

Chapter XI: Two Matrix Factorizations

Information Retrieval & Data Mining
Universität des Saarlandes, Saarbrücken
Winter Semester 2013/14

Chapter XI: Two Matrix Factorizations

1. Non-Negative Matrix Factorization

1.1. Idea and motivation

1.2. Algorithms

2. Boolean Matrix Factorization

2.1. Idea and motivation

2.2. Algorithms

Non-Negative Matrix Factorization

- Recall SVD: $A = U\Sigma V^T$
 - Σ is non-negative diagonal, but U and V can contain arbitrary real values
 - If the data is non-negative by nature and “direction of variance” is not a good interpretation of the data, SVD can be hard to interpret
- Non-negative data can be
 - Counting data (e.g. term frequencies)
 - Physical measurements (degrees of Kelvin, height, ...)
 - Images
 - ...

Definition of NMF

- Given a nonnegative n -by- m matrix X (i.e. $x_{ij} \geq 0$ for all i and j) and a positive integer k , find an n -by- k nonnegative matrix W and a k -by- m nonnegative matrix H s.t. $\|X - WH\|_F$ is minimized.
 - If $k = \min(n, m)$, we can do $W = X$ and $H = I_m$ (or vice versa)
 - Otherwise the problem is NP-hard
- If either W or H is fixed, we can find the other factor matrix in polynomial time
 - Which gives us our first algorithm...

The alternating least squares (ALS)

- Let's forget the nonnegativity constraint for a while
- The alternating least squares algorithm is the following:
 - Initialize W to a random matrix
 - **repeat**
 - Fix W and find H s.t. $\|X - WH\|_F$ is minimized
 - Fix H and find W s.t. $\|X - WH\|_F$ is minimized
 - **until** convergence
- For **unconstrained least squares** we can use $H = W^+X$ and $W = XH^+$
- ALS will typically converge to *local optimum*

NMF and ALS

- With the nonnegativity constraint the pseudo-inverse doesn't work
 - The problem is still **convex** with either of the factor matrices fixed (but not if both are free)
 - We can use **constrained convex optimization**
 - In theory, polynomial time
 - In practice, often too slow
- Poor man's nonnegative ALS:
 - Solve $\mathbf{H}$ using pseudo-inverse
 - Set all $h_{ij} < 0$ to 0
 - Repeat for $\mathbf{W}$

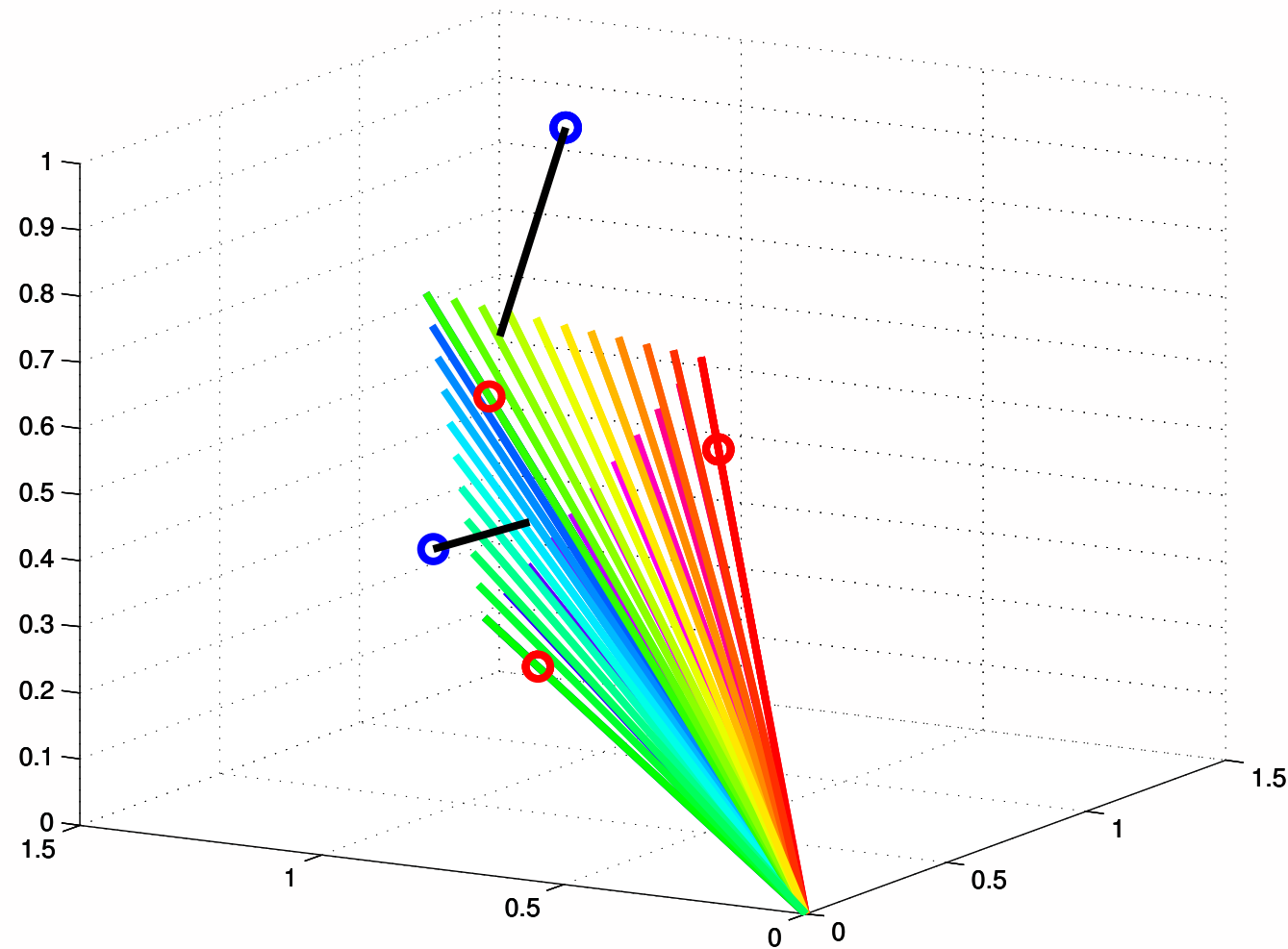
Geometry of NMF

NMF factors

Data points

Convex cone

Projections



Multiplicative update rules

- Idea: update W and H in small steps towards the locally optimum solution
 - Honor the non-negativity constraint
 - Lee & Seung, *Nature*, '99:
 1. Initialize W and H randomly to non-negative matrices
 2. **repeat**
 - 2.1. $H = H .* (W^T X) ./ (W^T W H + \varepsilon)$
 - 2.2. $W = W .* (X H^T) ./ (W H H^T + \varepsilon)$
 3. **until** convergence in $\|X - WH\|_F$
- Here $.*$ is element-wise product, $(A .* B)_{ij} = a_{ij} * b_{ij}$, and $./$ is element-wise division, $(A ./ B)_{ij} = a_{ij} / b_{ij}$
- Little value ε is added to avoid division by 0

Discussion on multiplicative updates

- If W and H are initialized to strictly positive matrices, they stay strictly positive throughout the algorithm
 - Multiplicative form of updates
- If W and H have zeros, the zeros stay
- Converges slowly
 - And has issues when the limit point lies in the boundary
- Lots of computation per update
 - Clever implementation helps
 - Simple to implement

Discussion on multiplicative updates

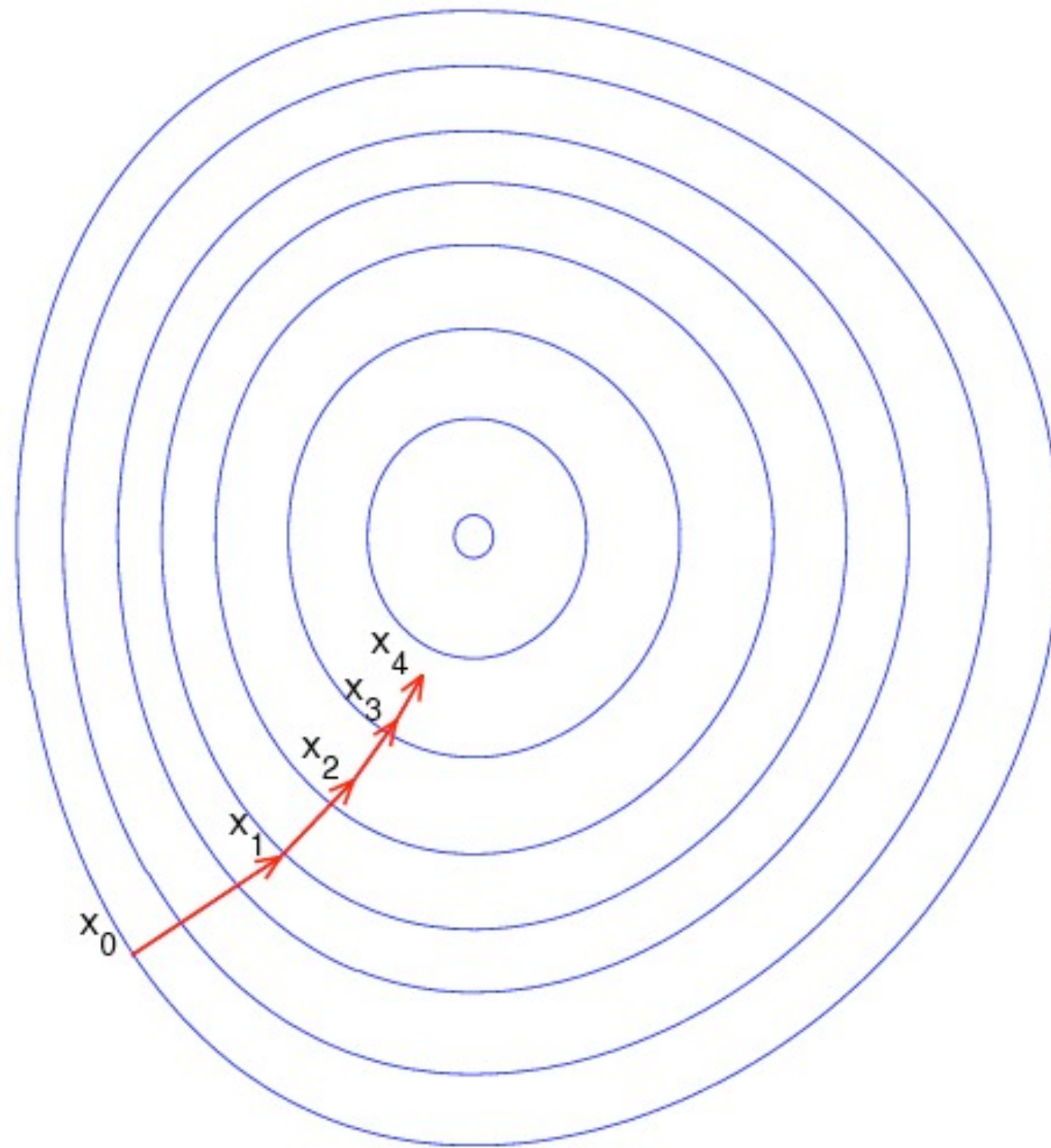
- If W and H are initialized to strictly positive matrices, they stay strictly positive throughout the algorithm
 - Multiplicative form of updates
- If W and H have zeros, the zeros stay
- Converges slowly
 - And has issues when the limit point lies in the boundary
- Lots of computation per update
 - Clever implementation helps
 - Simple to implement

Gradient descent

- Consider the representation error as a function of $\mathbf{W}$ and $\mathbf{H}$
 - $f: \mathbb{R}^{n \times k} \times \mathbb{R}^{k \times m} \rightarrow \mathbb{R}_+, f(\mathbf{W}, \mathbf{H}) = \|\mathbf{X} - \mathbf{WH}\|_F^2$
 - We can compute the partial derivatives $\partial f / \partial \mathbf{W}$ and $\partial f / \partial \mathbf{H}$
- **Observation:** The biggest decrease in f at point $(\mathbf{W}, \mathbf{H})$ happens at the opposite direction of the gradient
 - But this only holds in an ε -neighborhood of $(\mathbf{W}, \mathbf{H})$
 - Therefore, we make small steps opposite to gradient and re-compute the gradient

Example of gradient descent

Image: Wikipedia



NMF and gradient descent

Step size

1. Initialize W and H randomly to non-negative matrices
2. **repeat**
 - 2.1. $H = H - \varepsilon_H \partial f / \partial H$
 - 2.2. $W = W - \varepsilon_W \partial f / \partial W$
3. **until** convergence in $\|X - WH\|_F$

Step size

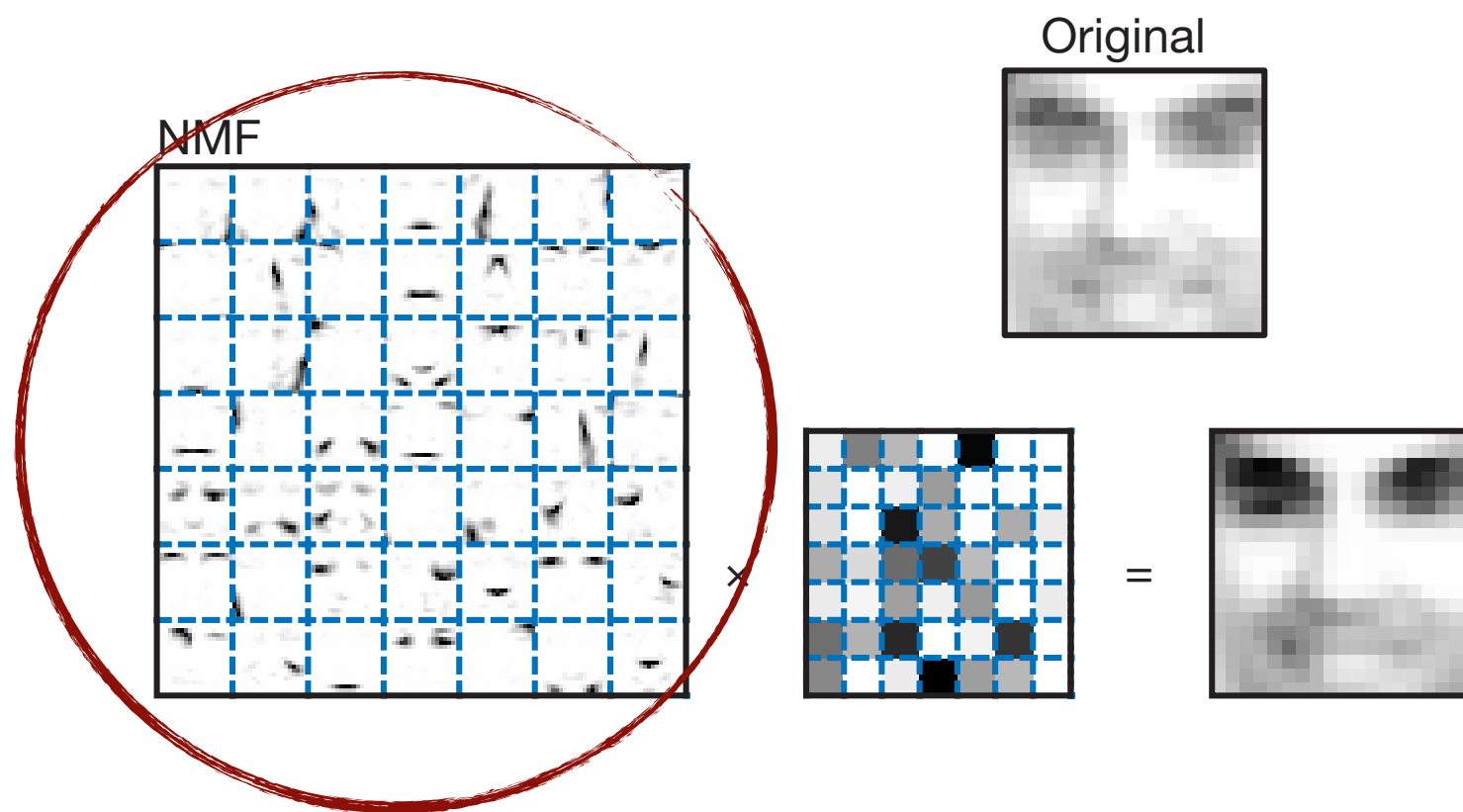
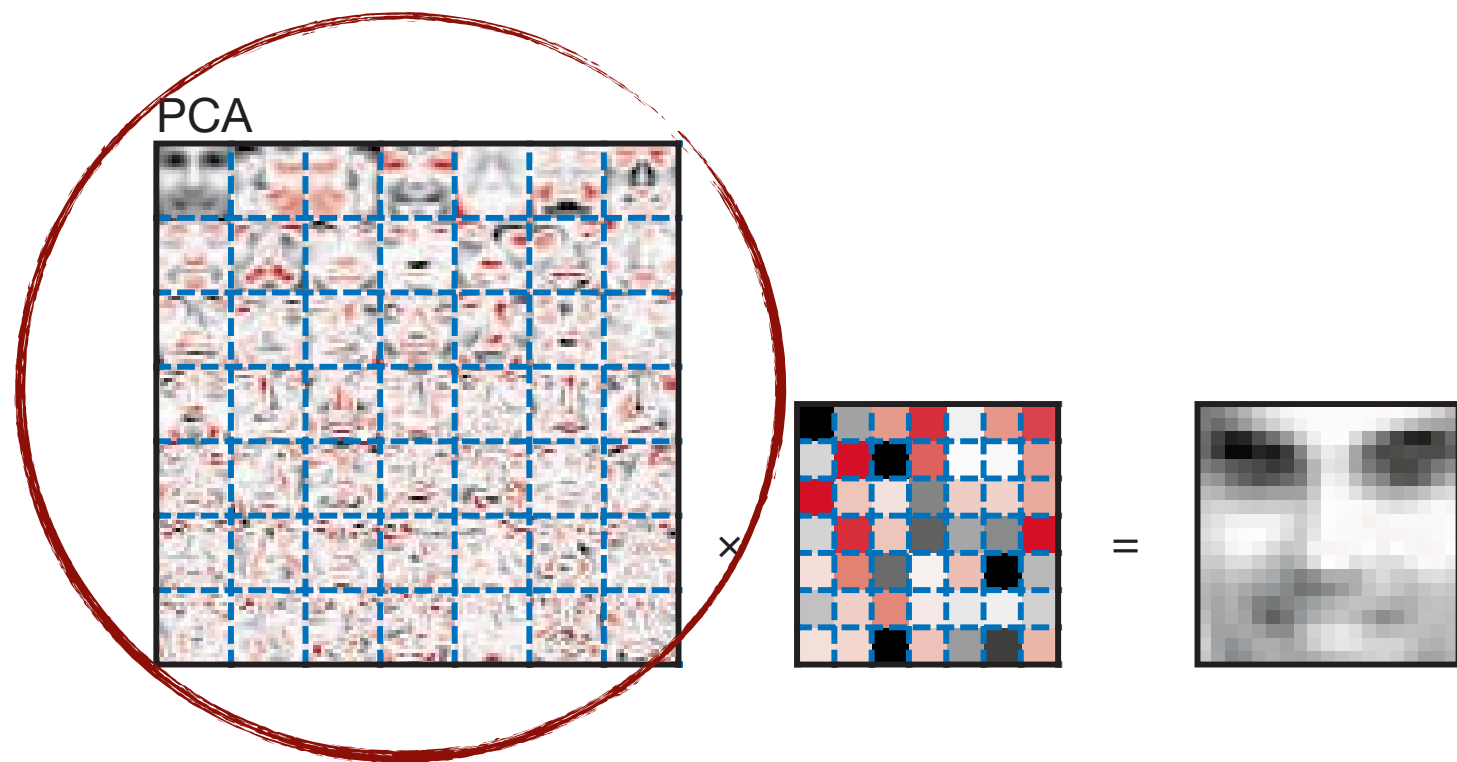
Issues with gradient descent

- Step sizes are important
 - Too big step size: error increases, not decrease
 - Too small step size: very slow convergence
 - Fixed step sizes don't work
 - Have to adjust somehow
 - Lots of research work put on this
- Ensuring the non-negativity
 - The updates can make factors negative
 - Easiest option: change all negative values to 0 after each update
- Updates are expensive
- Multiplicative update is a type of gradient descent
 - Essentially, the step size is adjusted

ALS vs. gradient descent

- Both are *general* techniques
 - Not tied to NMF
- More general version of ALS is called **alternating projections**
 - Like ALS, but not tied to least-squares optimization
 - We must know how to optimize one factor given the other
 - Or we can approximate this, too...
- In gradient descent function must be derivable
 - (Quasi-)Newton methods study also the second derivative
 - Even more computationally expensive
 - Stochastic gradient descent updates random parts of factors
 - Computationally cheaper but can yield slower convergence

NMF example: Face recognition



Boolean Matrix Factorization



Long-haired	1	1	0
Well-known	1	1	1
Male	0	1	1

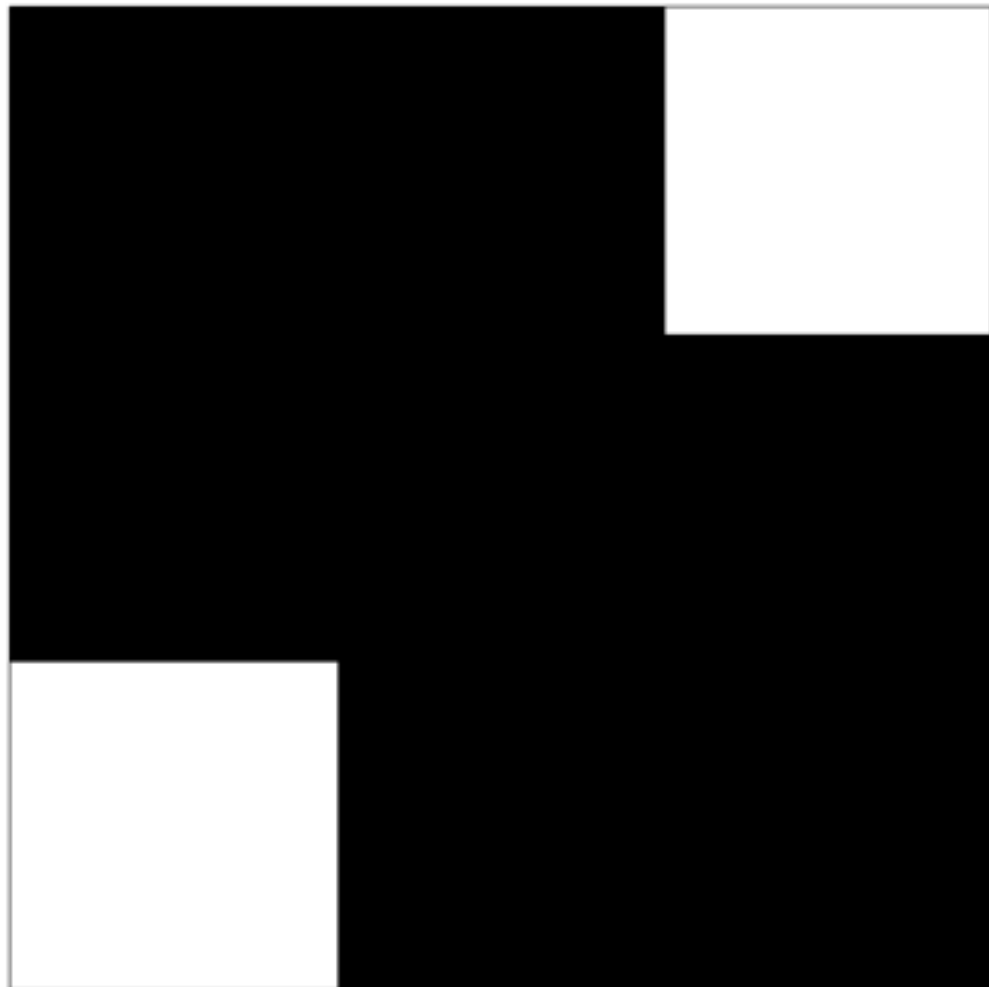
Can we find the groups?

- The data seems to have some patterns
 - Can we recover them?

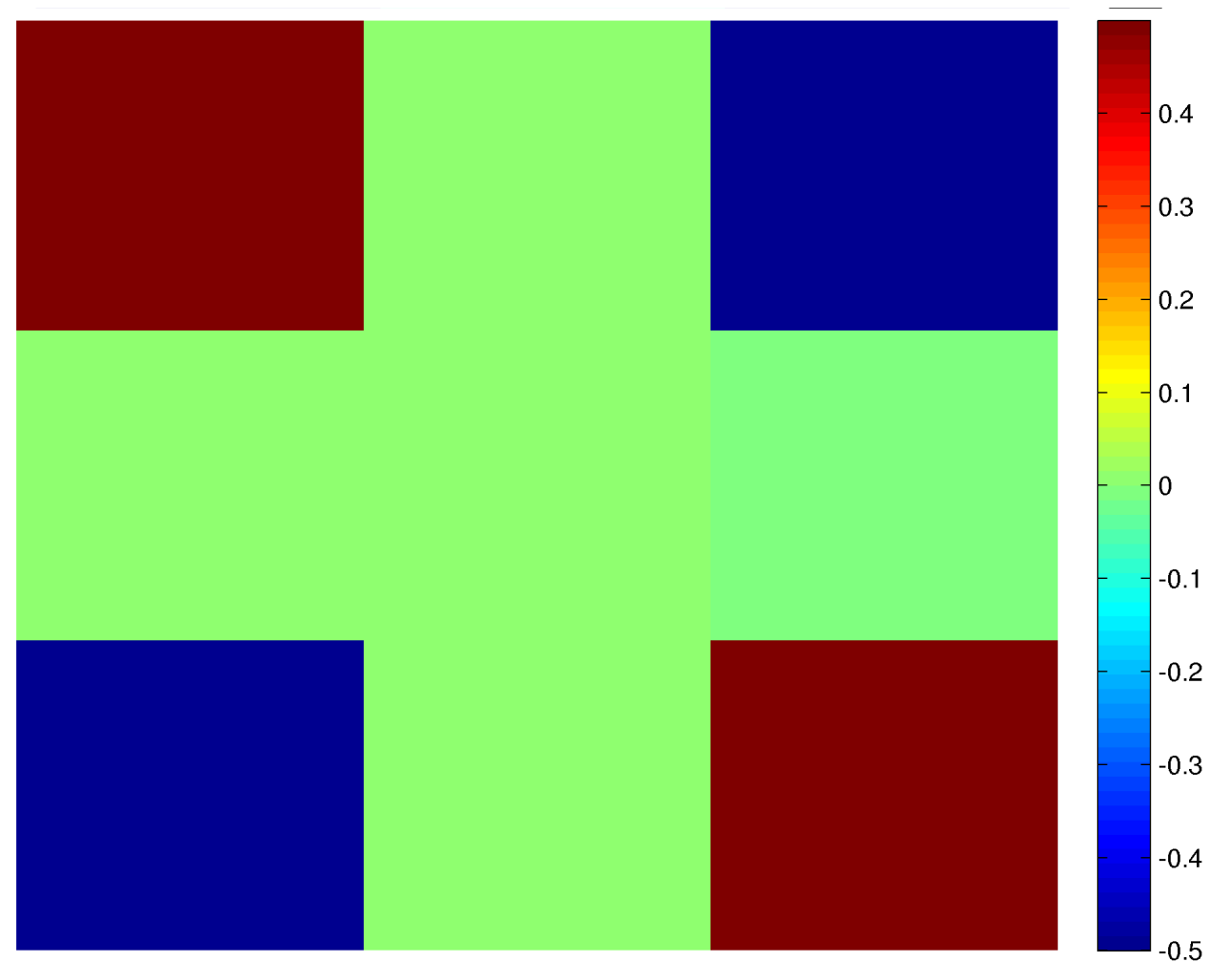
Long-haired	1	1	0
Well-known	1	1	1
Male	0	1	1

SVD?

Not good!



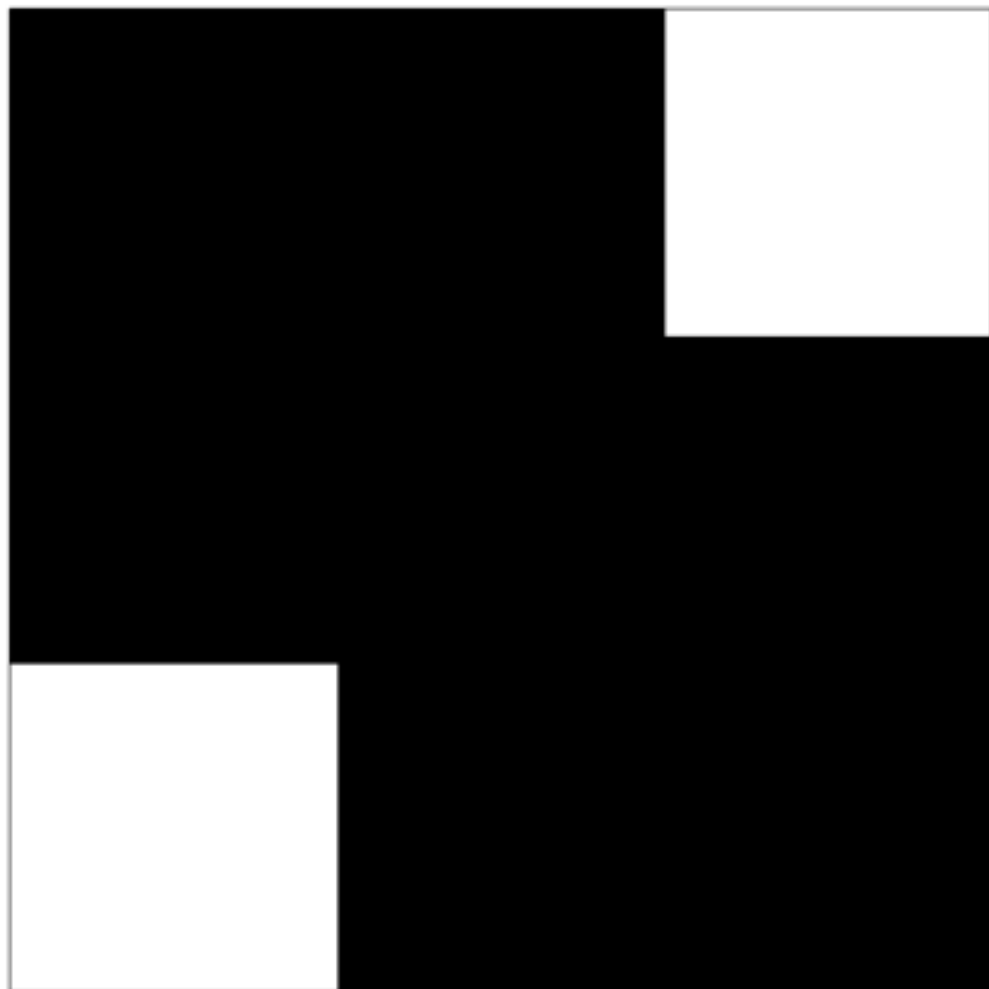
The data



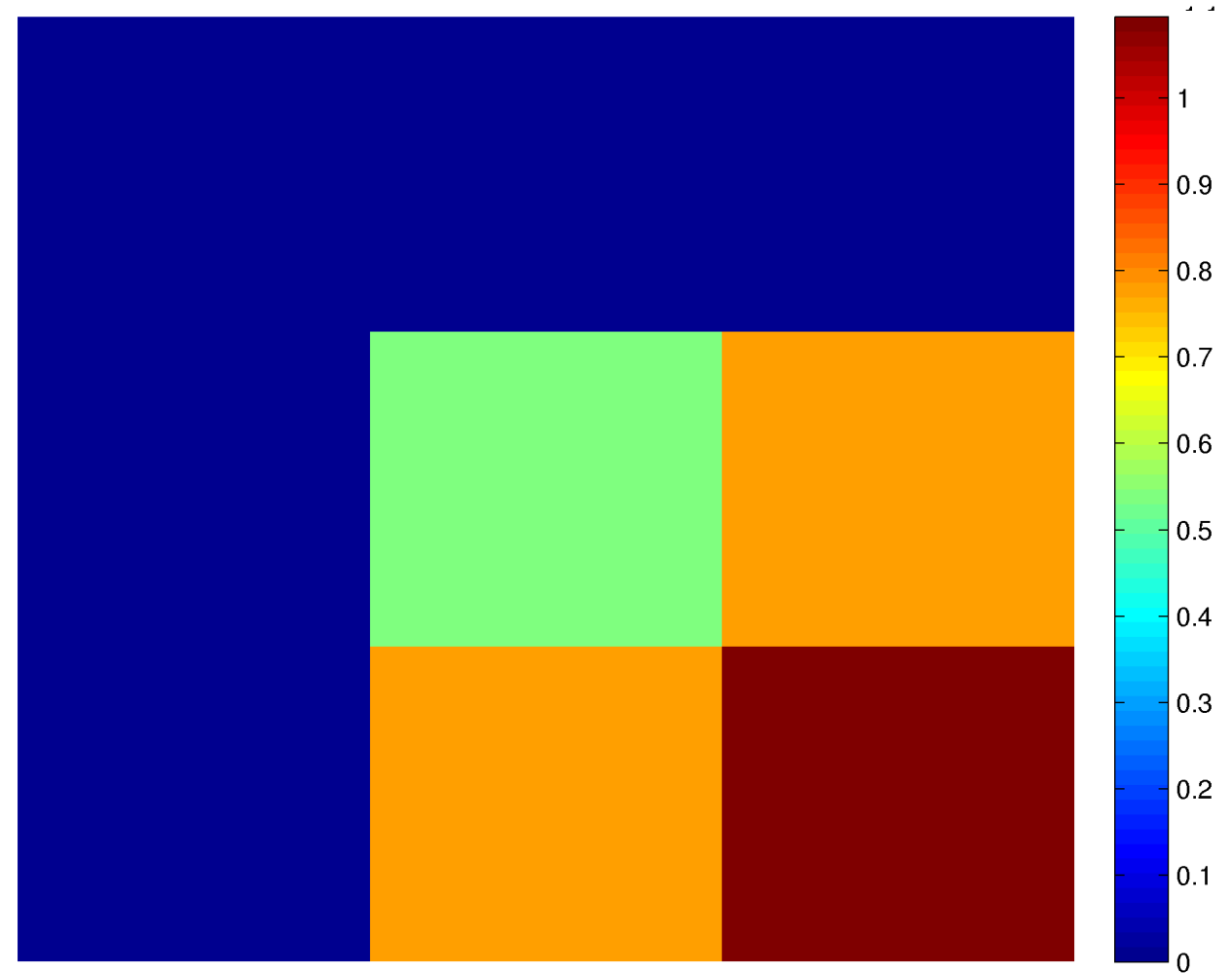
$$\mathbf{U}_2 \mathbf{\Sigma}_{2,2} \mathbf{V}_2^T$$

NMF?

Better, but not perfect



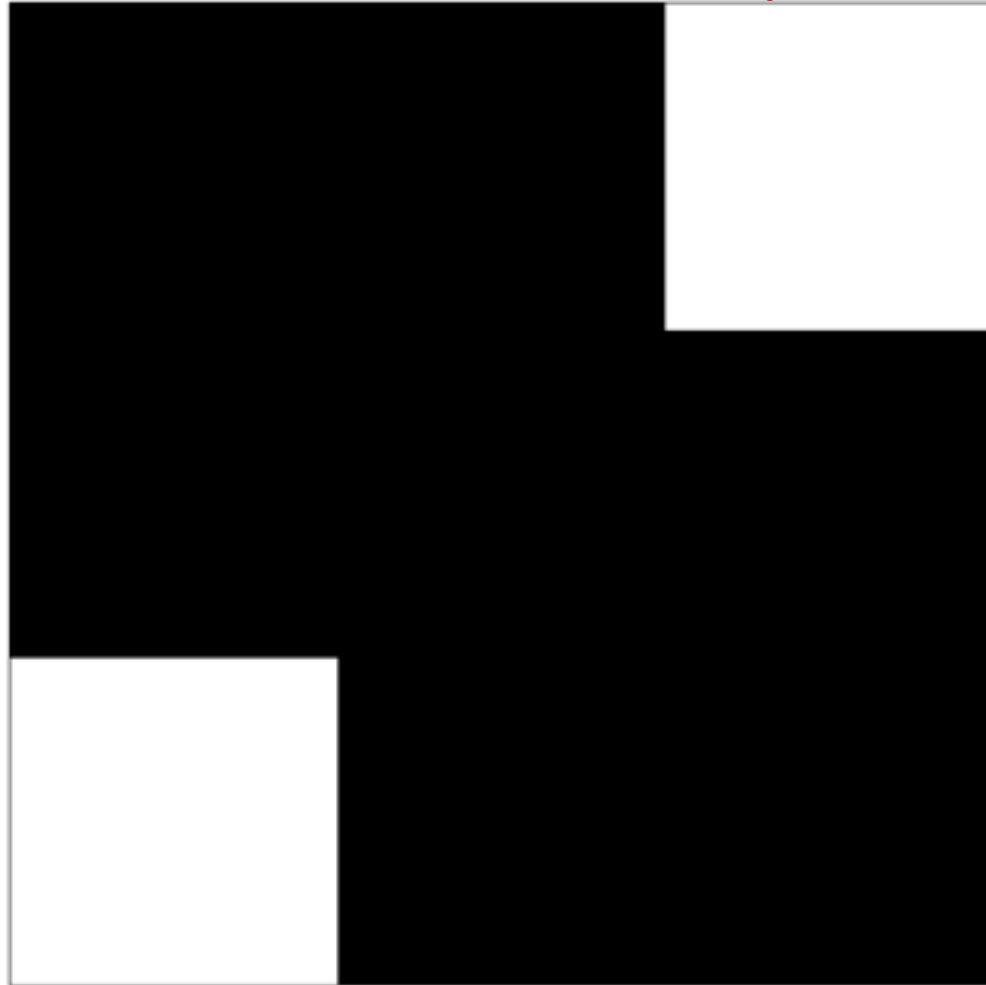
The data



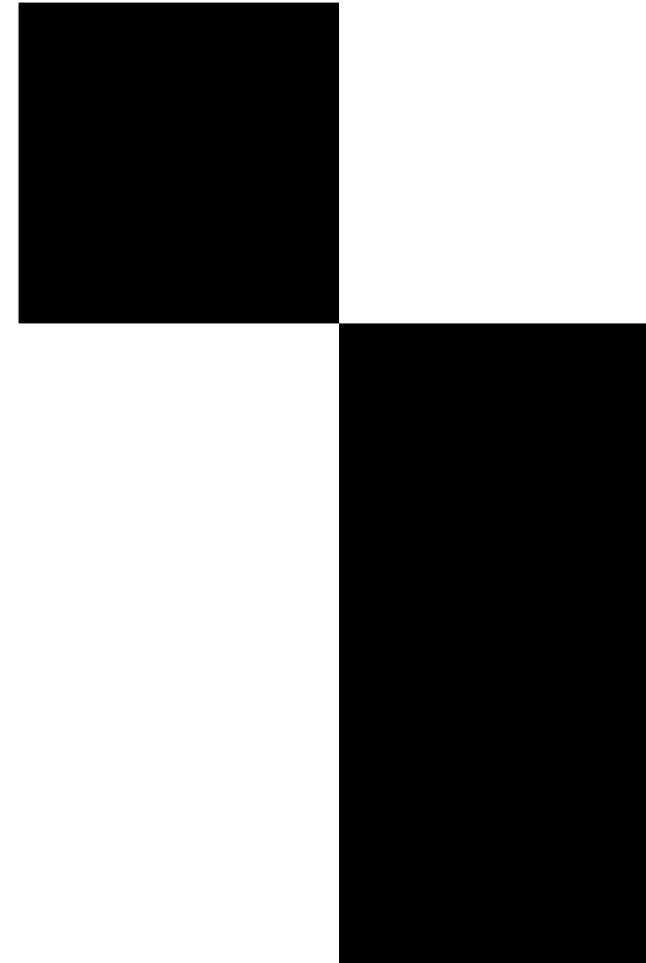
W_2H_2

Clustering?

Why is  assigned to one group but not to the other?

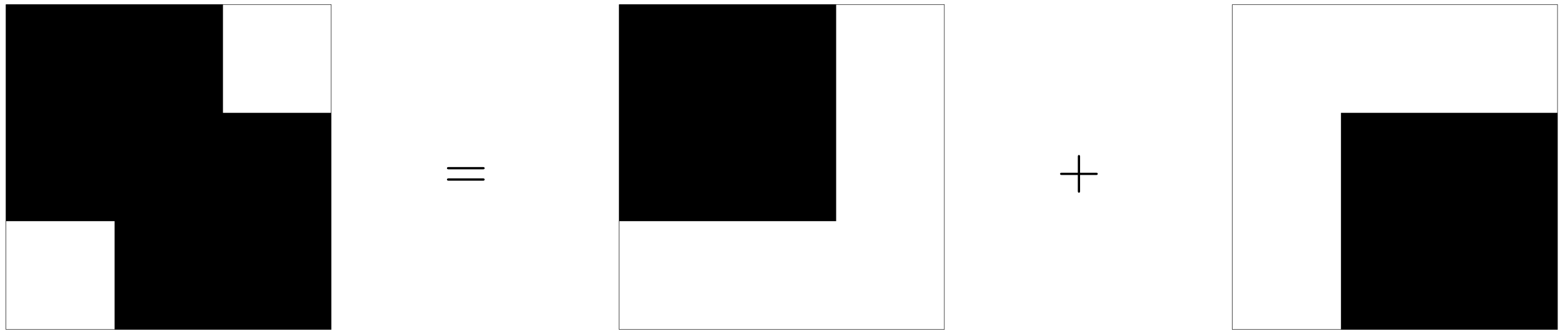


The data



Cluster assignment matrix

Boolean factors



- We would like to have rank-1 factors like this
- Problem: the sum of right-hand matrices is not the left-hand matrix
- Solution: don't care about multiplicity
 - Define $1 + 1 = 1$

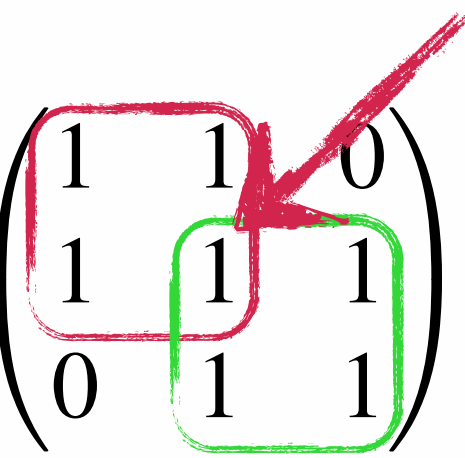
Boolean matrix multiplication and factorization

- The **Boolean matrix multiplication** of two binary matrices A and B is the binary matrix C s.t.

$$(C)_{ij} = (A \circ B)_{ij} = \bigvee_{l=1}^k a_{il} b_{lj}$$

- In **Boolean matrix factorization (BMF)** we are given a binary matrix X and rank k and we want to find binary factor matrices A and B that minimize $(\|X - A \circ B\|_F)^2$

BMF example

$$\mathbf{X} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix} = \begin{pmatrix} \boxed{1} & \boxed{1} & 0 \\ \boxed{1} & \boxed{1} & 1 \\ 0 & \boxed{1} & \boxed{1} \end{pmatrix} = \mathbf{A} \circ \mathbf{B}$$


$$(1 \quad 1 \quad 0) = \mathbf{b}_1$$

$$\mathbf{a}_1 = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \begin{pmatrix} \boxed{1} & \boxed{1} & 0 \\ \boxed{1} & \boxed{1} & 0 \\ 0 & 0 & 0 \end{pmatrix} = \mathbf{a}_1 \mathbf{b}_1$$

$$(0 \quad 1 \quad 1) = \mathbf{b}_2$$

$$\mathbf{a}_2 = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 \\ 0 & \boxed{1} & \boxed{1} \\ 0 & \boxed{1} & \boxed{1} \end{pmatrix} = \mathbf{a}_2 \mathbf{b}_2$$

Solving BMF

- Finding the least-error BMF is NP-hard
 - Also finding the Boolean rank is NP-hard
- Fixing one factor doesn't help
 - Finding $\mathbf{B}$ that minimizes $\|\mathbf{X} - \mathbf{A} \circ \mathbf{B}\|_F$ when $\mathbf{X}$ and $\mathbf{A}$ are given is still NP-hard
- All problems are also very hard to approximate
- Practical algorithms rely on greedy heuristics to solve the problem

Summary

- Two new matrix factorization methods
 - Though we've seen something similar earlier
- Both work on anti-negative semi-rings
 - No subtractions here!
- Both are motivated by interpretability and the ability to find different types of structures than what SVD finds