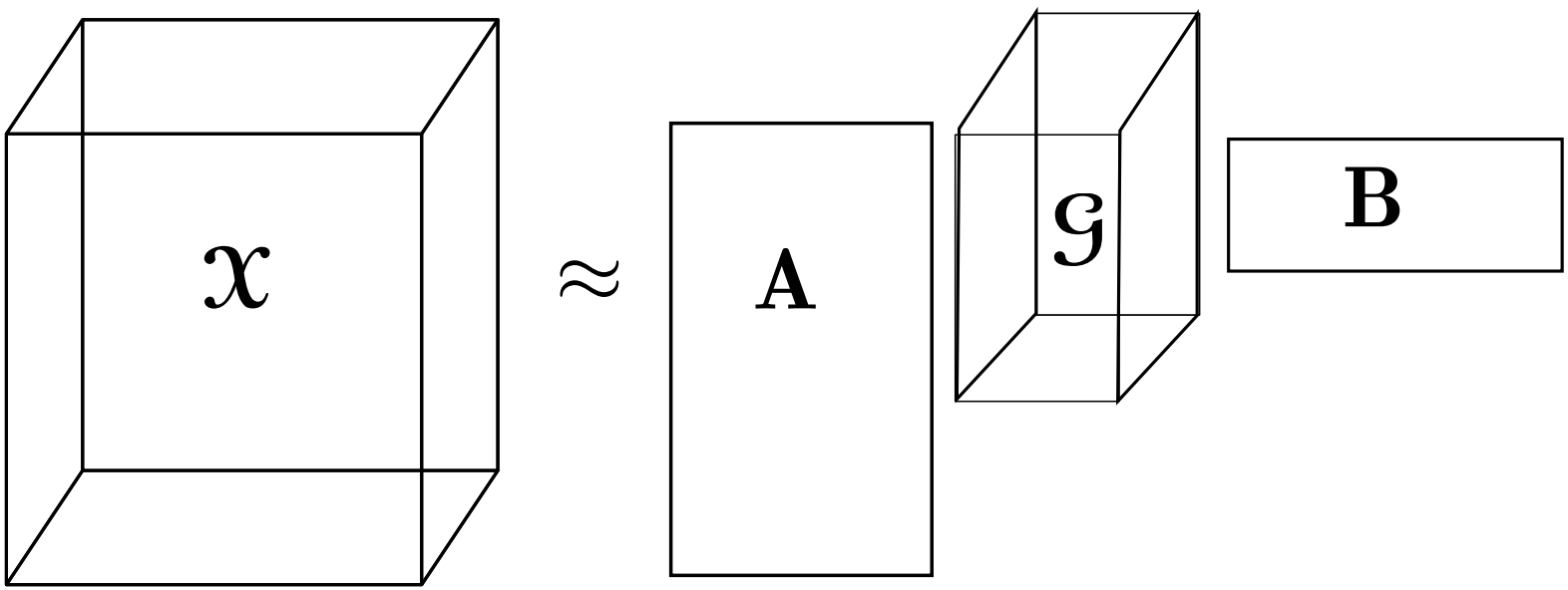
Some visualizations of $\mathcal{G} \times_2 \boldsymbol{U}_{\text{pos}} \times_3 \boldsymbol{U}_{\text{ill}} \times_4 \boldsymbol{U}_{\text{exp}} \times_5 \boldsymbol{U}_{\text{pix}}$. The rows show the componets w.r.t. people. The columns refer to different viewpoints, illuminations, and expressions.

Vasilescu & Terzopoulos, 2002

Outline

The Tucker2 decomposition

- Recall: Tucker3 decomposition decomposes 3-way tensor $\boldsymbol{\mathcal{X}}$ into three factor matrices $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$, and a core tensor $\boldsymbol{\mathcal{G}}$
- The **Tucker2 decomposition** decomposes a 3-way tensor into core and *two* factor matrices
  - Equivalently: third factor matrix is taken as the identity matrix
  - If data tensor is $I \times J \times K$, then core is $P \times Q \times K$



Why Tucker2?

- Use Tucker2 if you don't want to "compress" one of the modes
  - E.g., too small dimension (e.g., $500 \times 300 \times 3$)
  - E.g., want to handle this mode separately
  - For example, if third mode is time, we might first do Tucker2 and then analyze the slices $\{ \boldsymbol{G}_k \}$ over time
- Tucker2 is slightly simpler than Tucker3
  - We have $\boldsymbol{X}_k = \boldsymbol{A}\boldsymbol{G}_k\boldsymbol{B}^T$ for each frontal slice k
  - When using ALS, we can update each frontal slice separately during update of $\mathcal{G}$
- Forms basis for RESCAL and DEDICOM (up next)

The RESCAL decomposition

- The **RESCAL decomposition** combines Tucker2 and INDSCAL
- Given an I-by-I-by-K tensor $\boldsymbol{\mathcal{X}}$ and a rank R, find an $I \times R$ factor matrix $\boldsymbol{A}$ and an $R \times R \times K$ core tensor $\boldsymbol{\mathcal{R}}$ such that

$$\boldsymbol{\mathcal{X}} \approx \boldsymbol{\mathcal{R}} \times_1 \boldsymbol{A} \times_2 \boldsymbol{A}$$

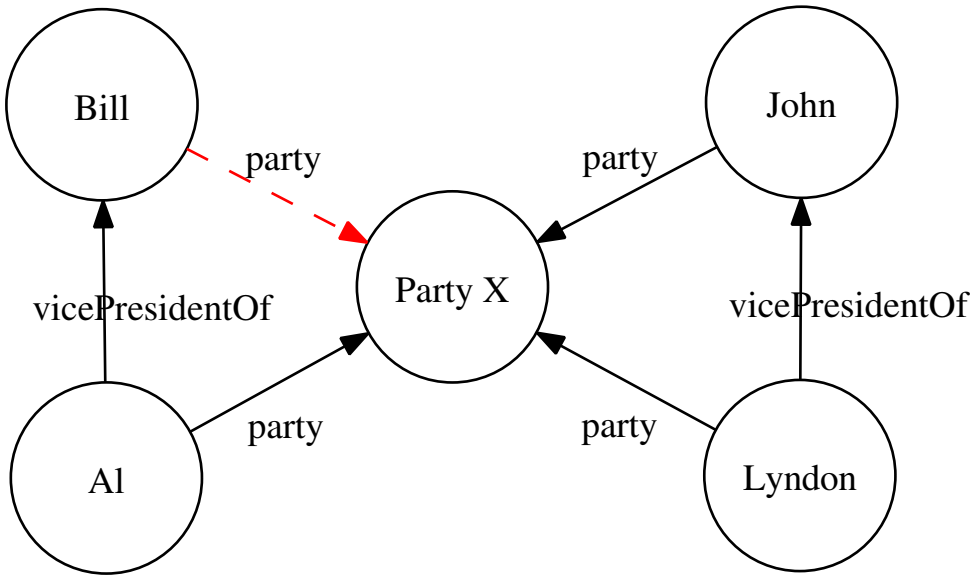
- I.e., minimize

$$\sum_{k=1}^{K} \| \boldsymbol{X}_k - \boldsymbol{A} \boldsymbol{R}_k \boldsymbol{A}^T \|_F^2.$$

- In practice, add regularization to avoid overfitting

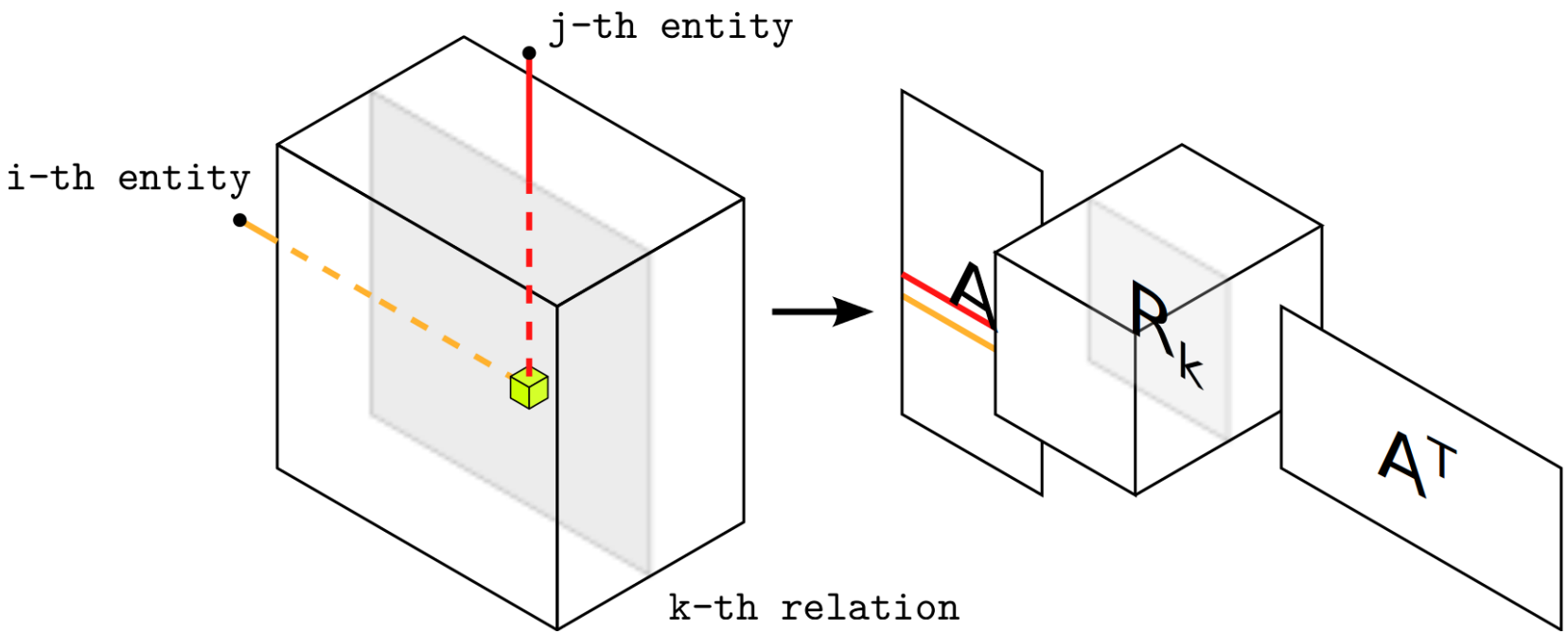
RESCAL and subject–object–predicate data (1)

- Proposed with subject–object–predicate data in mind
  - Designed for settings with few relations
  - E.g., YAGO knowledge base: < 100 relations, millions of entities
- Used for
  - Link prediction (by looking at $\hat{\mathcal{X}}$)
  - Assessment of entity similarity (by looking at $\boldsymbol{A}$)
  - Assessment of relation similarity (by looking at $\mathcal{R}$)
  - Currently a hot research area; many methods can be seen as constrained variants of RESCAL



Nickel et al., 2012

RESCAL and subject–object–predicate data (2)

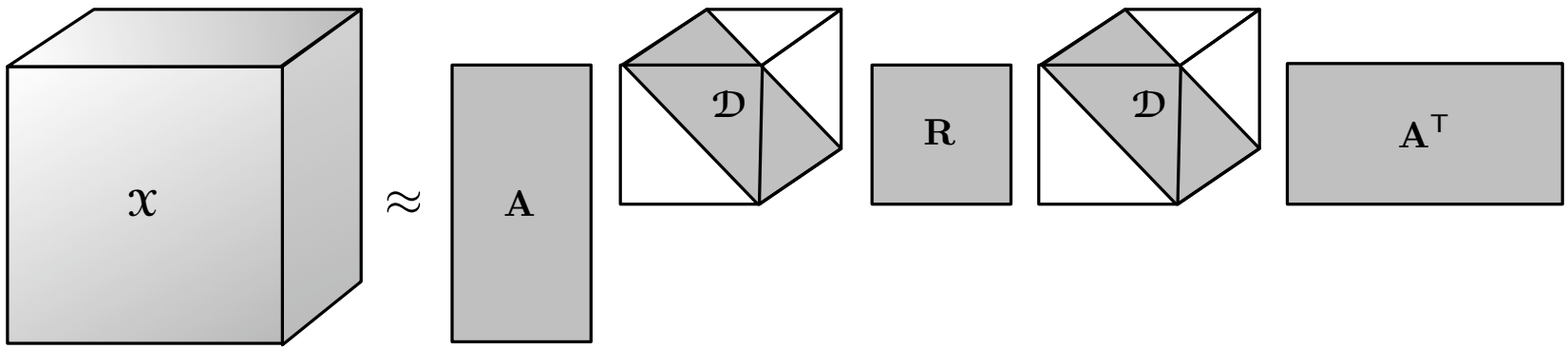


Why RESCAL?

- RESCAL computes an *embedding* for every entity ($\boldsymbol{A}_{i:}$) and for every relation ($\boldsymbol{R}_k$)
- With Tucker2, entity embeddings were different depending on whether the entity appears as subject or object
  - Having one embedding per entity ensures "information flow"
  - Facts with entity as subject help predicting facts with entity as obj.
- With INDSCAL, reconstructed relations were symmetric
  - That's not the case with RESCAL because $\boldsymbol{R}_k$ tells us how to mix embeddings to reconstruct the k-th relation
  - For symmetric relations, $\boldsymbol{R}_k$ should be symmetric
  - For asymmetric relations, $\boldsymbol{R}_k$ should be asymmetric
- Computation using ALS-style methods
  - To update $\boldsymbol{A}$, we use mode-1 matricization of RESCAL (as before)
  - $\boldsymbol{X}_{(1)} \approx \boldsymbol{A}\boldsymbol{R}_{(1)}(\boldsymbol{I} \otimes \boldsymbol{A})^T$
  - Hard because $\boldsymbol{A}$ appears on left- and right-hand side
  - Original paper uses hack (treat right-hand side as fixed + cleverness)

The DEDICOM decomposition

- The **DEDICOM decomposition** is a precursor to RESCAL
  - Mixing matrix is the same for all relations
  - But entity factors are weighed differently
  - $\boldsymbol{X}_k = \boldsymbol{A}\boldsymbol{D}_k\boldsymbol{R}\boldsymbol{D}_k\boldsymbol{A}^T$, where $\boldsymbol{D}_k$ is diagonal and contains the weights of each factor for the k-th relation
- Many of the ideas in RESCAL are based on DEDICOM
  - But DEDICOM is more involved to scale to large data
  - And DEDICOM generally needs larger R due to shared relation matrix



Harshman, 1978

Outline

Lessons learned

- Tensors generalize matrices
  - Many matrix concepts generalize well
  - But some don't, and some behave very differently
- Compared to matrix decomposition methods, tensor algorithms are in their youth
- Tensor decompositions are used in many different fields of science
  - Sometimes the wheel gets re-invented multiple times
  - Traditionally tensor problems were dense
    $\rightarrow$ Fewer algorithms for decomposing sparse tensors
- There are many tensor decompositions related to CP and Tucker
  - Need care to select the one that's best suited for the task at hand
  - Computational complexity can be an issue

Suggested reading

- Skillicorn, Ch. 9
- Kolda & Bader, *Tensor Decompositions and Applications*, SIAM Rew. 51(3), 2009
  - A great survey on tensor decompositions, includes many variants and applications
- Acar & Yener, *Unsupervised Multiway Data Analysis: A Literature Survey*, IEEE Trans. Knowl. Data Eng. 21(1), 2009
  - Another survey, shorter and more focused on applications
- All the papers linked at the bottom parts of the slides