Epilogue

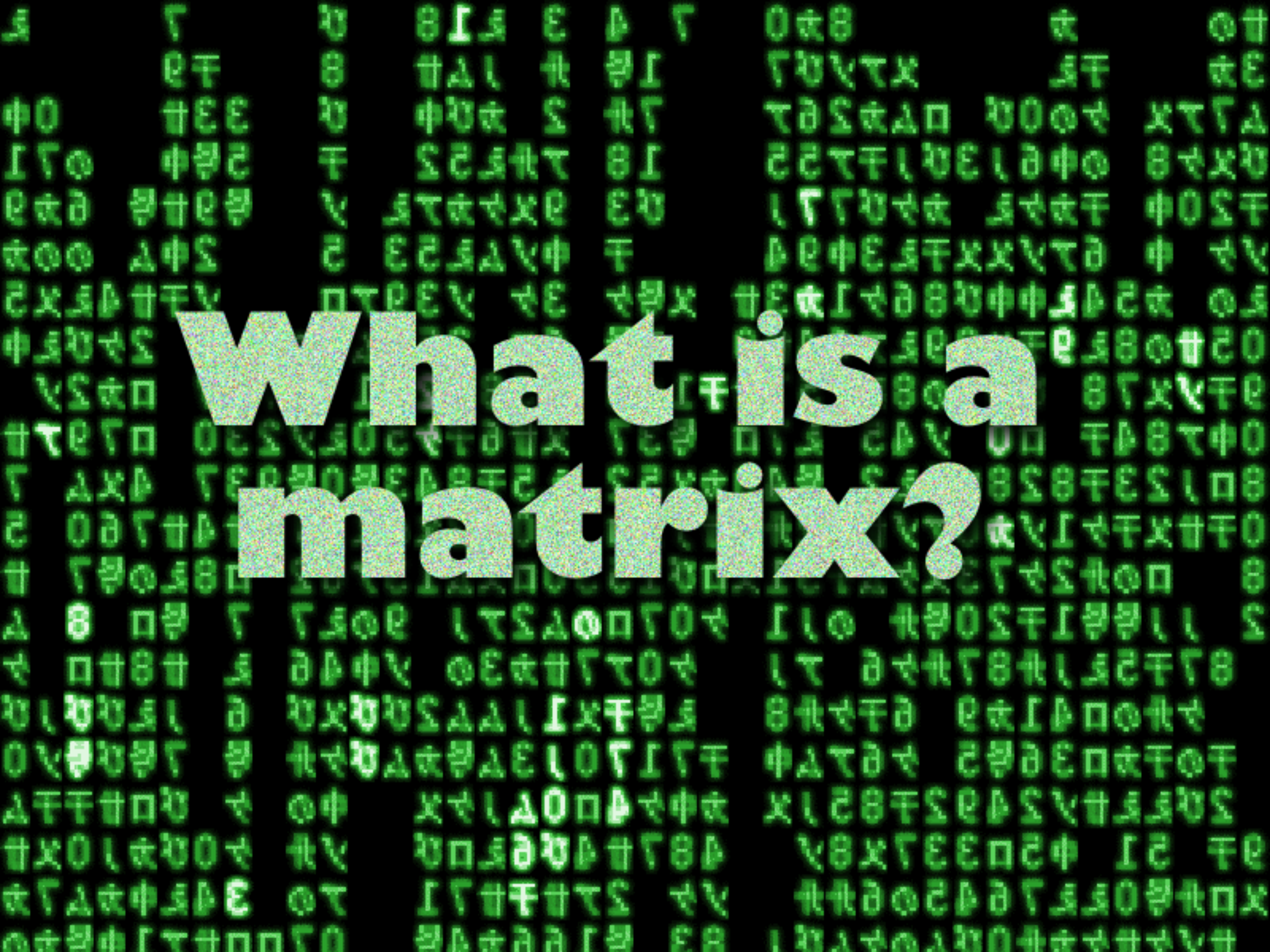
Wrap-Up

Recap of the Course

Data Mining & Matrices

Pauli Miettinen ◆ Saskia Metzler ◆ Sanjar Karaev

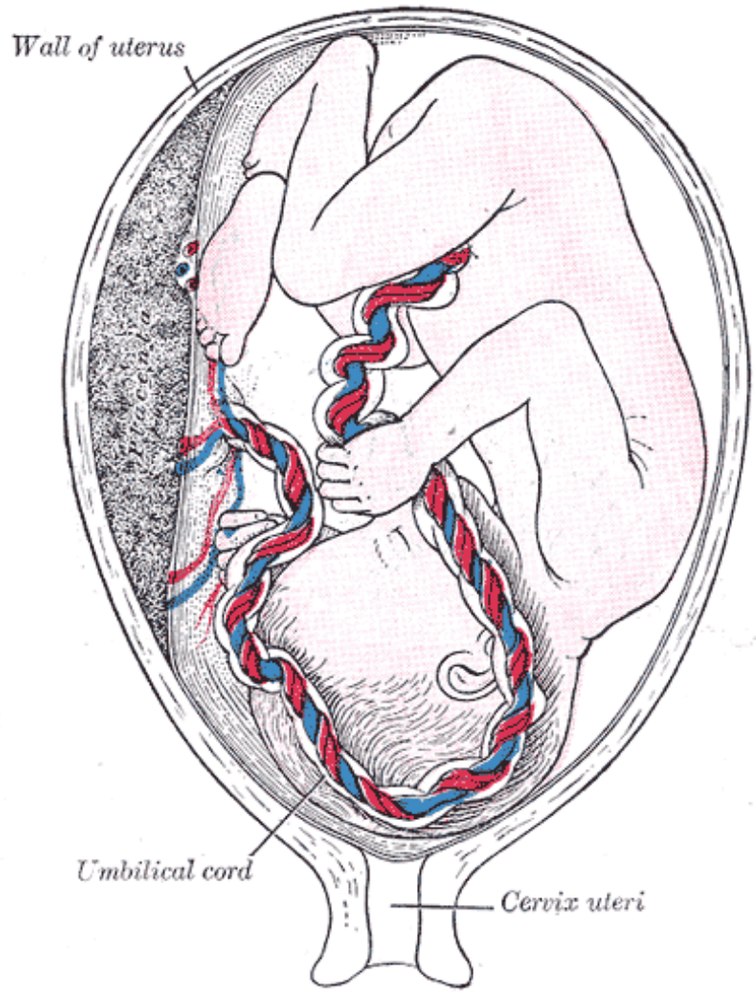
- ☛ SVD, PCA, CUR, NMF, ICA, …
- ☛ 6 credits
- ☛ Theoretical & hands-on assignments
- ☛ First lecture: tomorrow at high noon in room 029, building E1 5 (MPI-SWS)
- ☛ http://bit.ly/dmm15



What is a matrix?

A womb?

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



A womb

Matrices in data mining

$$\begin{array}{l} \\ \text{Anna} \\ \text{Bob} \\ \text{Charlie} \end{array} \begin{array}{c} \begin{array}{ccc} \text{Bread} & \text{Butter} & \text{Beer} \end{array} \\ \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix} \end{array}$$

Customer transactions

$$\begin{array}{l} \\ \text{Book 1} \\ \text{Book 2} \\ \text{Book 3} \end{array} \begin{array}{c} \begin{array}{ccc} \text{Data} & \text{Matrix} & \text{Mining} \end{array} \\ \begin{pmatrix} 5 & 0 & 3 \\ 0 & 0 & 7 \\ 4 & 6 & 5 \end{pmatrix} \end{array}$$

Document-term matrix

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

Incomplete rating matrix

$$\begin{array}{l} \\ \text{Saarbrücken} \\ \text{Helsinki} \\ \text{Cape Town} \end{array} \begin{array}{c} \begin{array}{ccc} \text{Jan} & \text{Jun} & \text{Sep} \end{array} \\ \begin{pmatrix} 1 & 11 & 10 \\ 6.5 & 10.9 & 8.7 \\ 15.7 & 7.8 & 8.7 \end{pmatrix} \end{array}$$

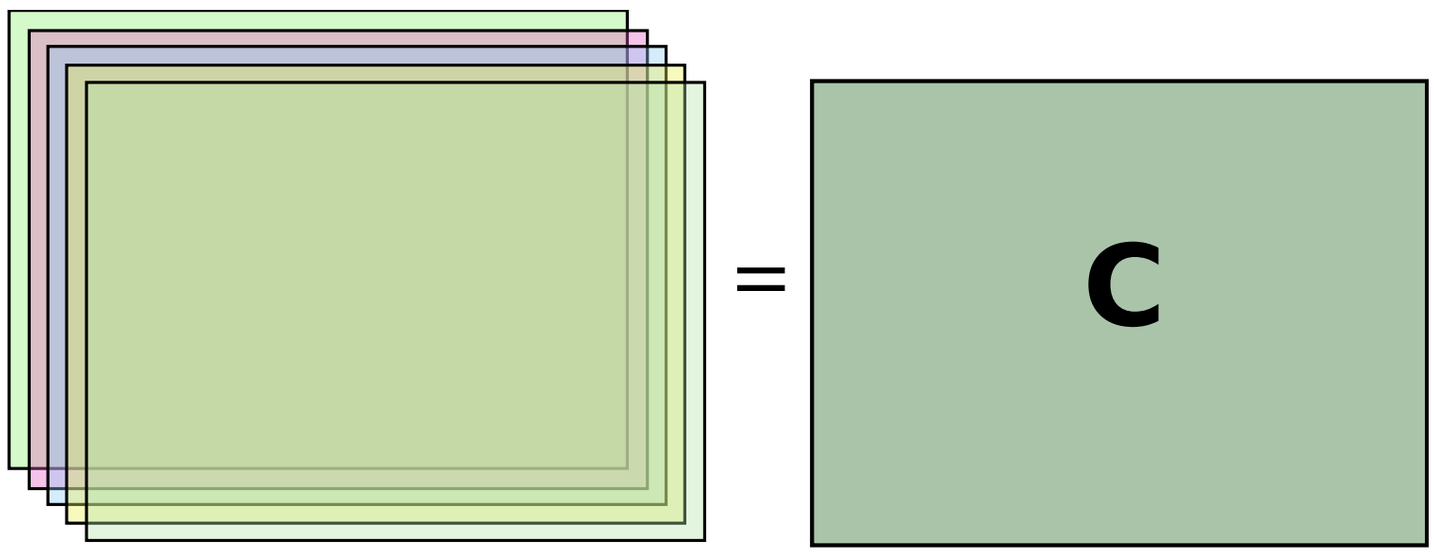
Cities and monthly temperatures

Matrix decompositions in data mining

- A common goal in data mining is to find regularities (or patterns) in the data
  - Often, to summarize the data
- A *matrix decomposition* presents the data as a sum of “simple” elements, i.e. patterns
  - but there’s also other uses... *stay tuned!*

Intuition for Matrix Multiplication

- Matrix $\boldsymbol{AB}$ is a sum of k matrices $\boldsymbol{a}_l\boldsymbol{b}_l^T$ obtained by multiplying the l-th column of $\boldsymbol{A}$ with the l-th row of $\boldsymbol{B}$



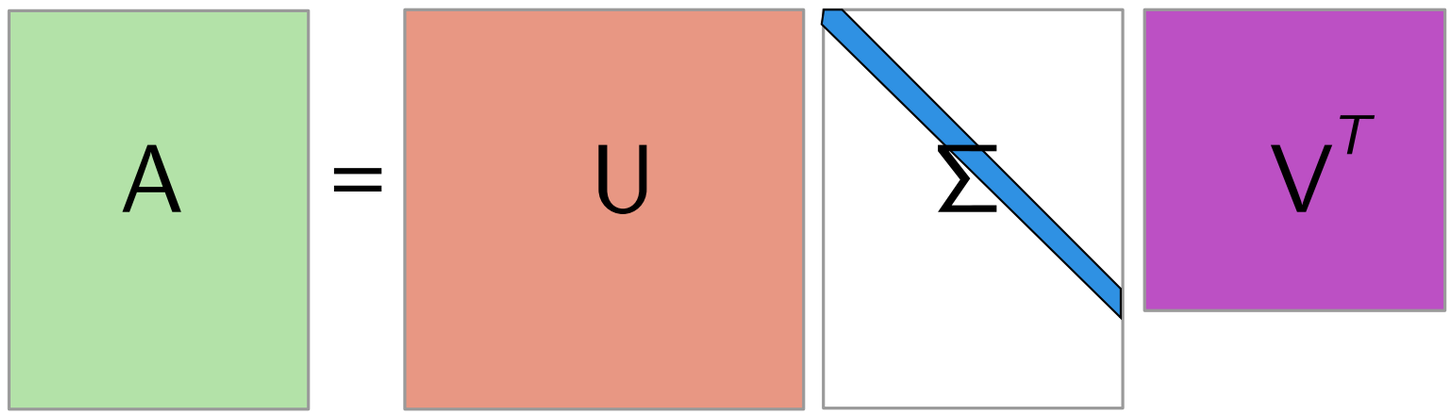
$$\boldsymbol{C} = \sum_{\ell=1}^{k} \boldsymbol{a}_\ell \boldsymbol{b}_\ell^T$$

“The SVD is the Swiss Army knife of matrix decompositions”

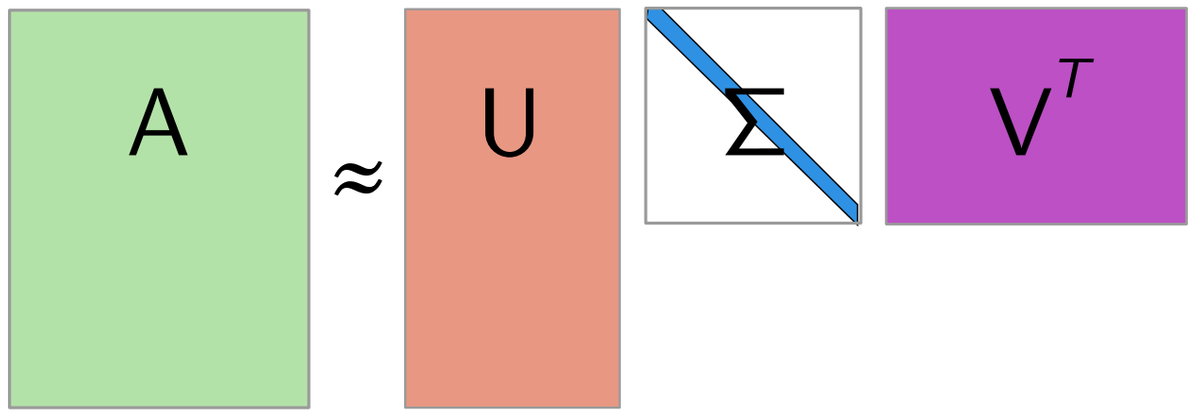
– Diane O’Leary, 2006

SVD

Full



Truncated



Why is SVD important?

- It gives us the **dimensions of the fundamental subspaces**
- It lets us **compute various norms**
- It tells about **sensitivity of linear systems**
- It gives us optimal solutions to **least-squares linear systems**
- It gives us the **least-error rank-*k* decomposition**
- **Every matrix has one**

The *z*-scores

- The ***z*-scores** are attributes whose values are transformed by
  - centering them to 0 by removing the (column) mean from each value
  - normalizing the magnitudes by dividing every value with the (column) standard deviation

$$X' = \frac{X - \mu}{\sigma}$$

Issues with the rank-selection methods

- Require computing the full SVD first or otherwise computationally heavy

  Guttman–Kaiser | scree | entropy-based | random flips

- Require subjective evaluation

  scree | random flips

- Based on arbitrary thresholds

  Guttman–Kaiser | entropy-based

Example use of SVD

- Data: people's ratings on different wines
- Scatterplot of first two LSV
  - SVD doesn't know what the data is
- Conclusion: winelovers like red and white alike, others are more biased

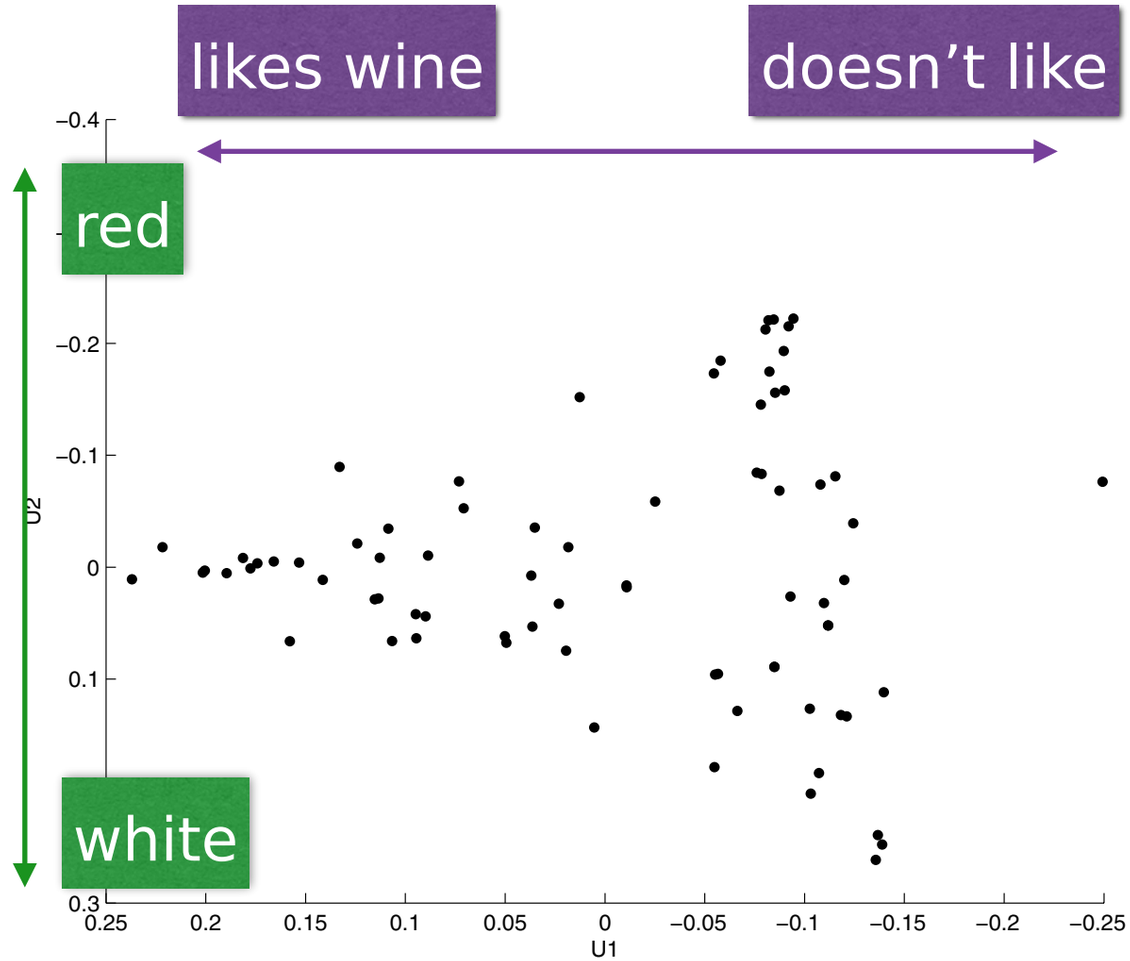
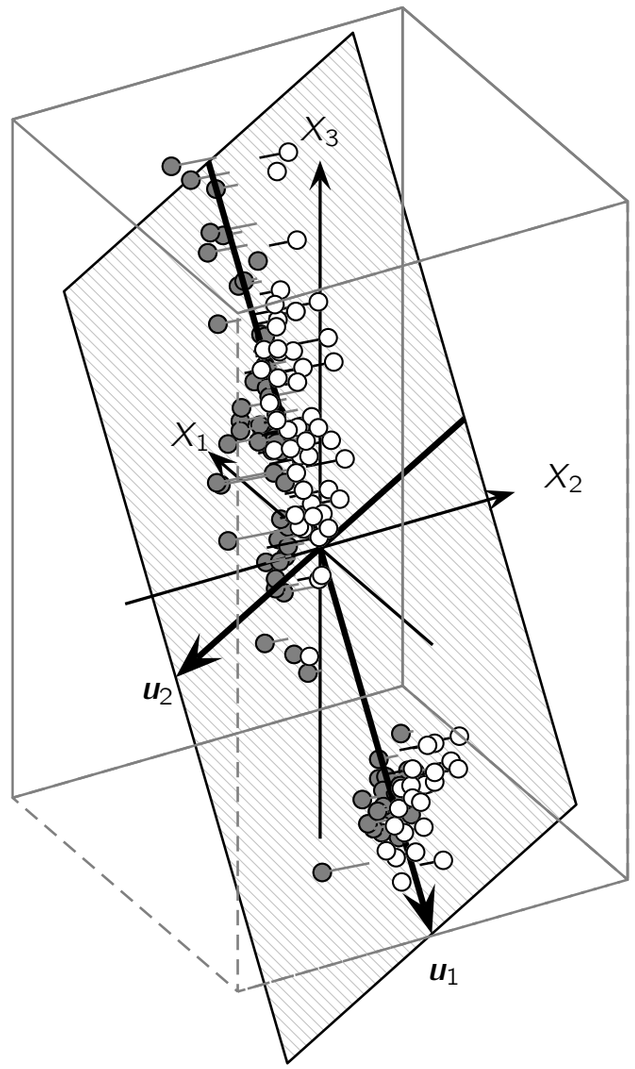


Figure 3.2. *The first two factors for a dataset ranking wines.*

SVD gives directions of largest variances

- The singular vectors give the directions of the largest variances
  - First singular vector points to the direction of the largest variance
    - Second to the second-largest
      - Spans a hyperplane with the first
- The projection distance to these hyperplanes is minimal over all hyperplanes (Eckart–Young)



Very general idea for solving SVD

- SVD is unique
  - If $\boldsymbol{U}$ and $\boldsymbol{V}$ are orthogonal s.t. $\boldsymbol{U}^T\boldsymbol{A}\boldsymbol{V} = \boldsymbol{\Sigma}$, then $\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ is the SVD of $\boldsymbol{A}$
- Idea: find orthogonal $\boldsymbol{U}$ and $\boldsymbol{V}$ s.t. $\boldsymbol{U}^T\boldsymbol{A}\boldsymbol{V}$ is as desired
  - Iterative process: find orthogonal $\boldsymbol{U}_1$, $\boldsymbol{U}_2$, ... and set $\boldsymbol{U} = \boldsymbol{U}_1\boldsymbol{U}_2\boldsymbol{U}_3\cdots$
    - Still orthogonal

Householder reflections

- A **Householder reflection** is n-by-n matrix

$$\boldsymbol{P} = \boldsymbol{I} - \beta \boldsymbol{v}\boldsymbol{v}^T \quad \text{where} \quad \beta = \frac{2}{\boldsymbol{v}^T\boldsymbol{v}}$$

  - If we set $\boldsymbol{v} = \boldsymbol{x} - ||\boldsymbol{x}||_2\boldsymbol{e}_1$, then $\boldsymbol{P}\boldsymbol{x} = ||\boldsymbol{x}||_2\boldsymbol{e}_1$
    - $\boldsymbol{e}_1 = (1, 0, 0, \ldots, 0)^T$
  - Note: $\boldsymbol{P}\boldsymbol{A} = \boldsymbol{A} - (\beta\boldsymbol{v})(\boldsymbol{v}^T\boldsymbol{A})$ where $\beta = 2/(\boldsymbol{v}^T\boldsymbol{v})$
    - We never have to compute matrix $\boldsymbol{P}$

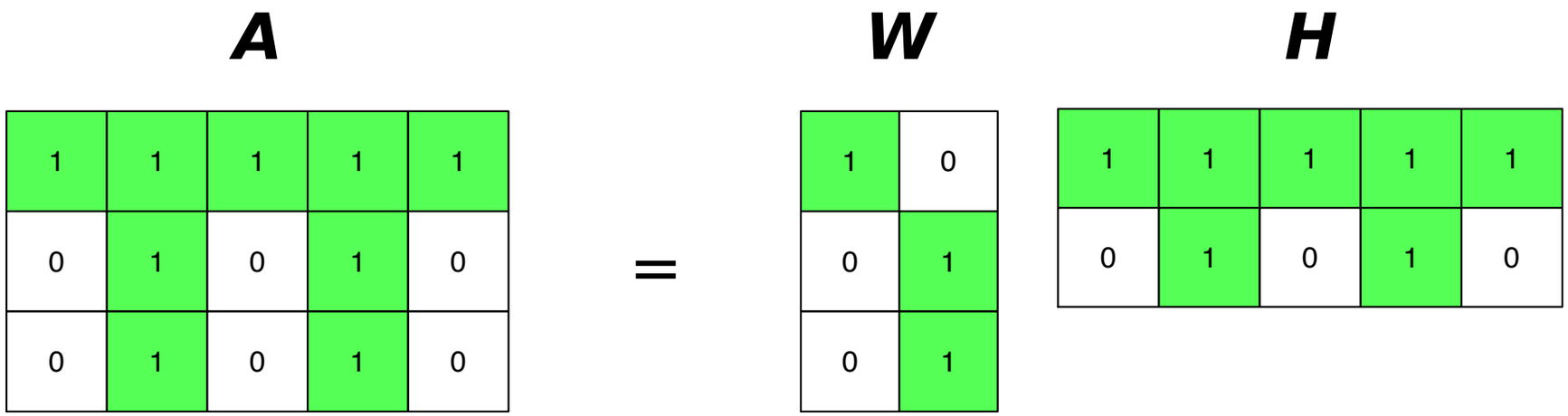
Givens rotations

- Householder is too crude to give identity
- **Givens rotations** are rank-2 corrections to the identity of form

$$\boldsymbol{G}(i,k,\theta) = \begin{pmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & \cos(\theta) & \cdots & \sin(\theta) & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & -\sin(\theta) & \cdots & \cos(\theta) & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{pmatrix} \begin{matrix} \\ \\ i \\ \\ k \\ \\ \\ \end{matrix}$$

$$\begin{matrix} & & i & & k & & \end{matrix}$$

Non-negative matrix factorization



Forcing the factors to be non-negative can, and often will, improve the interpretability of the factorization

Some comments on NMF

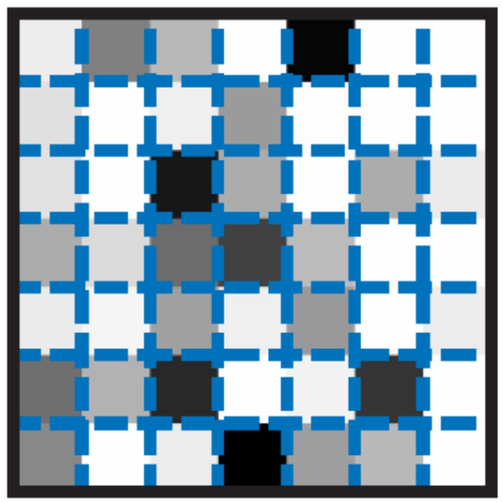
- NMF is **not** unique
  - If $\boldsymbol{X}$ is nonnegative and with nonnegative inverse, then $\boldsymbol{WXX}^{-1}\boldsymbol{H}$ is equivalent valid decomposition
- Computing NMF (and non-negative rank) is NP-hard
  - This was open until 2008

NMF example: faces

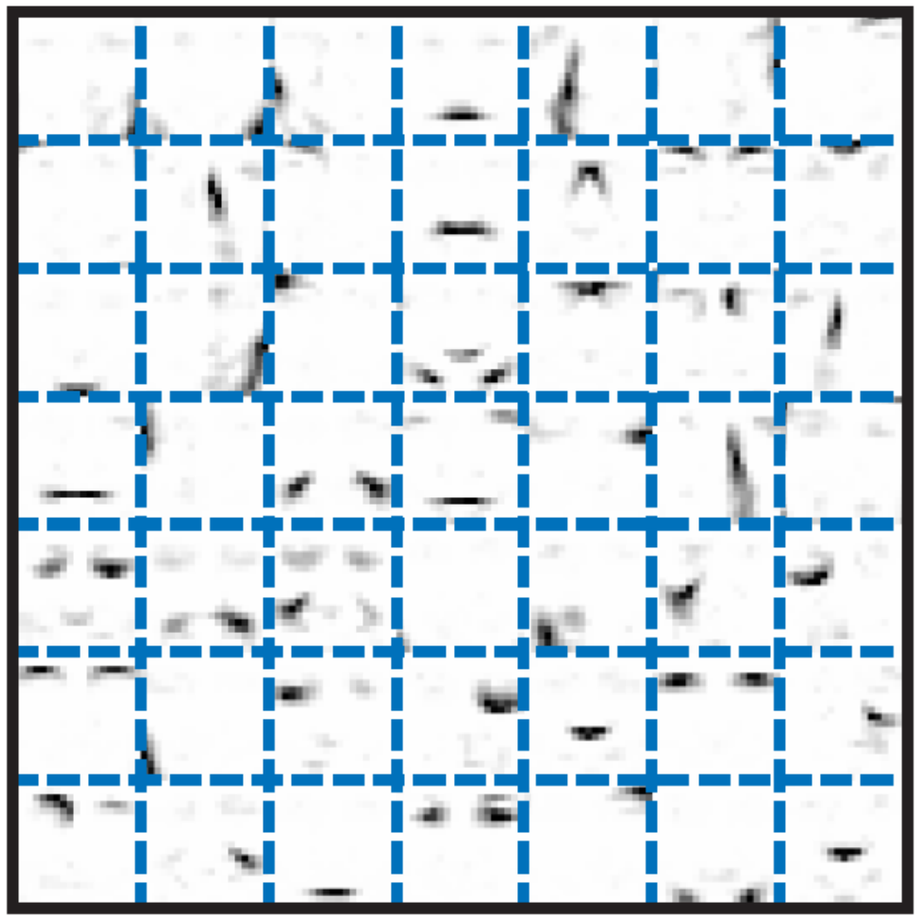
Row of original

NMF

=



×



General idea for solving NMF

- NMF is not convex, but it is **biconvex**
  - If $\boldsymbol{W}$ is fixed, $\frac{1}{2}\|\boldsymbol{A}-\boldsymbol{WH}\|_F^2$ is convex
  - Start from random $\boldsymbol{W}$ and **repeat**
    - Fix $\boldsymbol{W}$ and update $\boldsymbol{H}$
    - Fix $\boldsymbol{H}$ and update $\boldsymbol{W}$
  - **until** the error doesn’t decrease anymore

The NMF-ALS algorithm

1. $\boldsymbol{W} \leftarrow \text{random}(n, k)$
2. **repeat**
   - 2.1. $\boldsymbol{H} \leftarrow [\boldsymbol{W}^{+}\boldsymbol{A}]_{+}$
   - 2.2. $\boldsymbol{W} \leftarrow [\boldsymbol{A}\boldsymbol{H}^{+}]_{+}$
3. **until** convergence

The NMF gradient descent algorithm

1. $\boldsymbol{W} \leftarrow \text{random}(n, k)$
2. $\boldsymbol{H} \leftarrow \text{random}(k, m)$
3. **repeat**
   - 3.1. $\boldsymbol{H} \leftarrow \boldsymbol{H} - \varepsilon_{\boldsymbol{H}} \frac{\partial f}{\partial \boldsymbol{H}}$
   - 3.2. $\boldsymbol{W} \leftarrow \boldsymbol{W} - \varepsilon_{\boldsymbol{W}} \frac{\partial f}{\partial \boldsymbol{W}}$
4. **until** convergence

The NMF multiplicative updates algorithm

1. $\boldsymbol{W} \leftarrow \text{random}(n, k)$
2. $\boldsymbol{H} \leftarrow \text{random}(k, m)$
3. **repeat**
   - 3.1. $h_{ij} \leftarrow h_{ij} \frac{(\boldsymbol{W}^T \boldsymbol{A})_{ij}}{(\boldsymbol{W}^T \boldsymbol{W} \boldsymbol{H})_{ij} + \varepsilon}$
   - 3.2. $w_{ij} \leftarrow w_{ij} \frac{(\boldsymbol{A} \boldsymbol{H}^T)_{ij}}{(\boldsymbol{W} \boldsymbol{H} \boldsymbol{H}^T)_{ij} + \varepsilon}$
4. **until** convergence

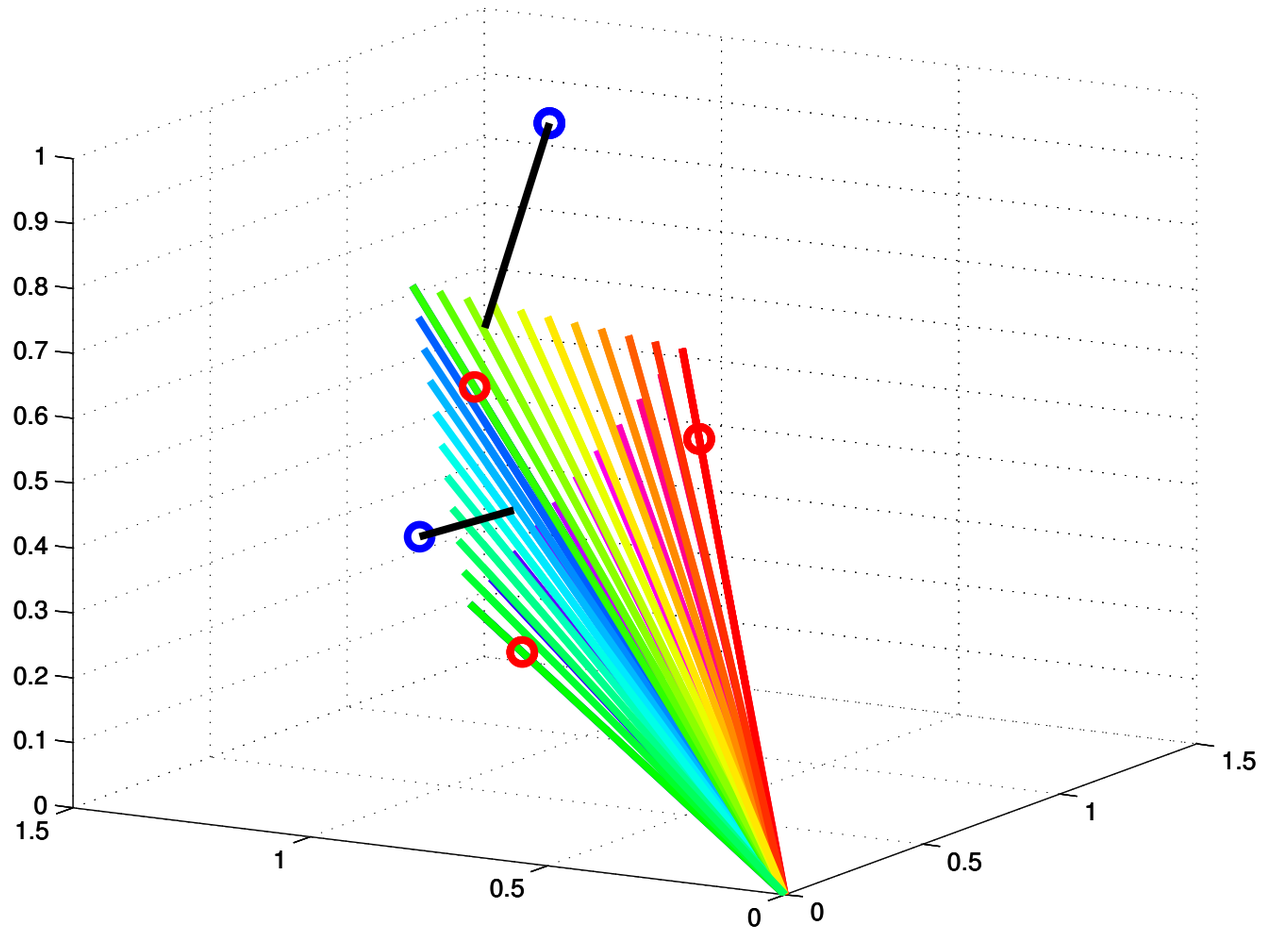
Geometry of NMF

NMF factors

Data points

Convex cone

Projections



Hoyer's sparse NMF

- Hoyer (2004) considers the following sparsity function for n-dimensional vector $\boldsymbol{x}$

$$\text{sparsity}(\boldsymbol{x}) = \frac{\sqrt{n} - \left(\sum_i |x_i|\right) / \sqrt{\sum_i x_i^2}}{\sqrt{n} - 1}$$

- sparsity($\boldsymbol{x}$) = 1 iff nnz($\boldsymbol{x}$) = 1
- sparsity($\boldsymbol{x}$) = 0 iff $|\boldsymbol{x}_i| = |\boldsymbol{x}_j|$ for all i, j

Hoyer 2004

Semi-orthogonal NMF

- In **semi-orthogonal NMF** we restrict $\boldsymbol{H}$ to row-orthogonal:

  minimize $||\boldsymbol{A} - \boldsymbol{WH}||_F$ s.t. $\boldsymbol{HH}^T = \boldsymbol{I}$ and $\boldsymbol{W}$ and $\boldsymbol{H}$ are nonnegative

  - Solutions are unique (up to permutations)
  - The problem is “equivalent” to *k*-means
    - In the sense that the optimal solutions have the same value

Ding et al. 2006

NMF and clustering

- In k-means, we minimize

$$\sum_{j=1}^{k} \sum_{i \in C_j} \left\| \boldsymbol{a}_i - \boldsymbol{\mu}_j \right\|_2^2 = \sum_{j=1}^{k} \sum_{i=1}^{n} \boldsymbol{G}_{ij} \left\| \boldsymbol{a}_i - \boldsymbol{\mu}_j \right\|_2^2$$

  - $\boldsymbol{\mu}_j$ is the centroid of the jth cluster C_j
  - $\boldsymbol{G}$ is n-by-k **cluster assignment matrix**
    - $\boldsymbol{G}_{ij} = 1$ if $i \in C_j$ and 0 otherwise
  - Equivalently: $\left\| \boldsymbol{A} - \boldsymbol{G}\boldsymbol{M} \right\|_F^2$

Type of NMF if $\boldsymbol{A}$ is nonnegative!

    - $\boldsymbol{M}$ is k-by-m containing the centroids as its rows

Orthogonal tri-factor NMF

- We can find NMF where both $\boldsymbol{W}$ and $\boldsymbol{H}$ are (column/row) orthogonal
  - Often too restrictive; cannot handle different scales
- In **orthogonal nonnegative tri-factorization** we add third non-negative matrix $\boldsymbol{S}$:

  minimize $||\boldsymbol{A} - \boldsymbol{WSH}||_F$ s.t. $\boldsymbol{W}^T\boldsymbol{W} = \boldsymbol{I}$, $\boldsymbol{H}\boldsymbol{H}^T = \boldsymbol{I}$, and all matrices are non-negative

Generalized KL-divergence and matrix factorizations

- The standard KL-divergence requires P and Q be probability distributions (e.g. $\sum_i P(i) = 1$)
  - The **generalized KL-divergence** (or **I-divergence**) removes this requirement:

$$D_{\mathrm{GKL}}(P\|Q) = \sum_i \left(P(i) \ln \frac{P(i)}{Q(i)} - P(i) + Q(i) \right)$$

- In NMF, $P = \boldsymbol{A}$ and $Q = \boldsymbol{WH}$:

$$D_{\mathrm{GKL}}(\boldsymbol{A}\|\boldsymbol{WH}) = \sum_{i,j} \left(\boldsymbol{A}_{ij} \ln \frac{\boldsymbol{A}_{ij}}{(\boldsymbol{WH})_{ij}} - \boldsymbol{A}_{ij} + (\boldsymbol{WH})_{ij} \right)$$

NMF for GKL

- The update rules for multiplicative GKL NMF algorithm are

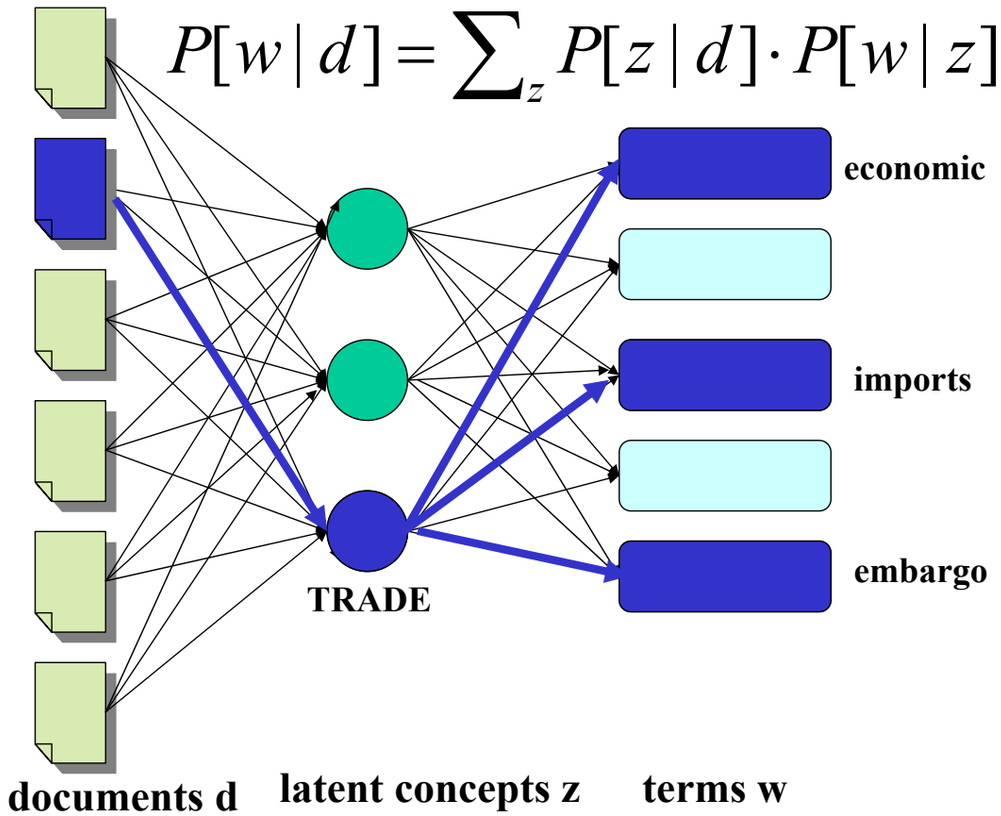
$$\boldsymbol{H}_{kj} \leftarrow \boldsymbol{H}_{kj} \frac{\sum_{i=1}^{n} \boldsymbol{W}_{ik} (\boldsymbol{A}_{ij} / (\boldsymbol{WH})_{ij})}{\sum_{i=1}^{n} \boldsymbol{W}_{ik}}$$

$$\boldsymbol{W}_{ik} \leftarrow \boldsymbol{W}_{ik} \frac{\sum_{j=1}^{m} (\boldsymbol{A}_{ij} / (\boldsymbol{WH})_{ij}) \boldsymbol{H}_{kj}}{\sum_{j=1}^{m} \boldsymbol{H}_{kj}}$$

  - The columns of $\boldsymbol{W}$ are normalized to sum to unity after every iteration

pLSI generative process

- Pick a document according to $P(d)$
- Select a topic according to $P(z \mid d)$
- Select a word according to $P(w \mid z)$



The CX decomposition

- In the **CX decomposition** we are given a matrix $\boldsymbol{A}$ and a rank k, and we need to select k **columns** of $\boldsymbol{A}$ into matrix $\boldsymbol{C}$ and build matrix $\boldsymbol{X}$ s.t. we minimize $||\boldsymbol{A} - \boldsymbol{CX}||_\xi$
  - ξ is either F or 2
  - A.k.a. **column subset selection problem** (CSSP)

Related idea: RRQR

- The **rank-revealing QR** (RRQR) factorization of matrix $\boldsymbol{A}$ is

$$\boldsymbol{A}\boldsymbol{\Pi} = \boldsymbol{Q}\boldsymbol{R} = \boldsymbol{Q}\begin{pmatrix} \boldsymbol{R}_{11} & \boldsymbol{R}_{12} \\ 0 & \boldsymbol{R}_{22} \end{pmatrix}$$

k-by-k upper-triangular w/ positive diagonal

k-by-$(m–k)$

n-by-n orthogonal

$(n–k)$-by-$(m–k)$

Permutation matrix

n-by-m

that satisfies

$$\frac{\sigma_k(\boldsymbol{A})}{p_1(k,m)} \leq \sigma_{\min}(\boldsymbol{R}_{11}) \leq \sigma_k(\boldsymbol{A})$$

$$\sigma_{k+1}(\boldsymbol{A}) \leq \sigma_{\max}(\boldsymbol{R}_{22}) \leq p_2(k,m)\sigma_{k+1}(\boldsymbol{A})$$

kth singular value of $\boldsymbol{A}$

Some polynomial on k and m

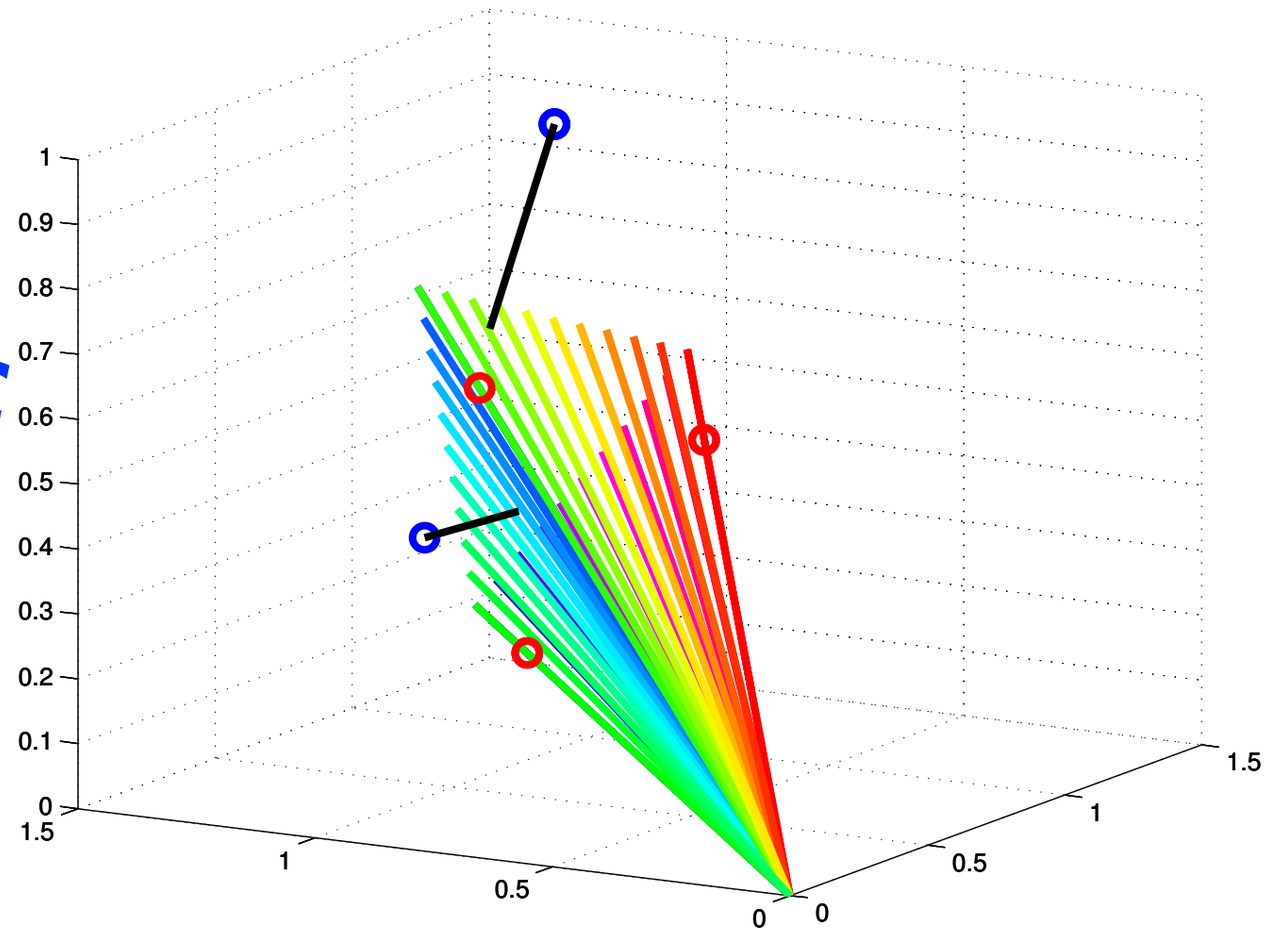
Geometry of NNCX

Columns in C

Columns not in C

Convex cone

Projections



The CUR decomposition

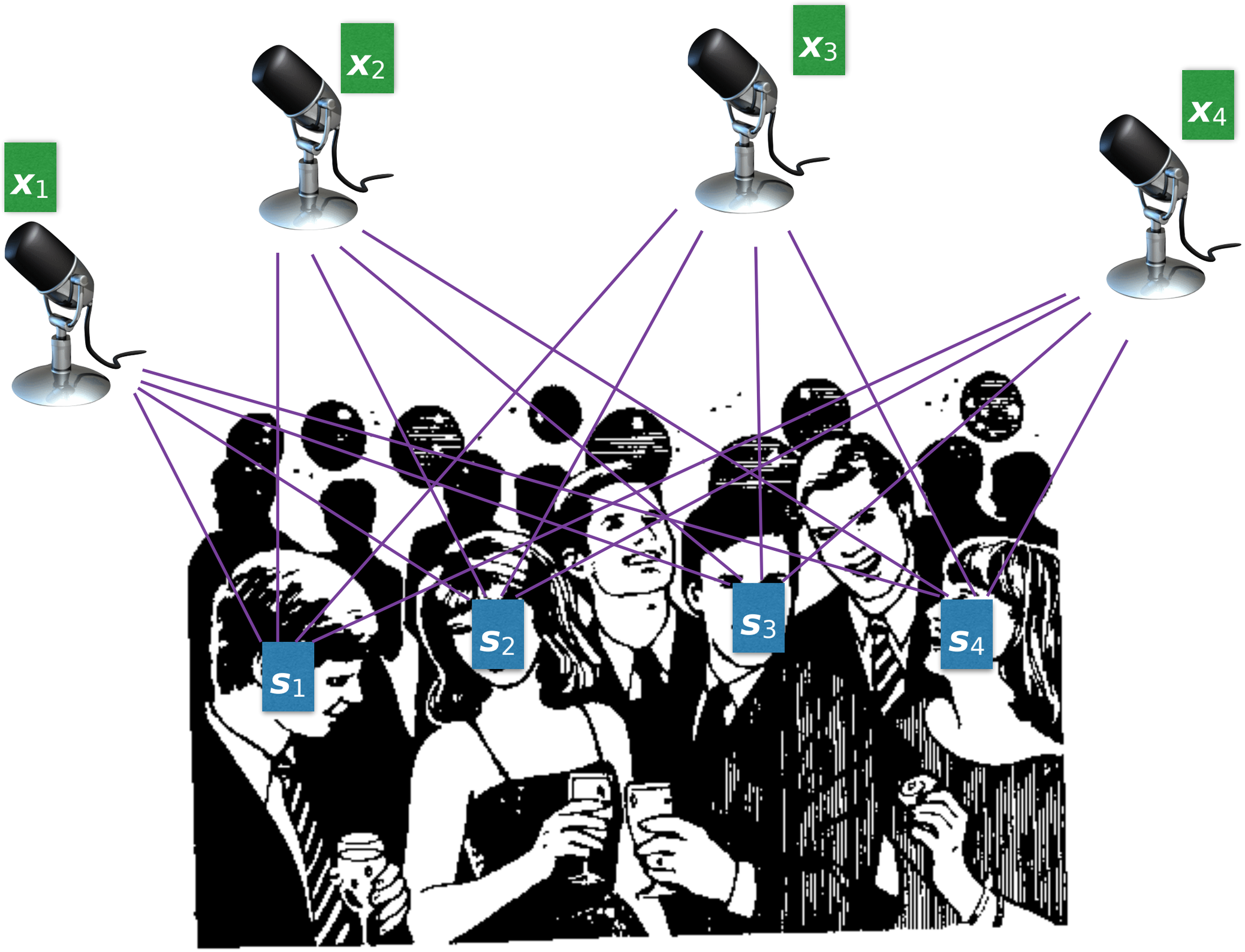
- In the **CUR decomposition** we are given matrix $\boldsymbol{A}$ and integers c and r, and our task is to select c columns of $\boldsymbol{A}$ to matrix $\boldsymbol{C}$ and r rows to matrix $\boldsymbol{R}$, and build c-by-r matrix $\boldsymbol{U}$ minimizing $||\boldsymbol{A} - \boldsymbol{CUR}||_F$
- Often $c = r = k$

ICA definition

- **Setting**. Let $x_j \in \mathbb{R}$, $j=1,\ldots,n$ be observed random variables. Assume there exists n latent random variables $s_j \in \mathbb{R}$ and latent coefficients a_{ij} such that $x_j = \sum_i a_{ij} s_i$ for all j.
  - $\boldsymbol{x} = \boldsymbol{sA}$ and for T observations, $\boldsymbol{X} = \boldsymbol{SA}$ where $\boldsymbol{X}$ and $\boldsymbol{S}$ have T rows
- **Problem**. Find $\boldsymbol{A}$ and $\boldsymbol{s}$ given $\boldsymbol{x}$

ICA assumptions (important slide!)

- Original signals s_i are mutually **statistically independent**
- At most one original signal s_i is normally distributed
- The mixing matrix $\boldsymbol{A}$ is square and invertible
  - This is not necessary but simplifies the theory


x_1
x_2
x_3
x_4
s_1
s_2
s_3
s_4

Computing ICA: Central limit theorem

- Average of i.i.d. variables converges to normal distribution
  - $\sqrt{n}\left(\left(\frac{1}{n}\sum_{i=1}^{n} X_i\right) - \mu\right) \xrightarrow{d} N(0, \sigma^2)$ as $n \to \infty$
- Hence $(X_1 + X_2)/2$ is “more Gaussian” than X_1 or X_2 alone
  - For i.i.d. zero-centered non-Gaussian X_1 and X_2
- Hence, we can try to find components s that are “maximally non-Gaussian”

FastICA for one IC

- Noticing that $||\boldsymbol{w}||^2 = 1$ by constraint and taking infinite step update, we get

  $\boldsymbol{w} \leftarrow \mathrm{E}[(\boldsymbol{z}\boldsymbol{w}^\mathrm{T})^3\boldsymbol{z}] - 3\boldsymbol{w}$

  - Again set $\boldsymbol{w} \leftarrow \boldsymbol{w}/||\boldsymbol{w}||$ after every update
- Expectation has naturally to be estimated
- No theoretical guarantees but works in practice

Multiple components

- So far we have found only one component
  - To find more, remember that vectors $\boldsymbol{w}_i$ are orthogonal (whitening!)
- General idea:
  - Find one vector $\boldsymbol{w}$
  - Find second that is orthogonal to the first one
  - Find third that is orthogonal to the two previous ones, etc.

Putting it all together

- Start with random $\boldsymbol{B}$ and γ, choose learning rates δ and δ_γ
- Iterate until convergence
  - $\boldsymbol{y} \leftarrow \boldsymbol{B}\boldsymbol{x}$ and normalize $\boldsymbol{y}$ to unit variance
  - $\gamma_i \leftarrow (1 - \delta_\gamma)\gamma_{i-1} + \delta_\gamma \mathrm{E}[-\tanh(y_i)y_i + (1 - \tanh(y_i)^2)]$
    - if $\gamma_i > 0$, use super-Gaussian g; o/w sub-Gaussian g
  - $\boldsymbol{B} \leftarrow \boldsymbol{B} + \delta(\boldsymbol{I} + \sum_t \boldsymbol{g}(\boldsymbol{y}_t)^T \boldsymbol{y}_t)\boldsymbol{B}$

Graph Laplacians

- The **Laplacian matrix *L*** of a graph is the adjacency matrix subtracted from the degree matrix

$$\boldsymbol{L} = \boldsymbol{\Delta} - \boldsymbol{A} = \begin{pmatrix} \sum_{j \neq 1} a_{1,j} & -a_{1,2} & \cdots & -a_{1,n} \\ -a_{2,1} & \sum_{j \neq 2} a_{2,j} & \cdots & -a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ -a_{n,1} & -a_{n,2} & \cdots & \sum_{j \neq n} a_{n,j} \end{pmatrix}$$

- The Laplacian is symmetric and positive semi-definite
  - Undirected graphs
  - Has n real, non-negative, orthogonal eigenvalues

    $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \ldots \geq \lambda_n \geq 0$

Graph cuts using matrices

$$\text{RatioCut} = \sum_{i=1}^{k} \frac{W(C_i, V \setminus C_i)}{|C_i|} = \sum_{i=1}^{k} \frac{\boldsymbol{c}_i^T \boldsymbol{L} \boldsymbol{c}_i}{\|\boldsymbol{c}_i\|^2}$$

$$\text{NormalizedCut} = \sum_{i=1}^{k} \frac{W(C_i, V \setminus C_i)}{vol(C_i)} = \sum_{i=1}^{k} \frac{\boldsymbol{c}_i^T \boldsymbol{L} \boldsymbol{c}_i}{\boldsymbol{c}_i^T \boldsymbol{\Delta} \boldsymbol{c}_i}$$

Spectral clustering pseudo-code

Assume connected graph

Algorithm 16.1: Spectral Clustering Algorithm

Spectral Clustering $(\mathbf{D}, k)$:

1 Compute the similarity matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ ← Sparsify if needed

2 **if** *ratio cut* **then** $\mathbf{B} \leftarrow \mathbf{L}$

3 **else if** *normalized cut* **then** $\mathbf{B} \leftarrow \mathbf{L}^s$ or $\mathbf{L}^a$

4 Solve $\mathbf{B}\mathbf{u}_i = \lambda_i \mathbf{u}_i$ for $i = n, \ldots, n-k+1$, where $\lambda_n \leq \lambda_{n-1} \leq \cdots \leq \lambda_{n-k+1}$

5 $\mathbf{U} \leftarrow \begin{pmatrix} \mathbf{u}_n & \mathbf{u}_{n-1} & \cdots & \mathbf{u}_{n-k+1} \end{pmatrix}$

6 $\mathbf{Y} \leftarrow$ normalize rows of $\mathbf{U}$ using (16.19)

7 $\mathcal{C} \leftarrow \{C_1, \ldots, C_k\}$ via K-means on $\mathbf{Y}$

Definition of the BMF

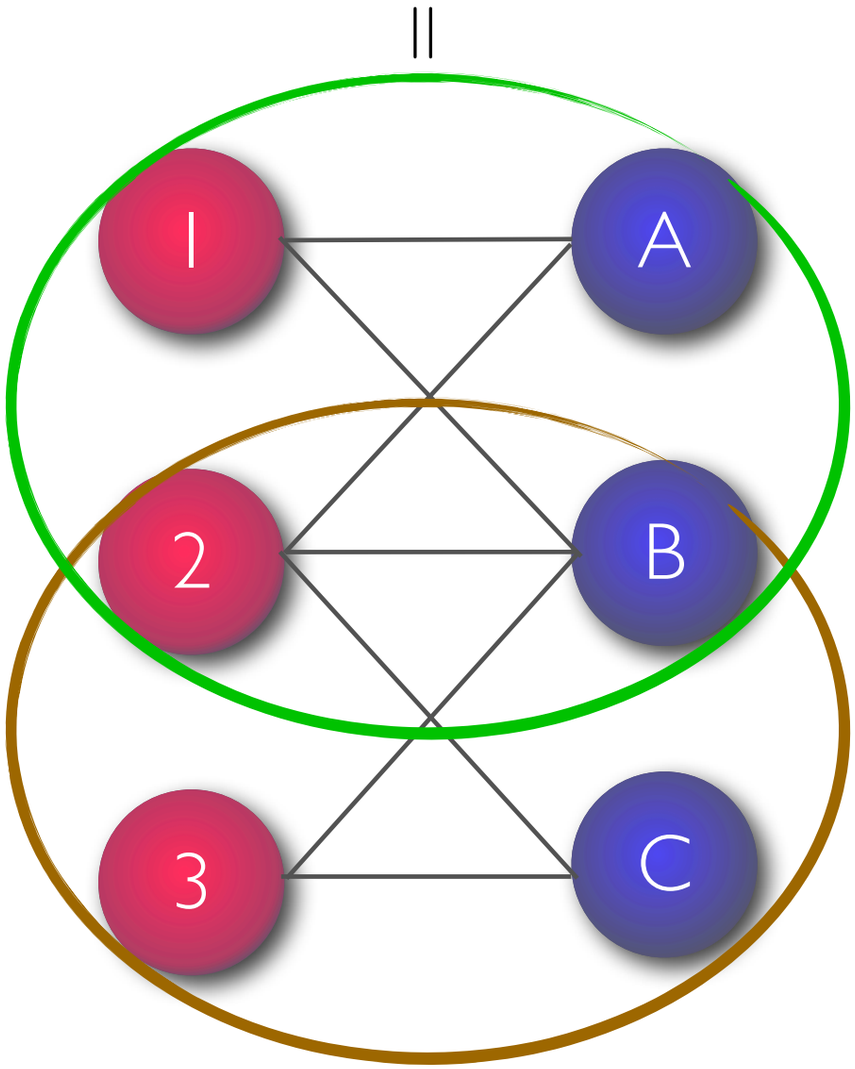
Boolean Matrix Factorization (BMF)

The (exact) **Boolean matrix factorization** of a binary matrix $\boldsymbol{A} \in \{0,1\}^{m \times n}$ expresses it as a Boolean product of two factor matrices, $\boldsymbol{B} \in \{0,1\}^{m \times k}$ and $\boldsymbol{C} \in \{0,1\}^{k \times n}$. That is $\boldsymbol{A} = \boldsymbol{B} \boxtimes \boldsymbol{C}$.

- Typically (in data mining), k is given, and we try to find $\boldsymbol{B}$ and $\boldsymbol{C}$ to get as close to $\boldsymbol{A}$ as possible
- Normally the optimization function is the squared Frobenius norm of the residual, $\|\boldsymbol{A} - (\boldsymbol{B} \boxtimes \boldsymbol{C})\|_F^2$
  - Equivalently, $|\boldsymbol{A} \oplus (\boldsymbol{B} \boxtimes \boldsymbol{C})|$ where
    - $|\boldsymbol{A}|$ is the sum of values of $\boldsymbol{A}$ (number of 1s for binary matrices)
    - $\oplus$ is the element-wise exclusive-or (1+1=0)
  - The alternative definition is more "combinatorial" in flavour

BMF and (quasi-)biclique covers

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

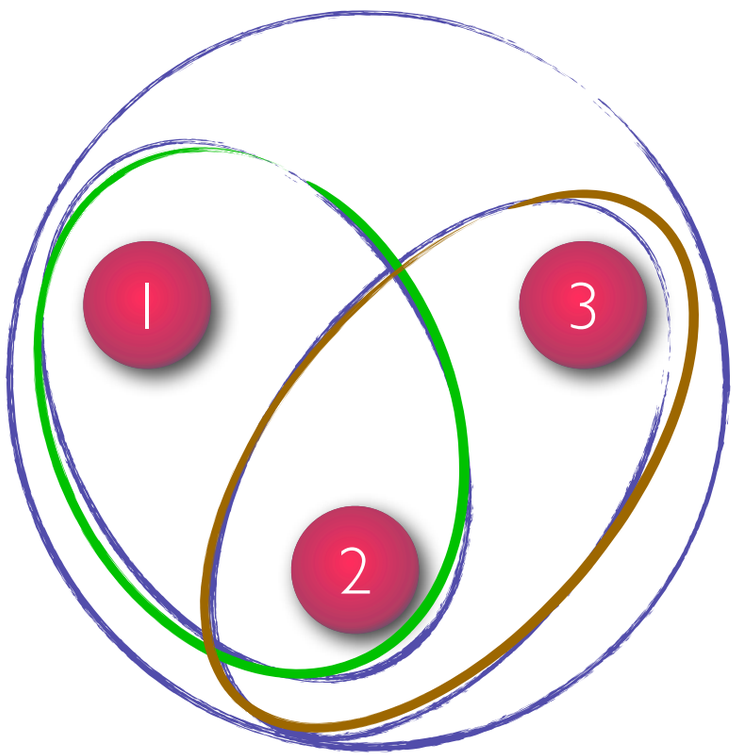


- A **biclique** is a complete bipartite graph
  - Each left-hand-side verted is connected to each right-hand-side vertex
- Each rank-1 binary matrix defines a biclique (subgraph)
  - If $\boldsymbol{v} \in \{0,1\}^m$ and $\boldsymbol{u} \in \{0,1\}^n$, then $\boldsymbol{v}\boldsymbol{u}^T$ is a biclique between $v_i \in V$ and $u_j \in U$ for which $\boldsymbol{v}_i = \boldsymbol{u}_j = 1$
- Exact BMF corresponds to covering each edge of the graph with at least one biclique
  - In approximate BMF, quasi-bicliques cover most edges

BMF and the Set Basis problem

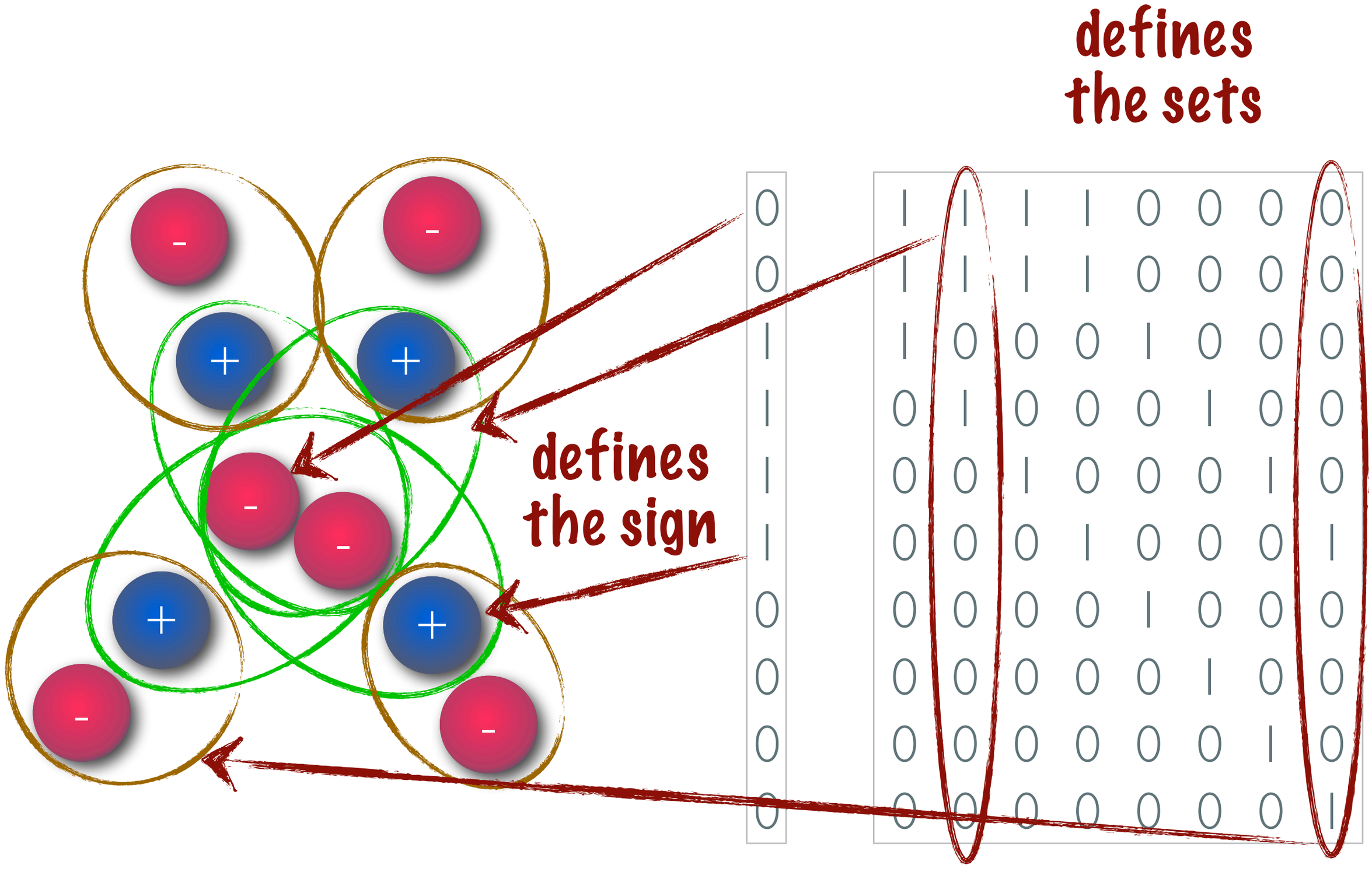
$$\begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

$$\equiv$$

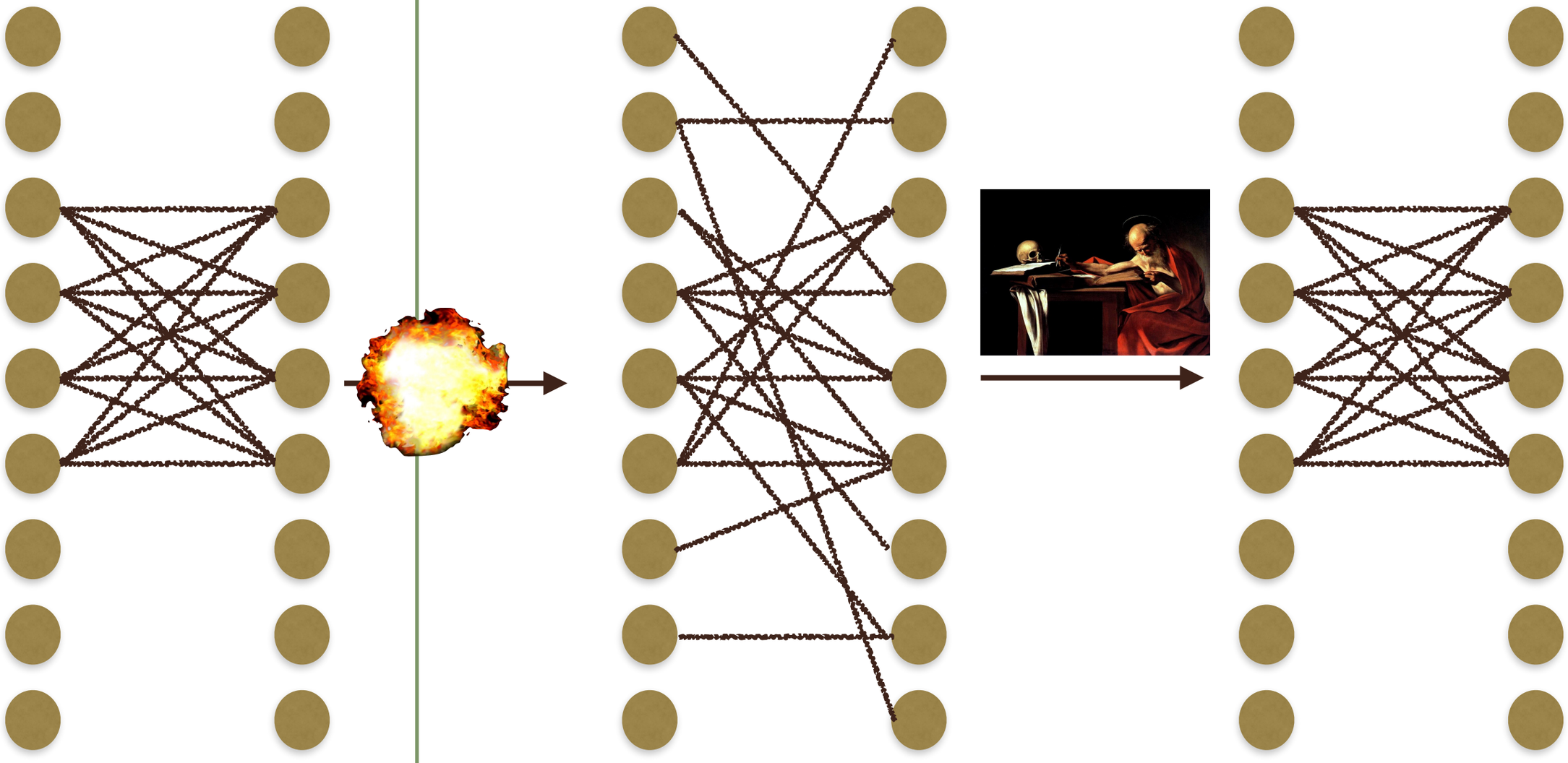


- In the **Set Basis** problem, we are given a set system $(U, \mathcal{S})$, and our task is to find collection $\mathcal{C} \subseteq 2^U$ such that we can cover each set $S \in \mathcal{S}$ with a union of some sets of $\mathcal{C}$
  - For each $S \in \mathcal{S}$, there is $\mathcal{C}_S \subseteq \mathcal{C}$ such that $S = \bigcup_{C \in \mathcal{C}_S} C$
- A set basis corresponds to exact BMF
  - The size of the smallest set basis is the Boolean rank
- N.B.: this is the same problem as covering with bicliques

Example of ±PSC and Basis Usage



Can we find the original pattern?



Planted partition model

- Let $G(\varphi, \boldsymbol{P})$ be a random graph distribution where $\varphi : V \to \{1,\ldots,k\}$ partition the vertices to k classes and $\boldsymbol{P}=(p_{ij})$ is a k-by-k matrix with $p_{ij} \in [0,1]$. Include edge (i, j) with probability $p_{\varphi(i)\varphi(j)}$.
- **Example**: planted clique. Let $\varphi(v) = 1$ iff v is in the clique. Set $p_{11} = 1$ and $p_{ij} = p$ elsewhere
- **Problem**. Given a sample $\boldsymbol{G}'$ from $G(\varphi, \boldsymbol{P})$, find a partition φ' s.t. $\varphi'(\boldsymbol{v}) = \varphi'(\boldsymbol{u})$ iff $\varphi(\boldsymbol{v}) = \varphi(\boldsymbol{u})$

Planted partition results

- Now if $||\boldsymbol{g}_u - \boldsymbol{g}_v||$ is large enough when $\varphi(v) \neq \varphi(u)$, we can find φ
  - Depends on the above error bounds
- With more complicated error bounds we get:
  - If s is the size of a planted clique, then there is a constant c s.t. for sufficiently large n we can recover φ with probability $1 - \delta$ if

$$\frac{1-p}{p} > c\left(\frac{n}{s^2} + \frac{\log(n/\delta)}{s}\right)$$

Maximum clique as rank minimization

- Maximum n-vertex clique in graph $G = (V, E)$ can be found with the following program

$$\min \quad \operatorname{rank}(\boldsymbol{X})$$

A clique is a rank-1 submatrix

$$\text{s.t.} \quad \sum_{i \in V} \sum_{j \in V} x_{ij} \geq n^2$$

of size n-by-n

Proper submatrix

$$x_{ij} = 0 \quad \text{if } \{i, j\} \notin E \text{ and } i \neq j$$

Symmetric

$$\boldsymbol{X} = \boldsymbol{X}^T$$

No entry larger than 1

$$\boldsymbol{X} \in [0, 1]^{V \times V}$$

Nuclear norm relaxation

- The rank minimization problem is NP-hard
- We can relax it to nuclear norm minimization:

$$\min \quad \|\boldsymbol{X}\|_*$$

$$\text{s.t.} \quad \sum_{i \in V} \sum_{j \in V} x_{ij} \geq n^2$$

← can be replaced with 1

$$x_{ij} = 0 \quad \text{if } \{i, j\} \notin E \text{ and } i \neq j$$

- The maximum clique is a valid solution and the unique optimizer under certain conditions
  - When this is the case, we can find the clique

Destructive noise models for bicliques

- So far we've added each edge independently with probability p
  - Erdős–Rényi random graph model
  - We can also follow the preferential attachment model
    - Barabási–Albert random graph model
    - Some vertices have big changes on neighbors, others less
      - If the noise follows the B–A model, it can't have large bicliques ⇒ easy

Results

- Erdős–Rényi: The minimum size of the original biclique $\zeta = \log(NM)$
- Barabási–Albert: $\log N \ll \zeta \ll \sqrt{N}$

On exam

Format & basic info

- Written exam
  - 28 July 2015 from 12:00–14:00
    - Times are sharp!
  - Lecture hall 001, building E1 3

What you can and cannot bring

- You can (must) bring
  - writing equipments & student ID
  - one (1) A4-sized “cheat sheet” paper
- You cannot bring (use)
  - electronic devices (incl. phones and pocket calculators and electric pencil sharpeners)
  - any other notes than the cheat sheet (incl. lecture slides, assignments, etc.)

Cheat sheet

- **Must contain your name!**
- A4-sized paper, text can be on both sides
- Any content is OK (as long as its legal)
  - Use your discretion what you think is important or consider hard
- Can be made with computer or be hand-written (or with typewriter)

What is covered in the exam?

- All lectures between 21 April to 21 July
  - Lecture on 21 July is a wrap-up, no new contents
- All pen-and-paper and first two programming assignments
- The chapters of books and articles cited in the lecture slides

What kind of questions are there in the exam?

- Simple mathematical proofs
  - Similar to those in homework assignments
- Developing variations of presented algorithms
  - “Explain how would you compute ABC decomposition with the following constraints”
- Short texts or longer essays comparing different decomposition methods and/or explaining their use cases and interpretations
  - “What are the main differences between ABC and XYZ?” “Given this-and-that kind of data, how would you interpret its ABC decomposition?”
- Short questions about features and properties of decompositions and methods
  - “Explain briefly the main idea behind algorithms computing ABC.” “True or false: computing the optimal XYZ decomposition (w.r.t. the Frobenius norm) is NP-hard.”

Exam checking day

- 31 July from 12:15 to 14:00
  - D5 rotunda (E1 4, 4th floor, left from elevator)
  - your only chance

Re-Exam

- Open for everybody
  - Bonus points count, better of two exams
- **You must register** via email **by 7 August 12 noon**
- Date & place TBD (late September)

Advertisements

Follow-up seminar

- I’m planning for a follow-up seminar in next semester
  - Topics involve deeper dive into new matrix (and maybe tensor) factorization methods for data analysis
    - Current plan (subject to change): Block seminar with two days of presentations in January/early February
- Limited attendance with first-come-first-served basis
  - Send me e-mail if you want to get notified as soon as registration to the seminar is possible

HiWi & MSc student positions

- I have positions for HiWis & MSc students on data mining
  - Matrices, tensors, and other stuff – tailored to taste
- This course is important evaluation point
  - I will not consider any application before I know the results of this course

    ⇒ If interested, **apply in late August/early September**

Ask Me Anything

Johnson–Lindenstrauss lemma

- Finding the decomposition can be expensive
- Decompositions give only *global* guarantees
  - Any pair of points can have very different distances
- Can we guarantee *local* similarity?

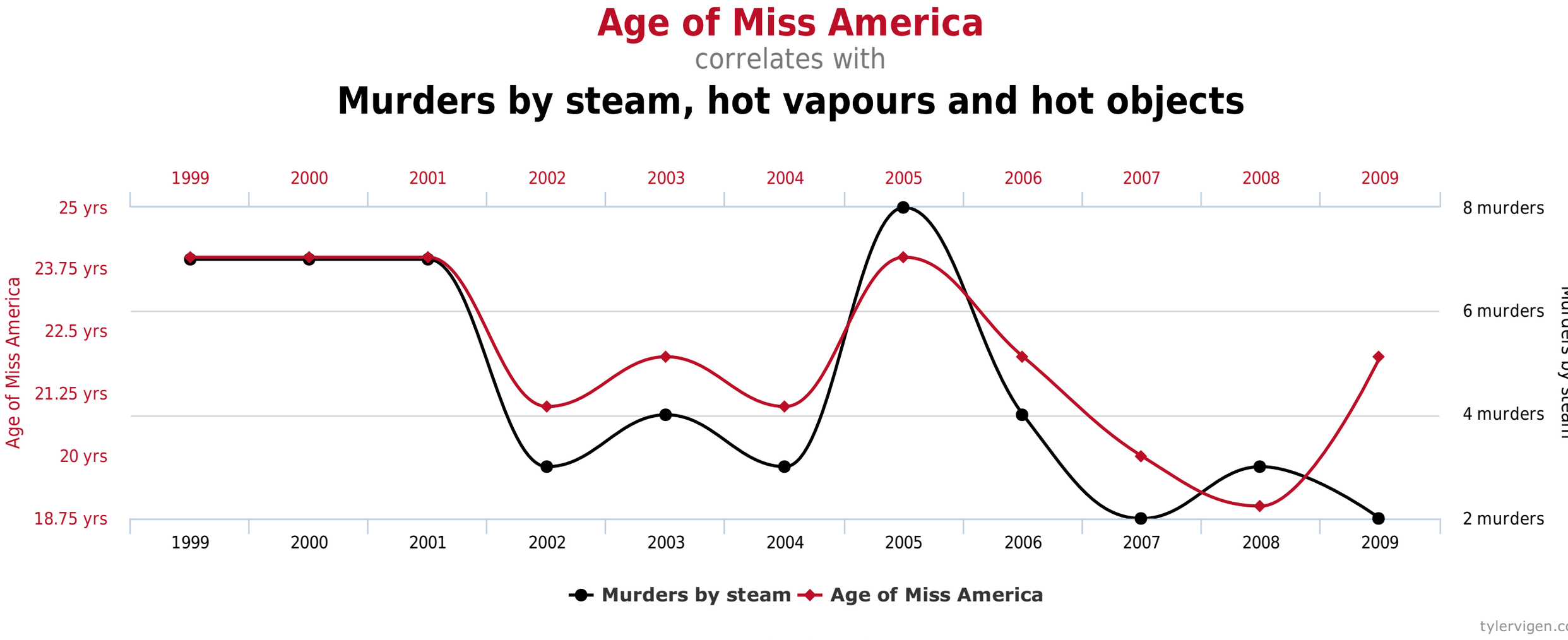
Johnson–Lindenstrauss lemma. Given $\varepsilon > 0$ and an integer n, let k be a positive integer such that $k \geq k_0 = O(\varepsilon^{-2}\log n)$. For every set X of n points in $\mathbb{R}^d$ there exists F: $\mathbb{R}^d \to \mathbb{R}^k$ such that for all $\boldsymbol{x}_i, \boldsymbol{x}_j \in X$

$$(1-\varepsilon)\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2 \leqslant \left\|\mathsf{F}(\mathbf{x}_i) - \mathsf{F}(\mathbf{x}_j)\right\|^2 \leqslant (1+\varepsilon)\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2$$

How to find the projections?

- We need to find an k-by-d matrix $\boldsymbol{R} = (r_{ij})$ such that function $\boldsymbol{x} \mapsto \boldsymbol{R}\boldsymbol{x}$ satisfies JL
- Remarkably, if we select $r_{ij} \sim \mathrm{N}(0,1)$, $\boldsymbol{R}$ satisfies JL with high probability
  - That is, JL holds for *all* points of X with high probability
- Achlioptas has show that we can also select $\Pr[r_{ij} = 1] = 1/2$ and $\Pr[r_{ij} = -1] = 1/2$ or $\Pr[r_{ij} = 1] = 1/6$, $\Pr[r_{ij} = 0] = 2/3$, $\Pr[r_{ij} = -1] = 1/6$
  - Sparse matrix

Spurious correlations



http://www.tylervigen.com/spurious-correlations