

Chapter 5

Decompositions for Combinatorial Structures

Part II: Boolean matrix factorization



Outline

- 1 Warm-Up
- 2 What is BMF
- 3 BMF vs. other three-letter abbreviations
- 4 Binary matrices, tiles, graphs, and sets
- 5 Computational Complexity
- 6 Algorithms
- 7 Wrap-Up

An example

- Let us consider a data set of people and their traits
 - ▶ People: Alice, Bob, and Charles
 - ▶ Traits: Long-haired, well-known, and male

An example

- Let us consider a data set of people and their traits
 - ▶ People: Alice, Bob, and Charles
 - ▶ Traits: Long-haired, well-known, and male



long-haired



well-known



male



An example

- Let us consider a data set of people and their traits
 - ▶ People: Alice, Bob, and Charles
 - ▶ Traits: Long-haired, well-known, and male



long-haired



well-known



male



An example

- Let us consider a data set of people and their traits
 - ▶ People: Alice, Bob, and Charles
 - ▶ Traits: Long-haired, well-known, and male



long-haired
well-known
male



An example



long-haired
well-known
male



- We can write this data as a binary matrix

An example



long-haired	1	1	0
well-known	1	1	1
male	0	1	1

- We can write this data as a binary matrix

An example





long-haired	1	1	0
well-known	1	1	1
male	0	1	1

- We can write this data as a binary matrix
- The data obviously has two groups of people and two groups of traits

An example



long-haired	1	1	0
well-known	1	1	1
male	0	1	1

- We can write this data as a binary matrix
- The data obviously has two groups of people and two groups of traits
 - ▶  and  are long-haired and well-known

An example



long-haired	1	1	0
well-known	1	1	1
male	0	1	1

- We can write this data as a binary matrix
- The data obviously has two groups of people and two groups of traits





▶  and  are long-haired and well-known

▶  and  are well-known males

An example

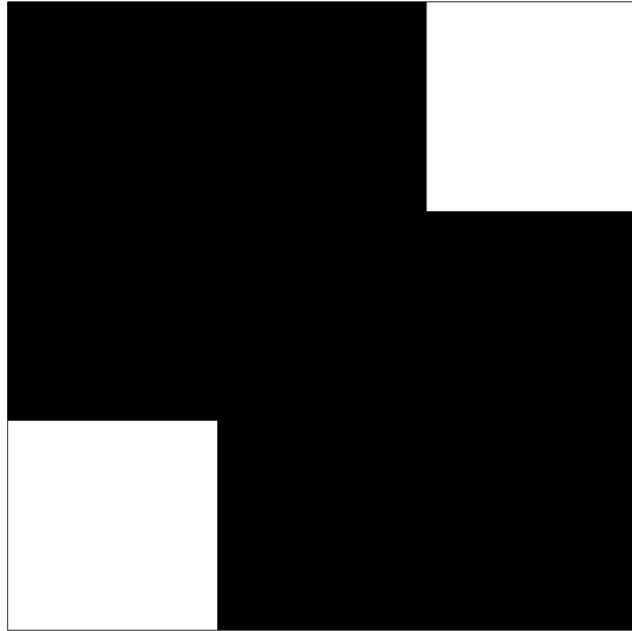


long-haired	1	1	0
well-known	1	1	1
male	0	1	1

- We can write this data as a binary matrix
- The data obviously has two groups of people and two groups of traits
 - ▶  and  are long-haired and well-known
 - ▶  and  are well-known males
- Can we find these groups automatically (using matrix factorization)?

SVD?

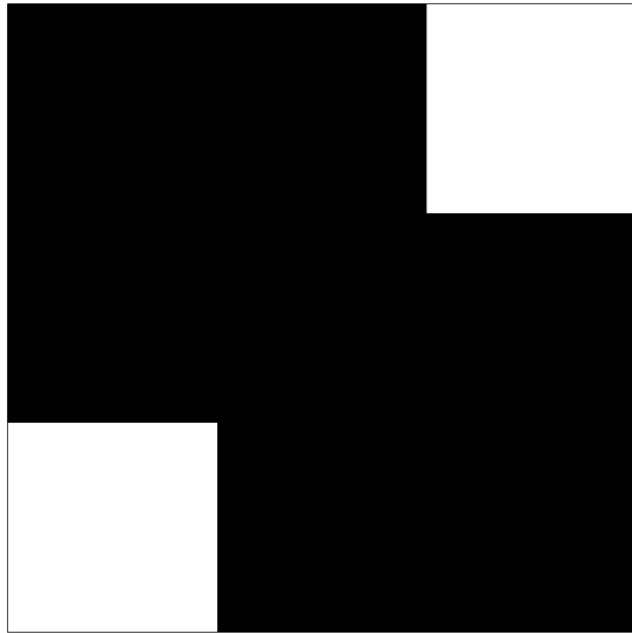
- Could we find the groups using SVD?



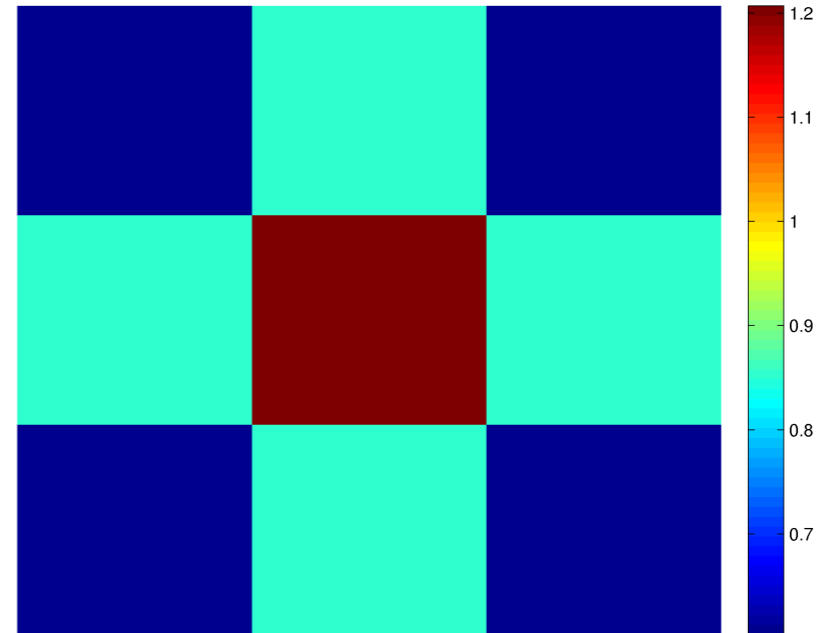
The data

SVD?

- Could we find the groups using SVD?



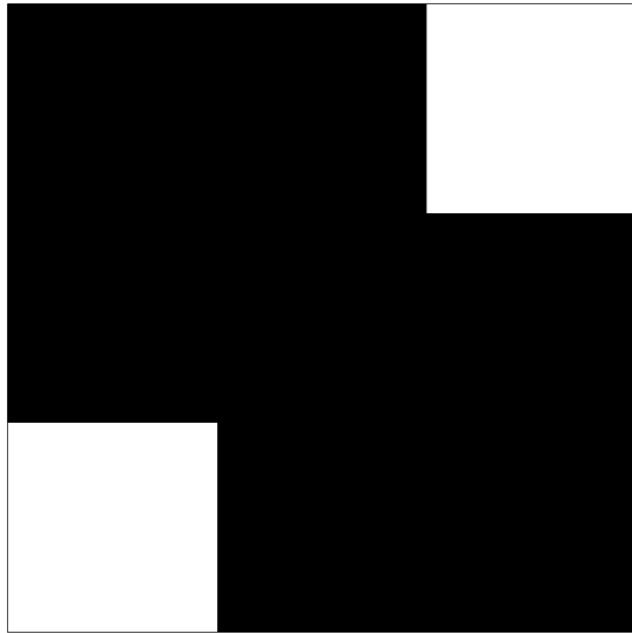
The data



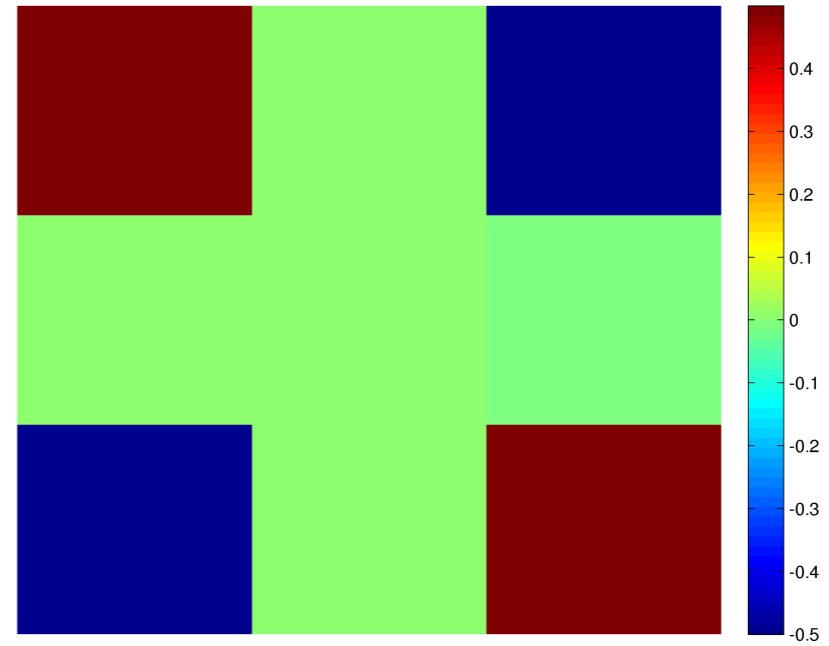
$$U_1 \Sigma_{1,1} V_1^T$$

SVD?

- Could we find the groups using SVD?



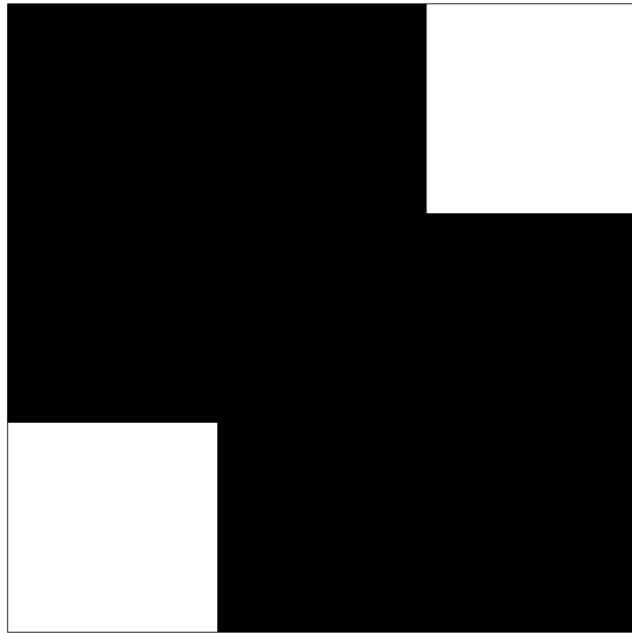
The data



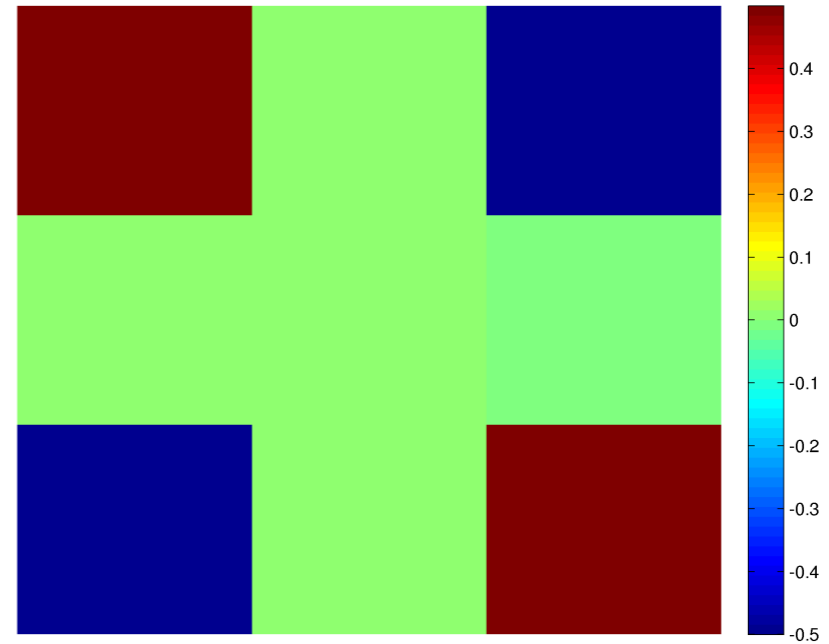
$$U_2 \Sigma_{2,2} V_2^T$$

SVD?

- Could we find the groups using SVD?



The data

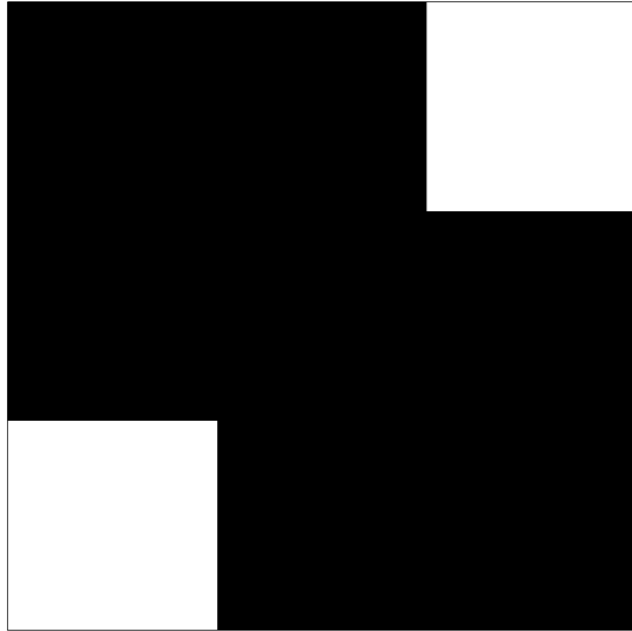


$$U_2 \Sigma_{2,2} V_2^T$$

- SVD cannot find the groups.

NMF?

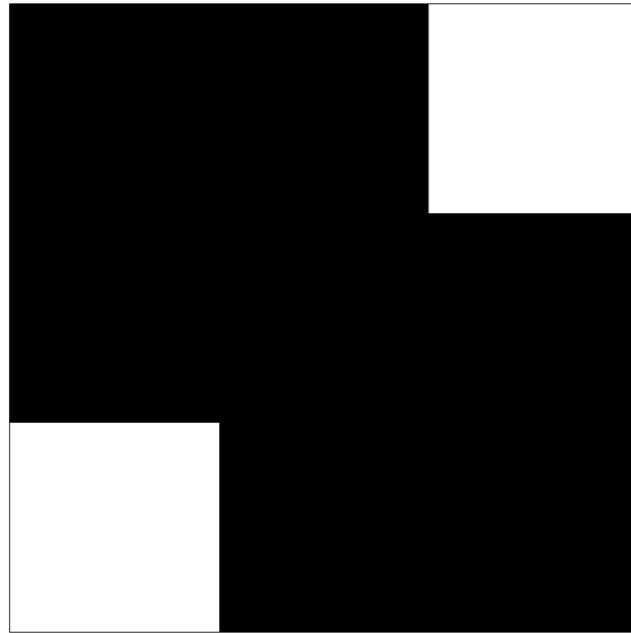
- The data is non-negative, so what about NMF?



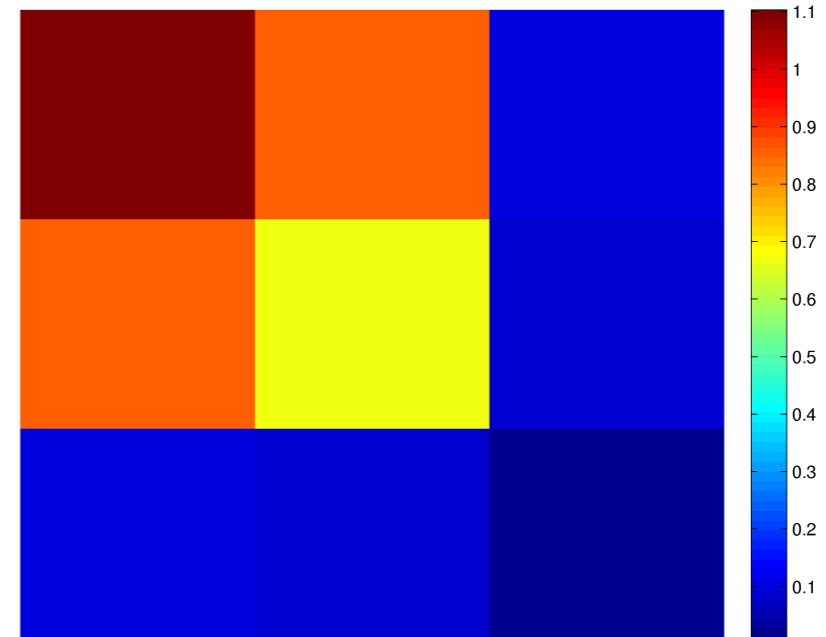
The data

NMF?

- The data is non-negative, so what about NMF?



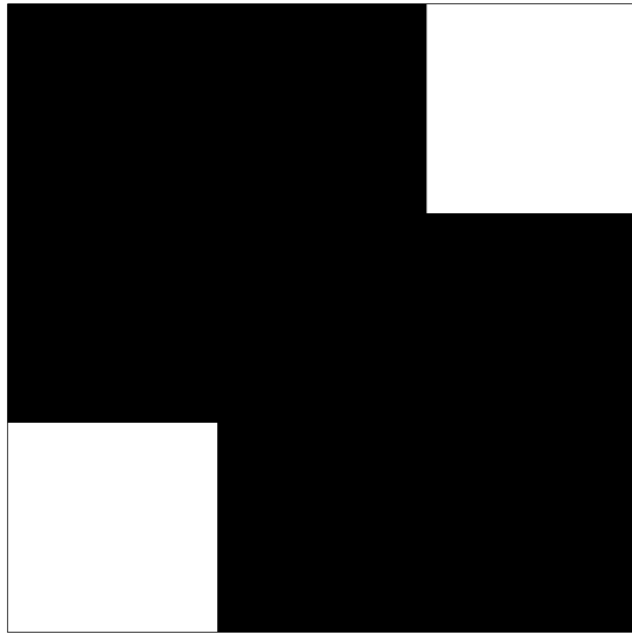
The data



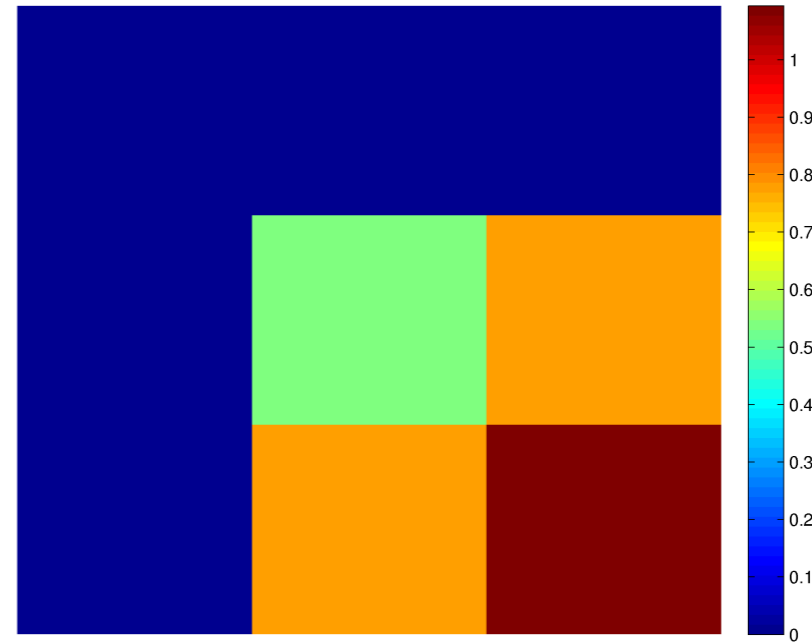
$W_1 H_1$

NMF?

- The data is non-negative, so what about NMF?



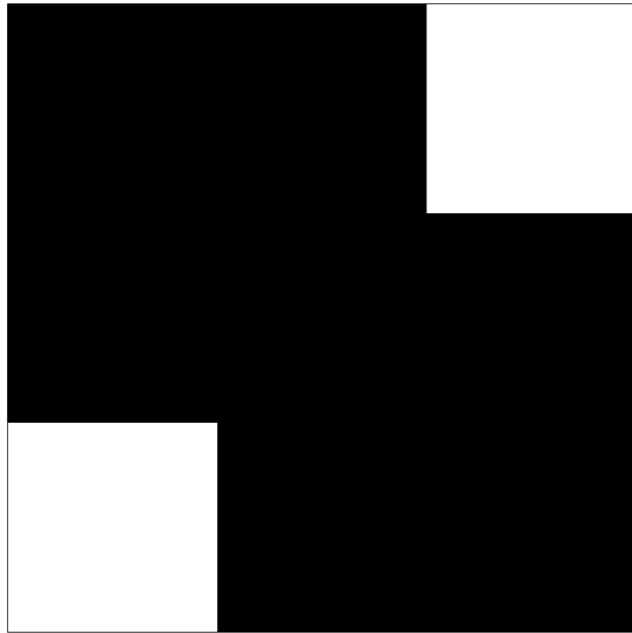
The data



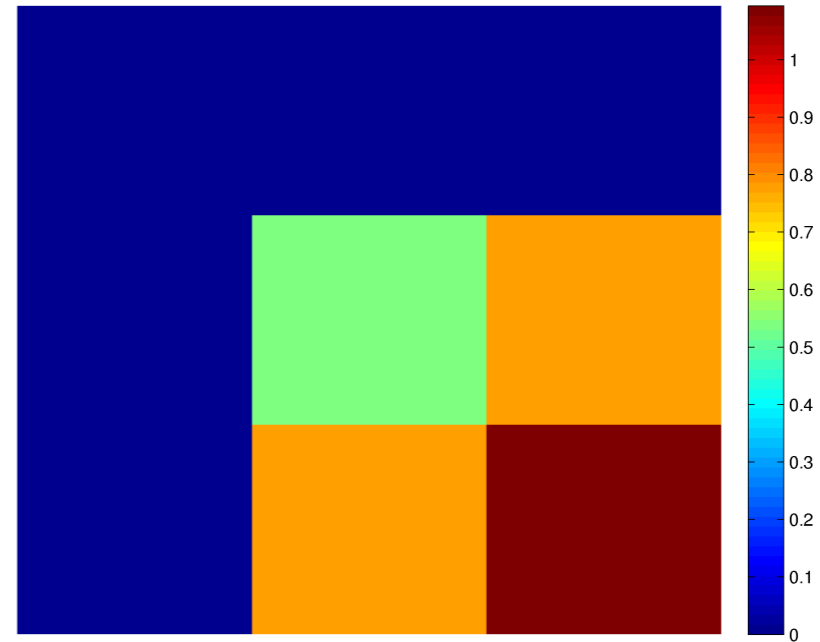
$W_2 H_2$

NMF?

- The data is non-negative, so what about NMF?



The data

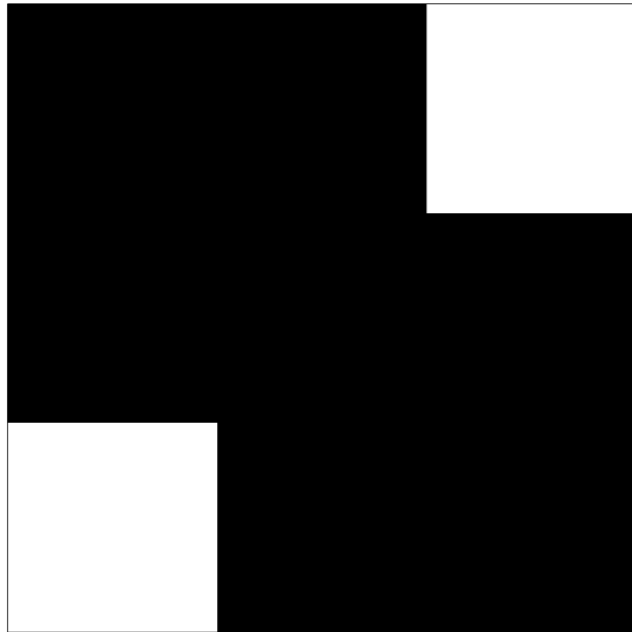


W_2H_2

- Already closer, but is the middle element in the group or out of the group?

Clustering?

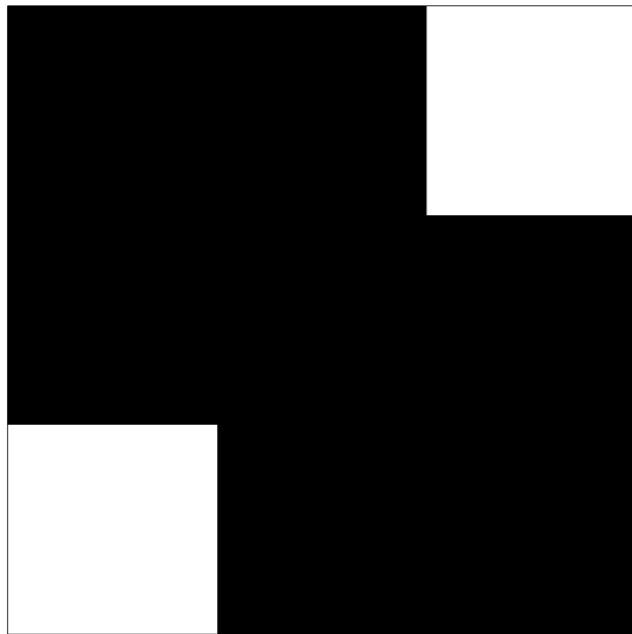
- So NMF's problem was that the results were not precise yes/no. Clustering can do that...



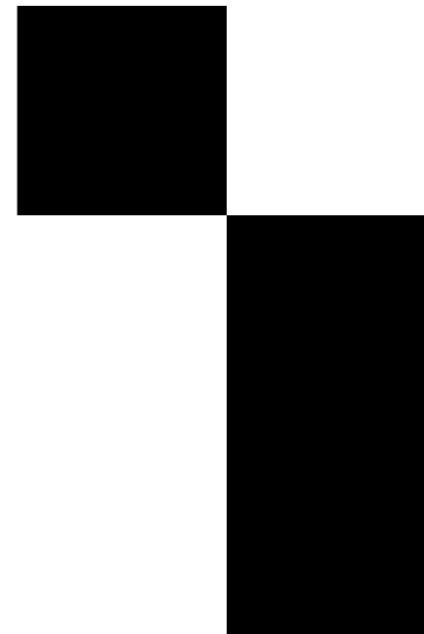
The data

Clustering?

- So NMF's problem was that the results were not precise yes/no. Clustering can do that...



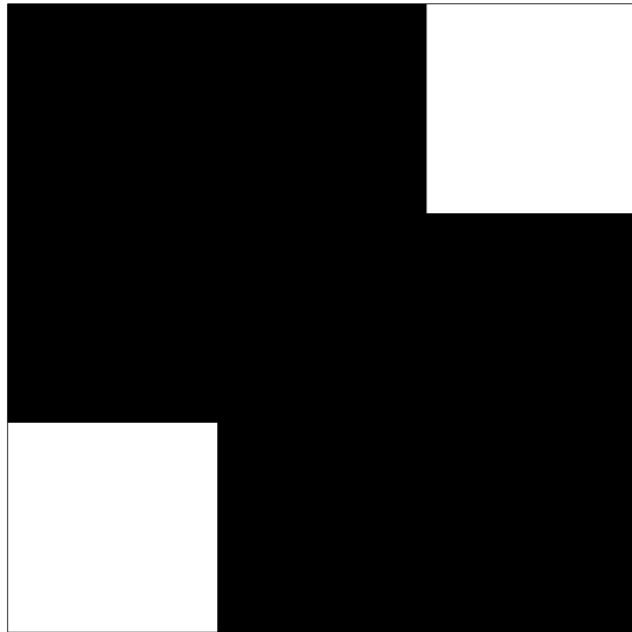
The data



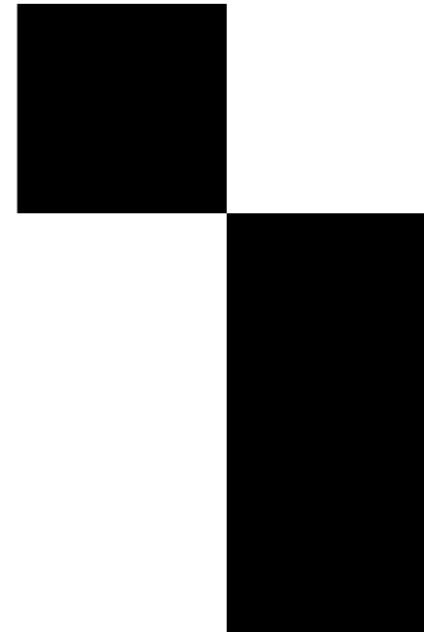
Cluster assignment matrix

Clustering?

- So NMF's problem was that the results were not precise yes/no. Clustering can do that...



The data

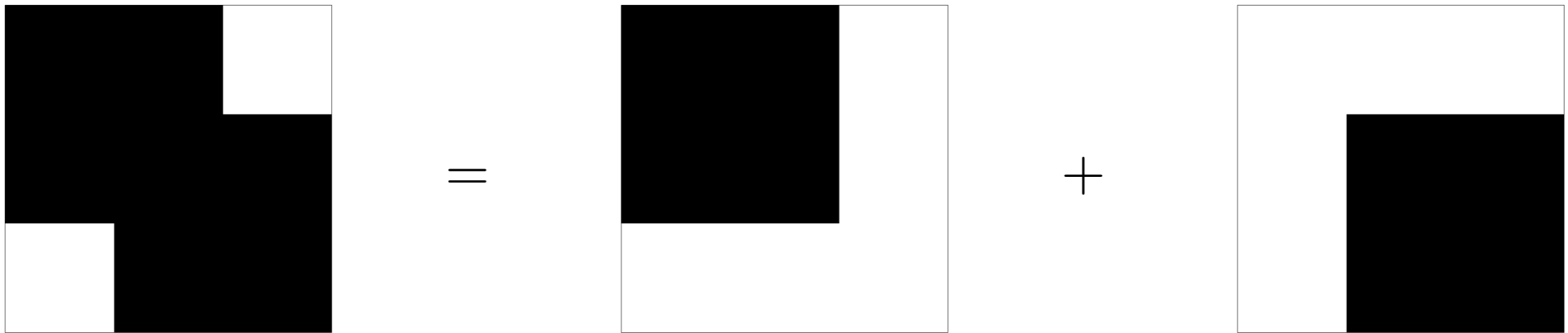


Cluster assignment matrix

- Precise, yes, but arbitrarily assigns  and “well-known” to one of the groups

Boolean matrix factorization

- What we want looks like this:



- The problem: the sum of these two components is *not* the data
 - ▶ The center element will have value 2
- Solution: don't care about multiplicity, but let $1 + 1 = 1$

Outline

- 1 Warm-Up
- 2 What is BMF**
- 3 BMF vs. other three-letter abbreviations
- 4 Binary matrices, tiles, graphs, and sets
- 5 Computational Complexity
- 6 Algorithms
- 7 Wrap-Up

Boolean matrix product

Boolean matrix product

The **Boolean product** of binary matrices $\mathbf{A} \in \{0, 1\}^{m \times k}$ and $\mathbf{B} \in \{0, 1\}^{k \times n}$, denoted $\mathbf{A} \boxtimes \mathbf{B}$, is such that

$$(\mathbf{A} \boxtimes \mathbf{B})_{ij} = \bigvee_{\ell=1}^k \mathbf{A}_{i\ell} \mathbf{B}_{\ell j} .$$

- The matrix product over the *Boolean semi-ring* $(\{0, 1\}, \wedge, \vee)$
 - ▶ Equivalently, normal matrix product with addition defined as $1 + 1 = 1$
 - ▶ Binary matrices equipped with such algebra are called **Boolean matrices**
- The Boolean product is only defined for binary matrices
- $\mathbf{A} \boxtimes \mathbf{B}$ is binary for all \mathbf{A} and \mathbf{B}

Definition of the BMF

Boolean Matrix Factorization (BMF)

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

- Typically (in data mining), k is given, and we try to find \mathbf{B} and \mathbf{C} to get as close to \mathbf{A} as possible
- Normally the optimization function is the squared Frobenius norm of the residual, $\|\mathbf{A} - (\mathbf{B} \boxtimes \mathbf{C})\|_F^2$
 - ▶ Equivalently, $|\mathbf{A} \oplus (\mathbf{B} \boxtimes \mathbf{C})|$ where
 - ★ $|\mathbf{A}|$ is the sum of values of \mathbf{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

The Boolean rank

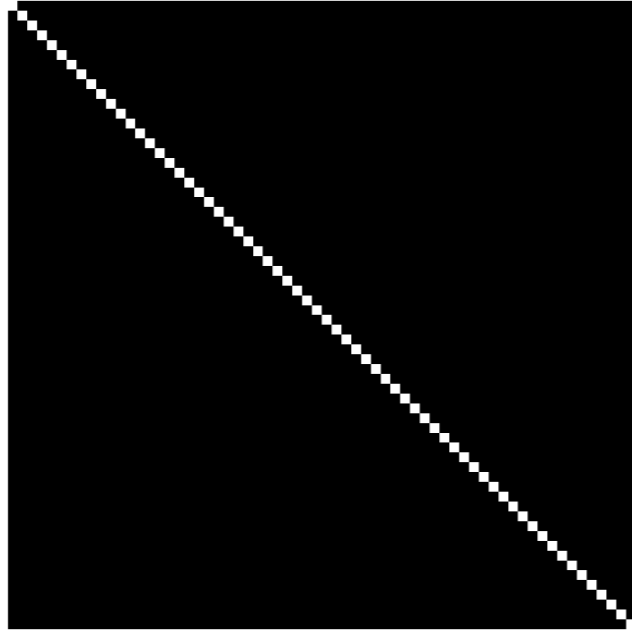
- The **Boolean rank** of a binary matrix $\mathbf{A} \in \{0, 1\}^{m \times n}$, $\text{rank}_B(\mathbf{A})$ is the smallest integer k such that there exists $\mathbf{B} \in \{0, 1\}^{m \times k}$ and $\mathbf{C} \in \{0, 1\}^{k \times n}$ for which $\mathbf{A} = \mathbf{B} \boxtimes \mathbf{C}$
 - ▶ Equivalently, the smallest k such that \mathbf{A} is the element-wise *or* of k rank-1 binary matrices
- Exactly like normal or nonnegative rank, but over Boolean algebra

The Boolean rank

- The **Boolean rank** of a binary matrix $\mathbf{A} \in \{0, 1\}^{m \times n}$, $\text{rank}_B(\mathbf{A})$ is the smallest integer k such that there exists $\mathbf{B} \in \{0, 1\}^{m \times k}$ and $\mathbf{C} \in \{0, 1\}^{k \times n}$ for which $\mathbf{A} = \mathbf{B} \boxtimes \mathbf{C}$
 - ▶ Equivalently, the smallest k such that \mathbf{A} is the element-wise *or* of k rank-1 binary matrices
- Exactly like normal or nonnegative rank, but over Boolean algebra
- There exists binary matrices for which $\text{rank}(\mathbf{A}) \approx \frac{1}{2} \text{rank}_B(\mathbf{A})$
- There exists binary matrices for which $\text{rank}_B(\mathbf{A}) = O(\log(\text{rank}(\mathbf{A})))$
- The logarithmic ratio is essentially the best possible

Another example

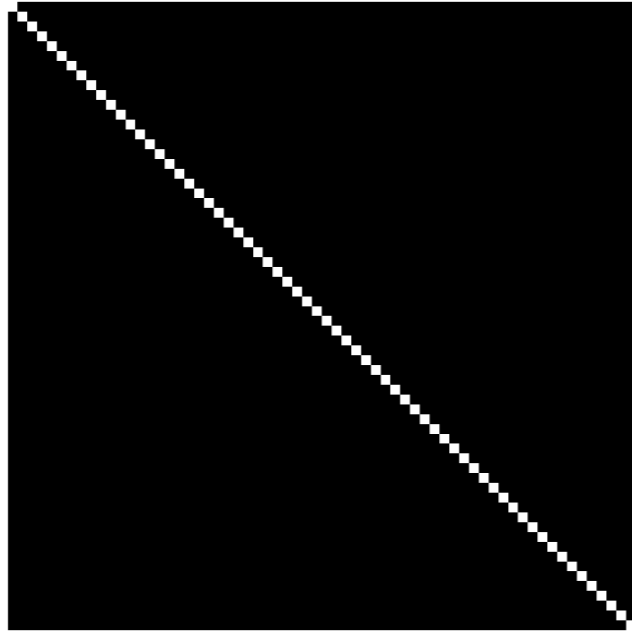
- Consider the complement of the identity matrix \bar{I}
 - ▶ It has full normal rank, but what about the Boolean rank?



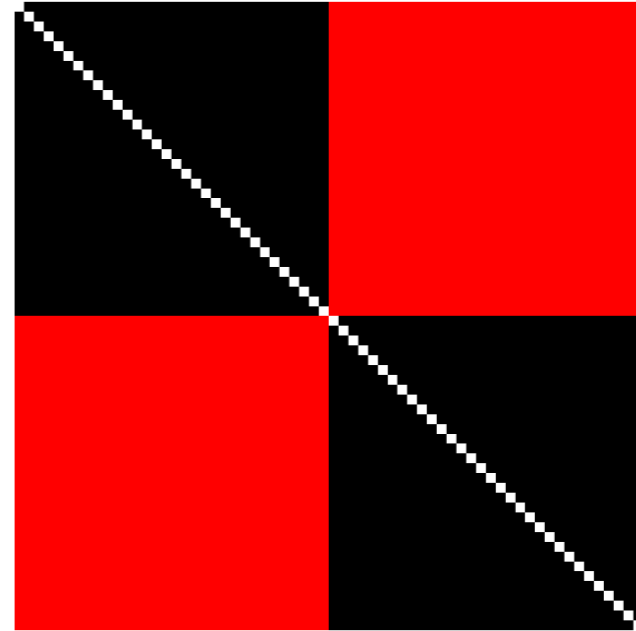
\bar{I}_{64}

Another example

- Consider the complement of the identity matrix \bar{I}
 - ▶ It has full normal rank, but what about the Boolean rank?



\bar{I}_{64}

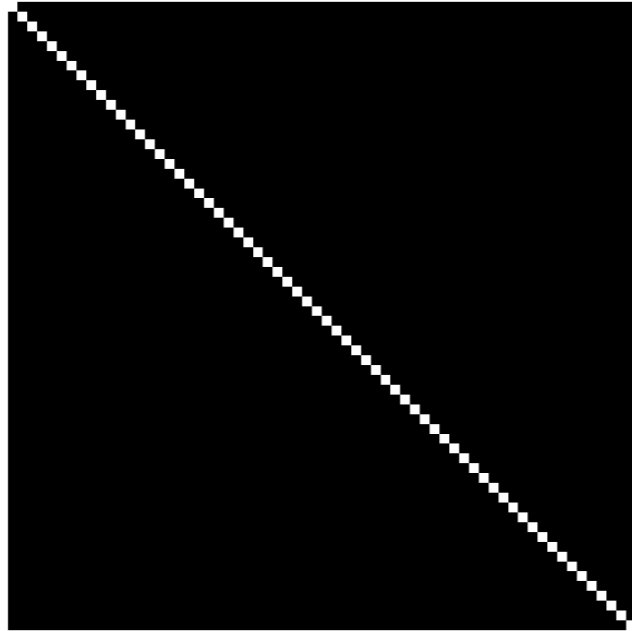


Boolean rank-2

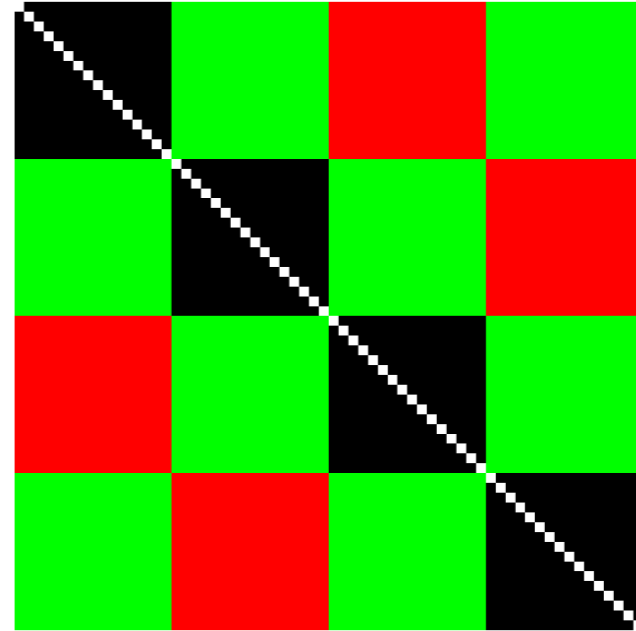
- The factorization is symmetric on diagonal so we draw two factors at a time

Another example

- Consider the complement of the identity matrix \bar{I}
 - ▶ It has full normal rank, but what about the Boolean rank?



\bar{I}_{64}

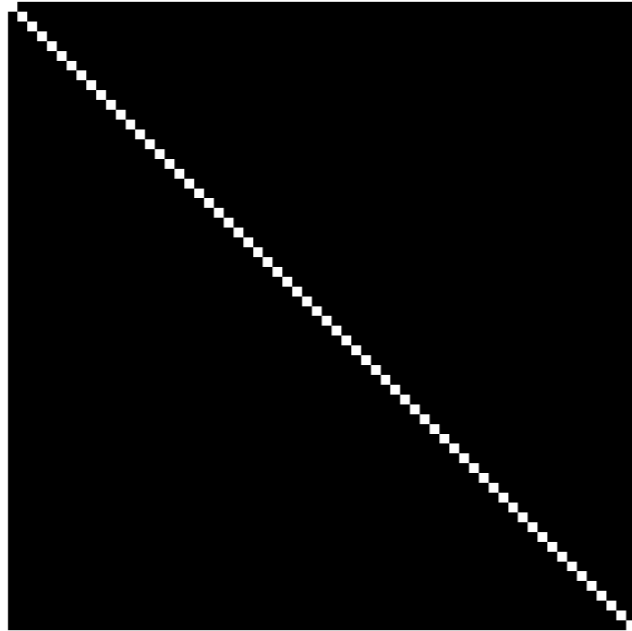


Boolean rank-4

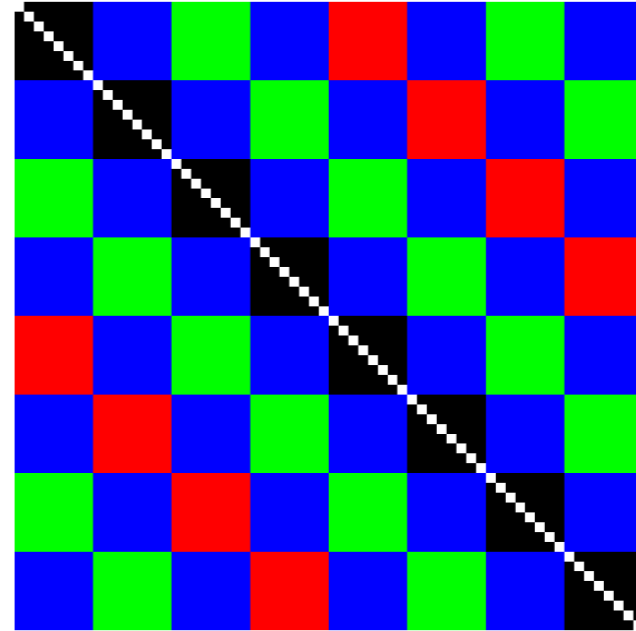
- The factorization is symmetric on diagonal so we draw two factors at a time

Another example

- Consider the complement of the identity matrix \bar{I}
 - ▶ It has full normal rank, but what about the Boolean rank?



\bar{I}_{64}

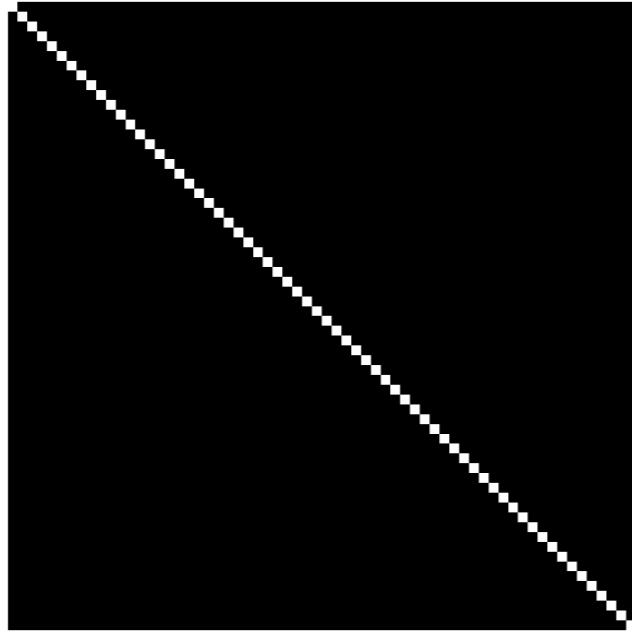


Boolean rank-6

- The factorization is symmetric on diagonal so we draw two factors at a time

Another example

- Consider the complement of the identity matrix \bar{I}
 - ▶ It has full normal rank, but what about the Boolean rank?



\bar{I}_{64}

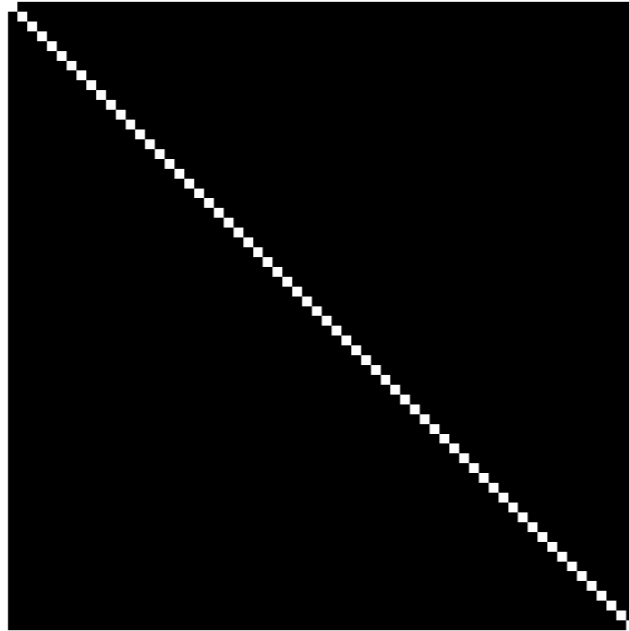


Boolean rank-8

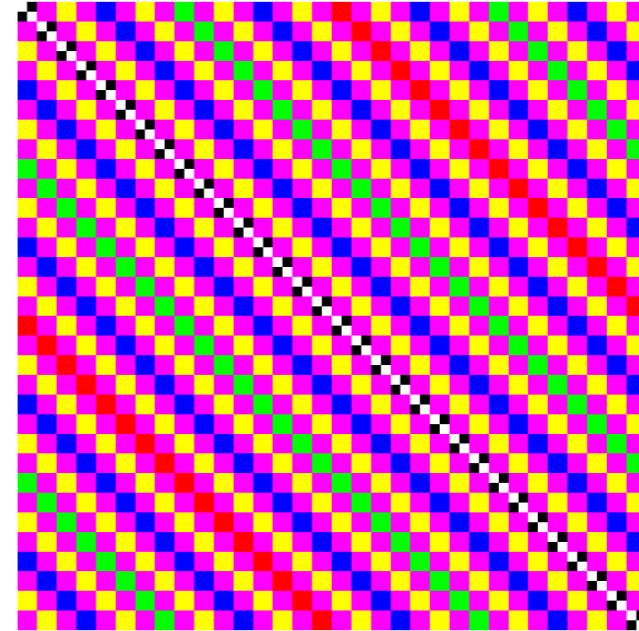
- The factorization is symmetric on diagonal so we draw two factors at a time

Another example

- Consider the complement of the identity matrix \bar{I}
 - ▶ It has full normal rank, but what about the Boolean rank?



\bar{I}_{64}

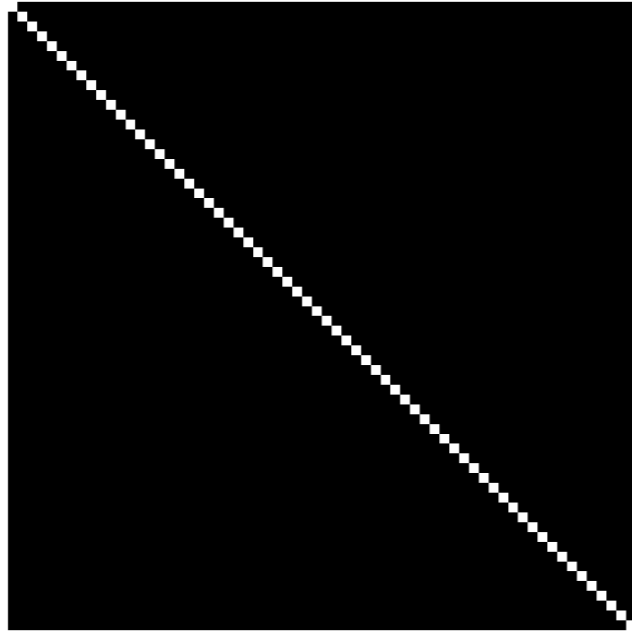


Boolean rank-10

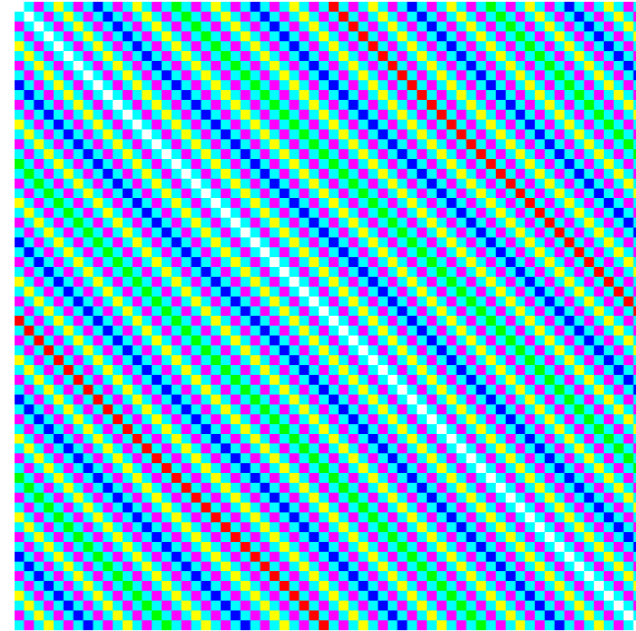
- The factorization is symmetric on diagonal so we draw two factors at a time

Another example

- Consider the complement of the identity matrix \bar{I}
 - ▶ It has full normal rank, but what about the Boolean rank?



\bar{I}_{64}

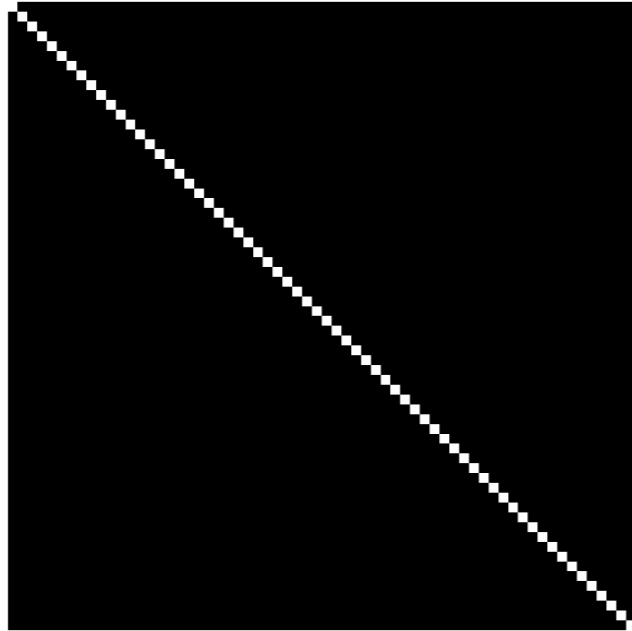


Boolean rank-12

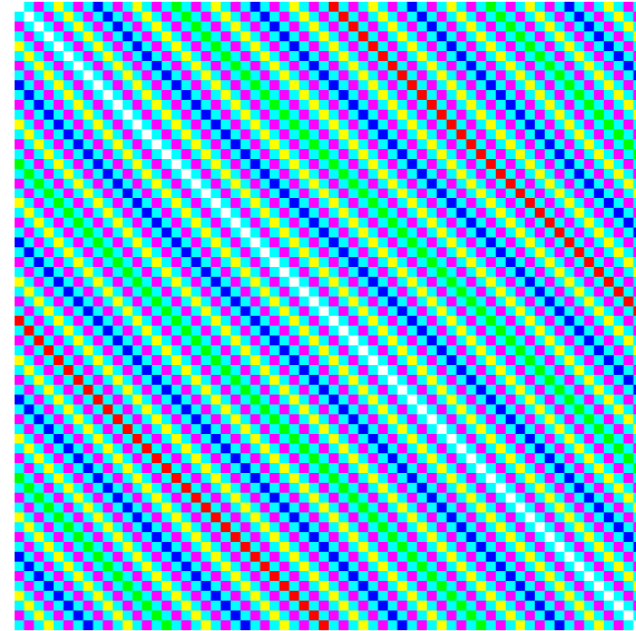
- The factorization is symmetric on diagonal so we draw two factors at a time

Another example

- Consider the complement of the identity matrix \bar{I}
 - ▶ It has full normal rank, but what about the Boolean rank?



\bar{I}_{64}

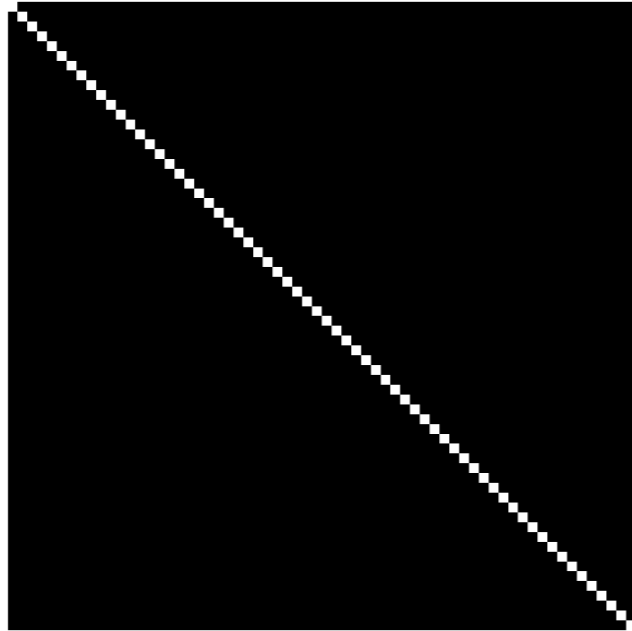


Boolean rank-12

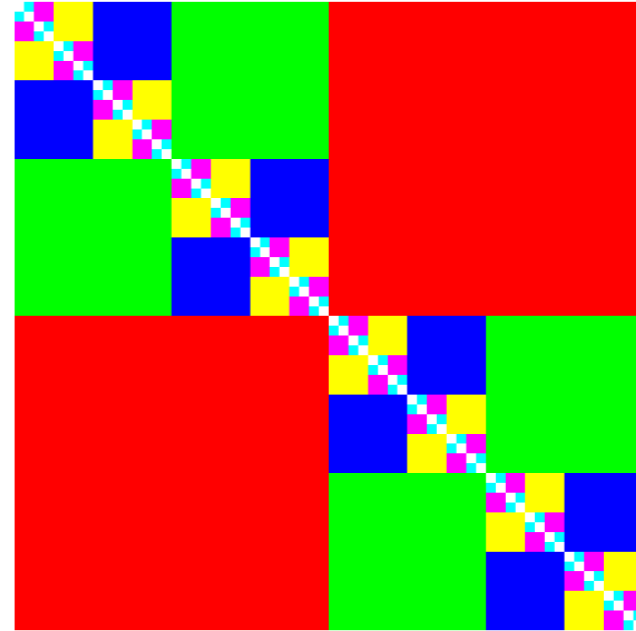
- The factorization is symmetric on diagonal so we draw two factors at a time
- The Boolean rank of the data is $12 = 2 \log_2(64)$

Another example

- Consider the complement of the identity matrix \bar{I}
 - ▶ It has full normal rank, but what about the Boolean rank?



\bar{I}_{64}

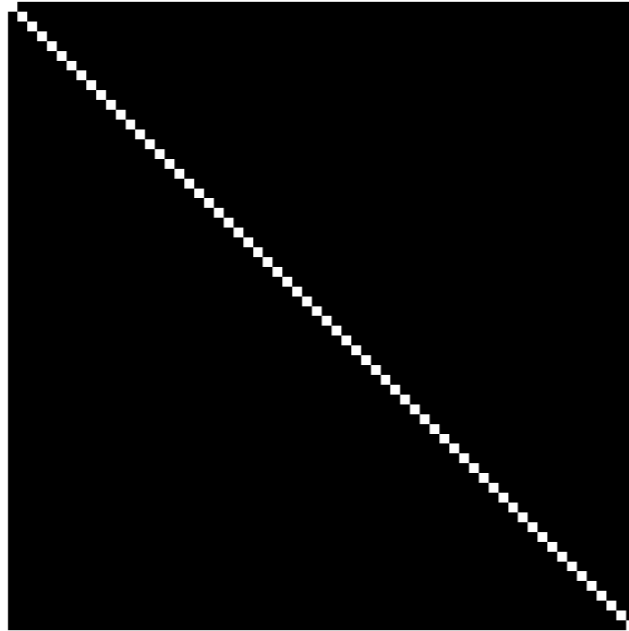


Boolean rank-12

- The factorization is symmetric on diagonal so we draw two factors at a time
- The Boolean rank of the data is $12 = 2 \log_2(64)$
- Let's draw the components in reverse order to see the structure

Another example

- Consider the complement of the identity matrix \bar{I}
 - ▶ It has full normal rank, but what about the Boolean rank?



\bar{I}_{64}



Factor matrices

- The factorization is symmetric on diagonal so we draw two factors at a time
- The Boolean rank of the data is $12 = 2 \log_2(64)$
- Let's draw the components in reverse order to see the structure
 - ▶ And the factor matrices have nice structure, too

Outline

- 1 Warm-Up
- 2 What is BMF
- 3 BMF vs. other three-letter abbreviations**
- 4 Binary matrices, tiles, graphs, and sets
- 5 Computational Complexity
- 6 Algorithms
- 7 Wrap-Up

BMF vs. SVD

- Truncated SVD gives Frobenius-optimal rank- k approximations of the matrix
- But we've already seen that matrices can have smaller Boolean than real rank \Rightarrow BMF can give exact decompositions where SVD cannot
 - ▶ Contradiction?

BMF vs. SVD

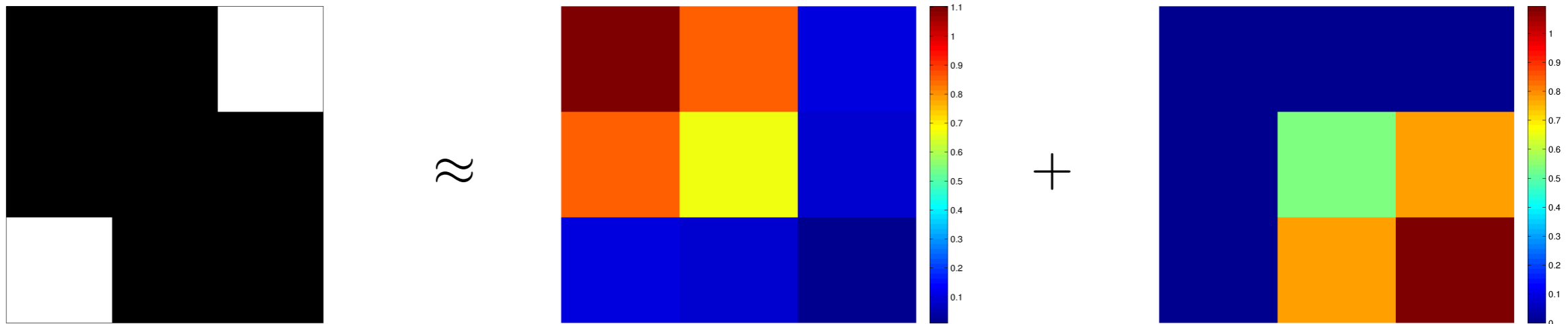
- Truncated SVD gives Frobenius-optimal rank- k approximations of the matrix
- But we've already seen that matrices can have smaller Boolean than real rank \Rightarrow BMF can give exact decompositions where SVD cannot
 - ▶ Contradiction?
- The answer lies in different algebras: SVD is optimal if you're using the normal algebra
 - ▶ BMF can utilize its different addition in some cases very effectively

BMF vs. SVD

- Truncated SVD gives Frobenius-optimal rank- k approximations of the matrix
- But we've already seen that matrices can have smaller Boolean than real rank \Rightarrow BMF can give exact decompositions where SVD cannot
 - ▶ Contradiction?
- The answer lies in different algebras: SVD is optimal if you're using the normal algebra
 - ▶ BMF can utilize its different addition in some cases very effectively
- In practice, however, SVD usually gives the smallest reconstruction error
 - ▶ Even when it's not exactly correct, it's very close
- But reconstruction error isn't all that matters
 - ▶ BMF can be more interpretable and more sparse
 - ▶ BMF finds different structure than SVD

BMF vs. NMF

- Both BMF and NMF work on anti-negative semi-rings
 - ▶ There is no inverse to addition
 - ▶ “Parts-of-whole”
- BMF and NMF can be very close to each other
 - ▶ Especially after NMF is rounded to binary factor matrices
- But NMF has to scale down overlapping components



BMF vs. clustering

- BMF is a relaxed version of clustering in the hypercube $\{0, 1\}^n$
 - ▶ The left factor matrix \mathbf{B} is sort-of cluster assignment matrix, but the “clusters” don’t have to partition the rows
 - ▶ The right factor matrix \mathbf{C} gives the centroids in $\{0, 1\}^n$
- If we restrict \mathbf{B} to a cluster assignment matrix (each row has exactly one 1) we get a clustering problem
 - ▶ Computationally much easier than BMF
 - ▶ Simple local search works well
- But clustering also loses the power of overlapping components

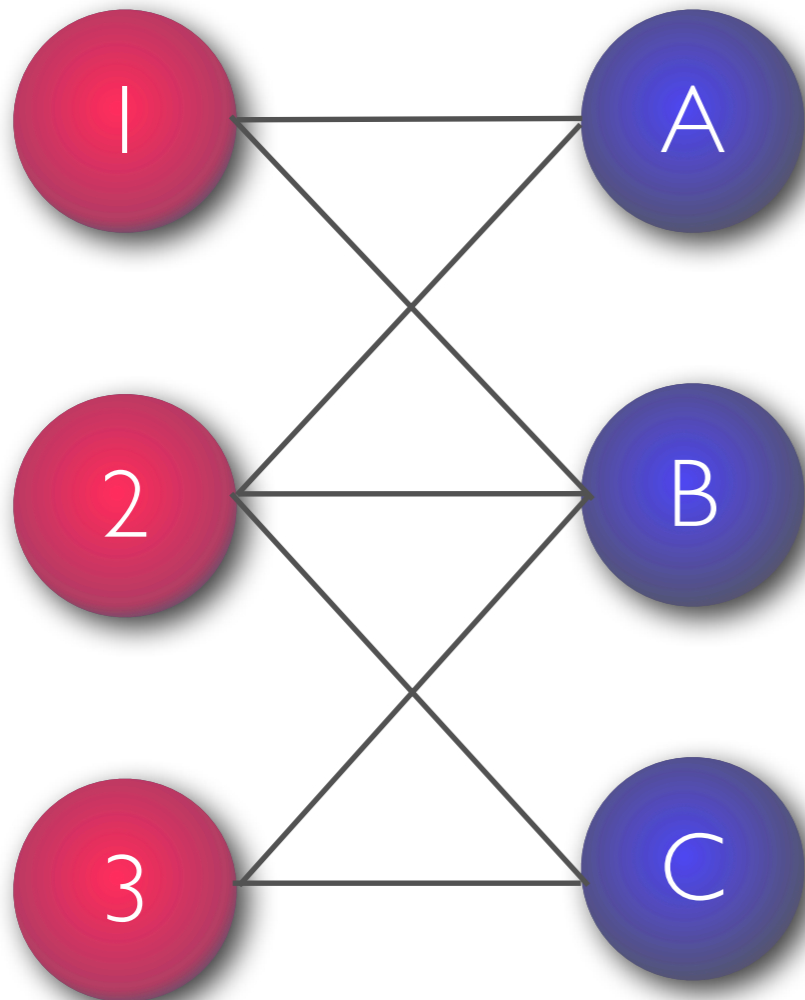
Outline

- 1 Warm-Up
- 2 What is BMF
- 3 BMF vs. other three-letter abbreviations
- 4 Binary matrices, tiles, graphs, and sets**
- 5 Computational Complexity
- 6 Algorithms
- 7 Wrap-Up

Binary matrices and bipartite graphs

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

||

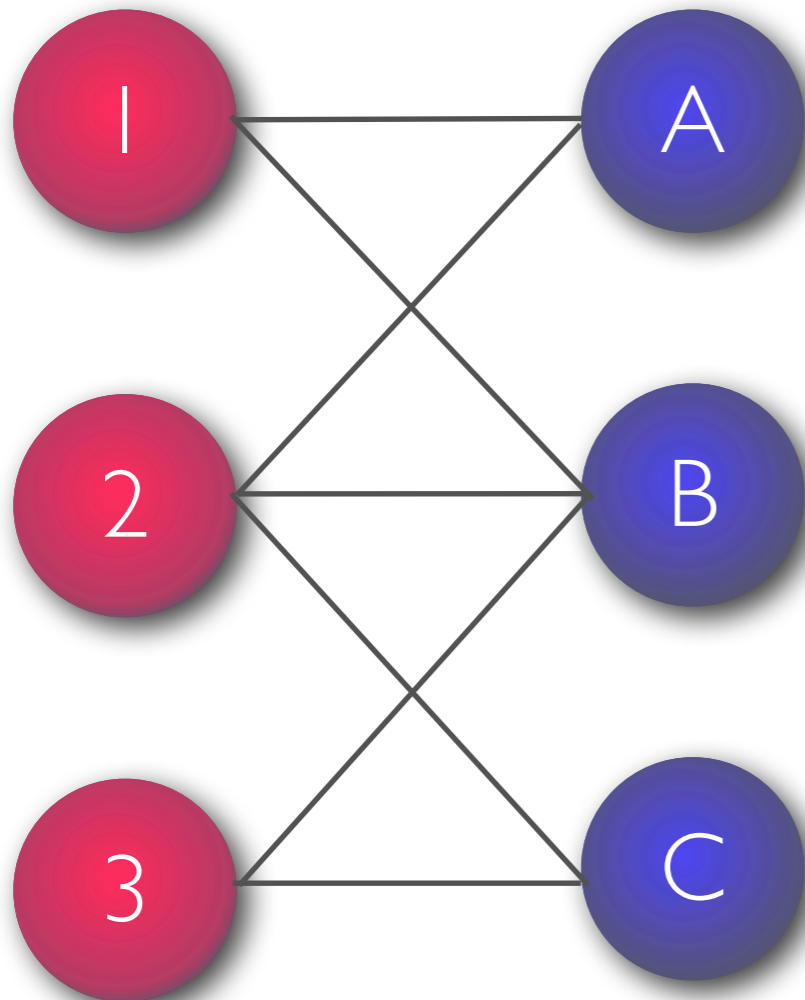


- There is a bijection between $\{0, 1\}^{m \times n}$ and (unweighted, undirected) bipartite graphs of $m + n$ vertices
 - ▶ Every $\mathbf{A} \in \{0, 1\}^{m \times n}$ is a **bi-adjacency matrix** of some bipartite graph $G = (V \cup U, E)$
 - ▶ V has m vertices, U has n vertices and $(v_i, u_j) \in E$ iff $\mathbf{A}_{ij} = 1$

BMF and (quasi-)biclique covers

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

||

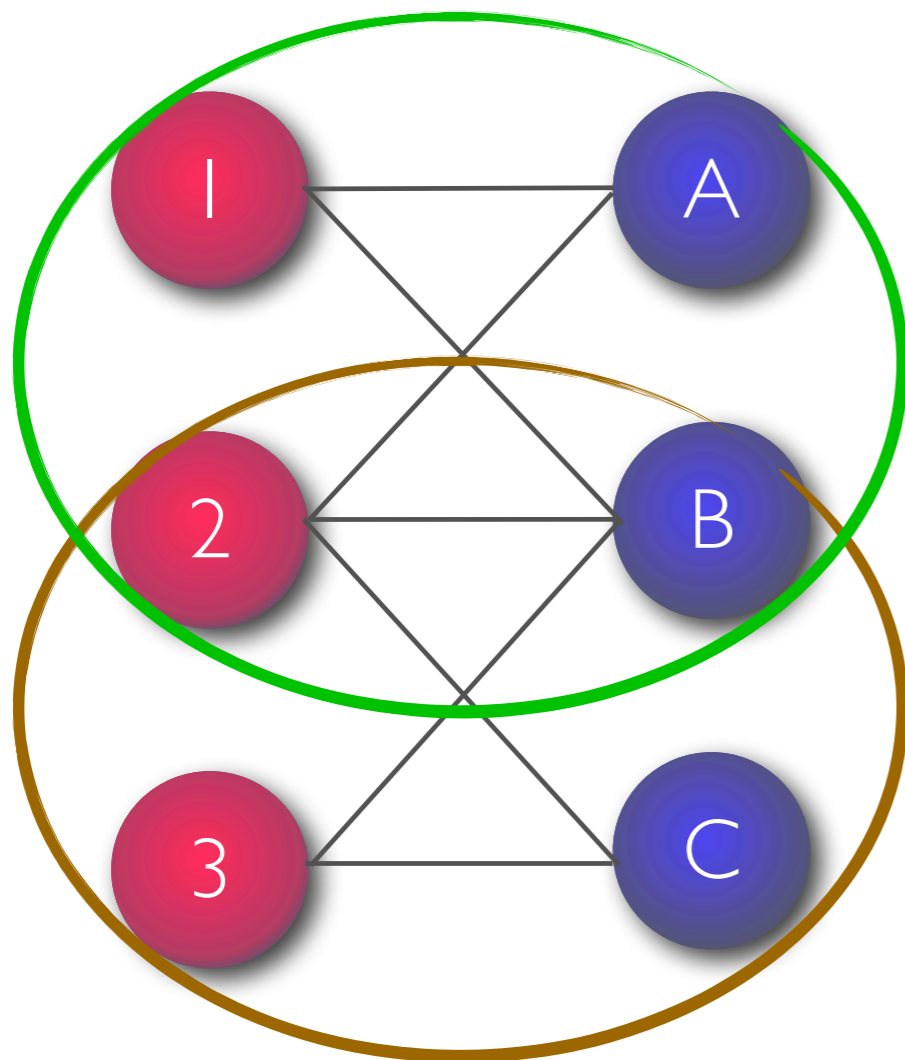


- A **biclique** is a complete bipartite graph
 - ▶ Each left-hand-side vertex is connected to each right-hand-side vertex
- Each rank-1 binary matrix defines a biclique (subgraph)
 - ▶ If $\mathbf{v} \in \{0, 1\}^m$ and $\mathbf{u} \in \{0, 1\}^n$, then $\mathbf{v}\mathbf{u}^T$ is a biclique between $v_i \in V$ and $u_j \in U$ for which $\mathbf{v}_i = \mathbf{u}_j = 1$

BMF and (quasi-)biclique covers

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

||

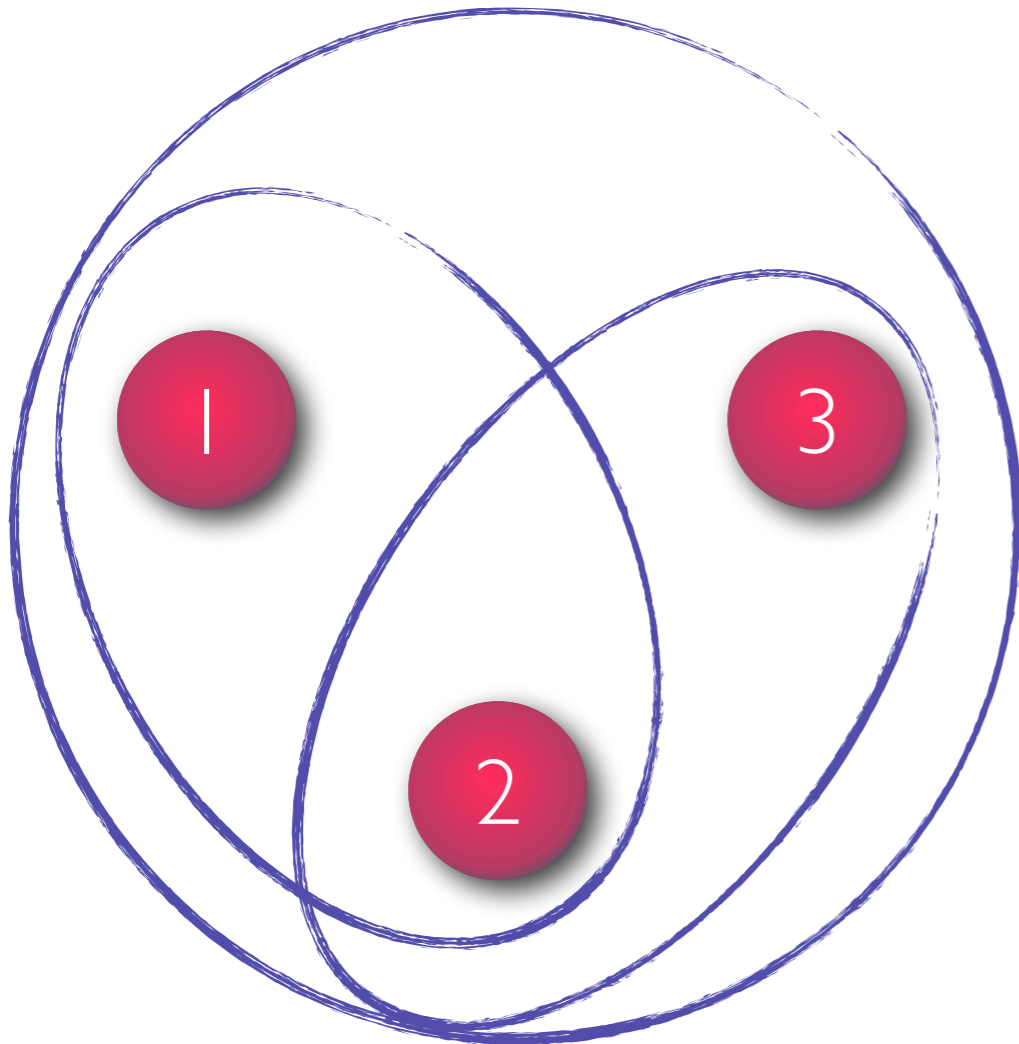


- A **biclique** is a complete bipartite graph
 - ▶ Each left-hand-side vertex is connected to each right-hand-side vertex
- Each rank-1 binary matrix defines a biclique (subgraph)
 - ▶ If $\mathbf{v} \in \{0, 1\}^m$ and $\mathbf{u} \in \{0, 1\}^n$, then $\mathbf{v}\mathbf{u}^T$ is a biclique between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

Binary matrices and sets

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

||

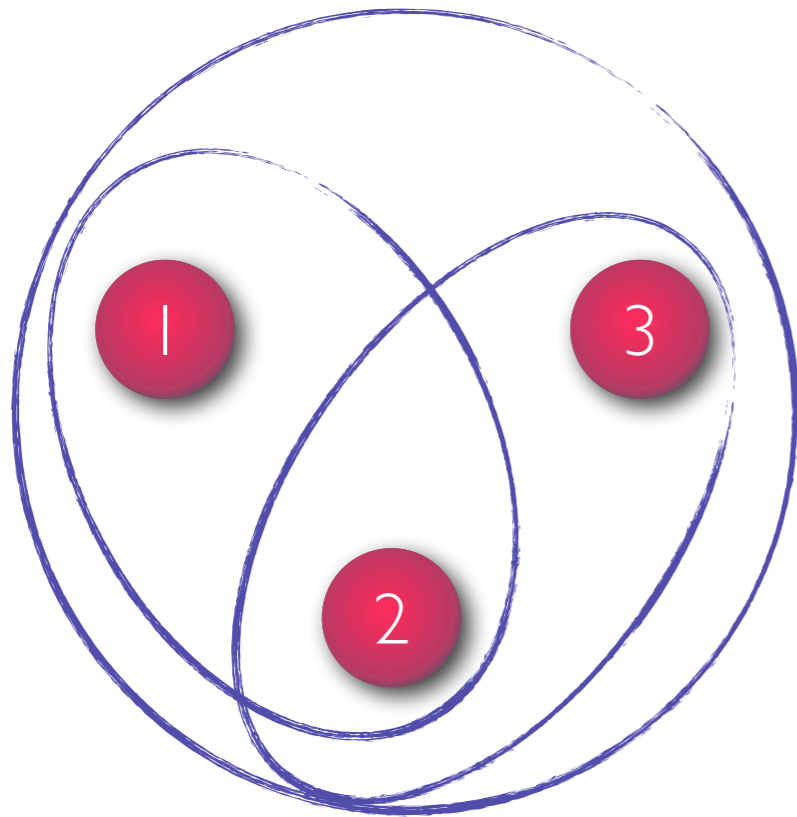


- There is a bijection between $\{0, 1\}^{m \times n}$ and sets systems of m sets over n -element universes, $(U, \mathcal{S} \in 2^U)$, $|\mathcal{S}| = m$, $|U| = n$
 - ▶ Up to labeling of elements in U
 - ▶ The columns of $\mathbf{A} \in \{0, 1\}^{m \times n}$ correspond to the elements of U
 - ▶ The rows of \mathbf{A} correspond to the sets in \mathcal{S}
 - ▶ If $S_i \in \mathcal{S}$, then $u_j \in S_i$ iff $\mathbf{A}_{ij} = 1$

BMF and the Set Basis problem

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

||

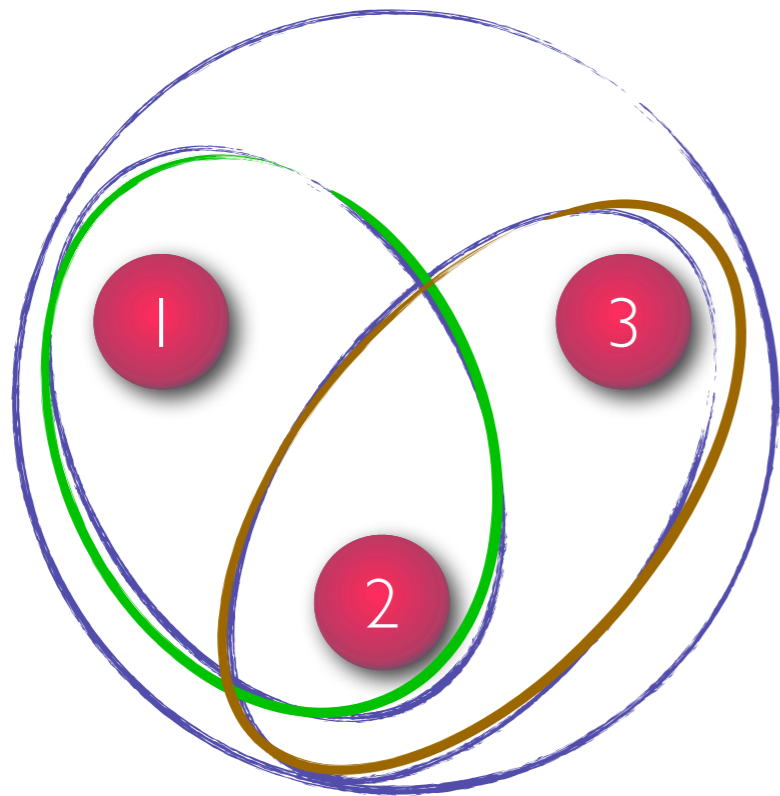


- In the **Set Basis** problem, we are given a set system (U, \mathcal{S}) , and our task is to find collection $\mathcal{C} \subseteq 2^U$ such that we can cover each set $S \in \mathcal{S}$ with a union of some sets of \mathcal{C}
 - ▶ For each $S \in \mathcal{S}$, there is $\mathcal{C}_S \subseteq \mathcal{C}$ such that $S = \bigcup_{C \in \mathcal{C}_S} C$

BMF and the Set Basis problem

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

||

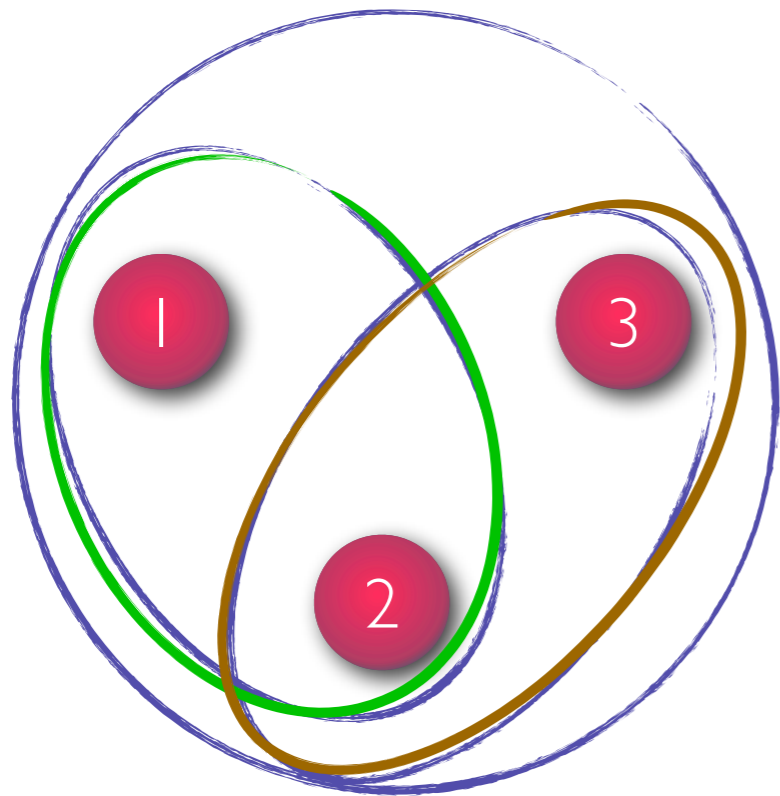


- In the **Set Basis** problem, we are given a set system (U, \mathcal{S}) , and our task is to find collection $\mathcal{C} \subseteq 2^U$ such that we can cover each set $S \in \mathcal{S}$ with a union of some sets of \mathcal{C}
 - ▶ For each $S \in \mathcal{S}$, there is $\mathcal{C}_S \subseteq \mathcal{C}$ 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

BMF and the Set Basis problem

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

||



- 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

Binary matrices in data mining

- A common use for binary matrices is to represent presence/absence data
 - ▶ Animals in spatial areas
 - ▶ Items in transactions
- Another common use are binary relations
 - ▶ “has seen” between users and movies
 - ▶ “links to” between anchor texts and web pages
- Also any directed graphs are typical
- A common problem is that presence/absence data doesn't necessarily tell about absence
 - ▶ We know that 1s are probably “true” 1s, but 0s might either be “true” 0s or missing values
 - ★ If a species is not in some area, is it because we haven't seen it or because it's not there?

Outline

- 1 Warm-Up
- 2 What is BMF
- 3 BMF vs. other three-letter abbreviations
- 4 Binary matrices, tiles, graphs, and sets
- 5 Computational Complexity**
- 6 Algorithms
- 7 Wrap-Up

The Basis Usage problem

- Alternating projections -style algorithms are very common tool for finding matrix factorizations
 - ▶ E.g. the alternating least squares algorithm
- As a subproblem they require you to solve the following problem: Given matrices \mathbf{Y} and \mathbf{A} , find matrix \mathbf{X} such that $\|\mathbf{Y} - \mathbf{AX}\|$ is minimized
 - ▶ Each column of \mathbf{X} is independent: Given vector \mathbf{y} and matrix \mathbf{A} , find a vector \mathbf{x} that minimizes $\|\mathbf{y} - \mathbf{Ax}\|$
 - ★ Linear regression if no constraints on \mathbf{x} and Euclidean norm is used
- The **Basis Usage** problem is the Boolean variant of this problem:

Basis Usage problem

Given binary matrices \mathbf{A} and \mathbf{B} , find binary matrix \mathbf{C} that minimizes $\|\mathbf{A} - (\mathbf{B} \boxtimes \mathbf{C})\|_F^2$.

The Basis Usage problem

- Alternating projections -style algorithms are very common tool for finding matrix factorizations
 - ▶ E.g. the alternating least squares algorithm
- As a subproblem they require you to solve the following problem: Given matrices \mathbf{Y} and \mathbf{A} , find matrix \mathbf{X} such that $\|\mathbf{Y} - \mathbf{AX}\|$ is minimized
 - ▶ Each column of \mathbf{X} is independent: Given vector \mathbf{y} and matrix \mathbf{A} , find a vector \mathbf{x} that minimizes $\|\mathbf{y} - \mathbf{Ax}\|$
 - ★ Linear regression if no constraints on \mathbf{x} and Euclidean norm is used
- The **Basis Usage** problem is the Boolean variant of this problem:

Basis Usage problem

Given binary matrices \mathbf{A} and \mathbf{B} , find binary matrix \mathbf{C} that minimizes $\|\mathbf{A} - (\mathbf{B} \boxtimes \mathbf{C})\|_F^2$.

- How hard can it be?

The Positive-Negative Partial Set Cover problem

Positive-Negative Partial Set Cover problem (\pm PSC)

Given a set system $(P \cup N, \mathcal{S} \in 2^{P \cup N})$ and integer k , find a partial cover $\mathcal{C} \subset \mathcal{S}$ of size k such that \mathcal{C} minimizes $|P \setminus (\cup \mathcal{C})| + |N \cap (\cup \mathcal{C})|$.

- \pm PSC minimizes the number of uncovered positive elements plus the number of covered elements

Back to the Basis Usage

- But what has the Basis Usage problem to do with \pm PSC?

Back to the Basis Usage

- But what has the Basis Usage problem to do with \pm PSC?
 - ▶ They're also almost equivalent problems

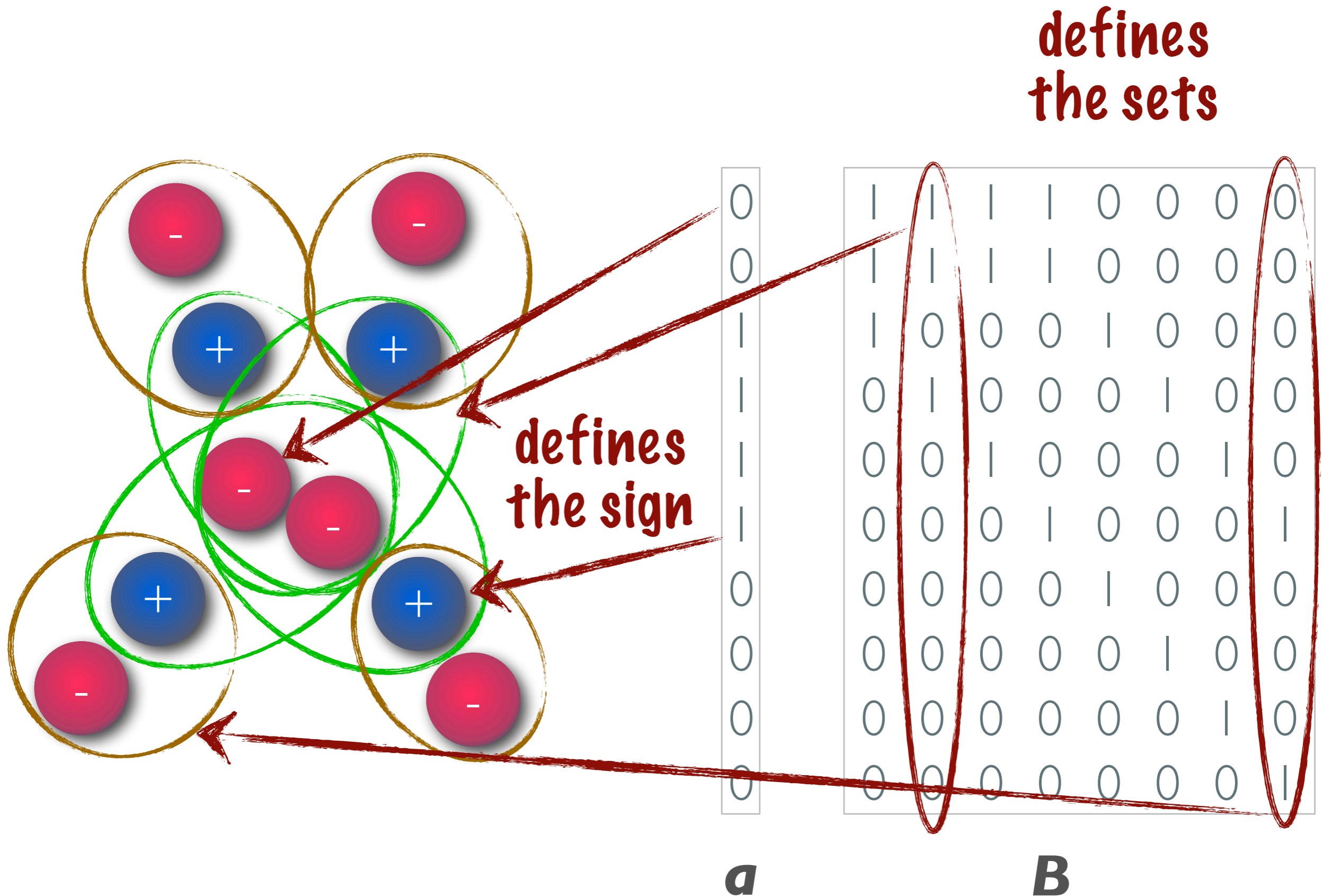
Back to the Basis Usage

- But what has the Basis Usage problem to do with \pm PSC?
 - ▶ They're also almost equivalent problems
- To see the equivalence, consider the one-column problem: given \mathbf{a} and \mathbf{B} , find \mathbf{c} such that $\|\mathbf{a} - \mathbf{B} \boxtimes \mathbf{c}\|_F^2$ is minimized
 - ▶ $\mathbf{a}_i \in P$ if $\mathbf{a}_i = 1$, o/w $\mathbf{a}_i \in N$
 - ▶ Sets in \mathcal{S} are defined by the columns of \mathbf{B} : $\mathbf{a}_i \in S_j$ if $\mathbf{B}_{ij} = 1$
 - ▶ If set S_j is selected to \mathcal{C} , then $\mathbf{c}_j = 1$ (o/w $\mathbf{c}_j = 0$)
 - ▶ And $|P \setminus (\cup \mathcal{C})| + |N \cap (\cup \mathcal{C})| = |\mathbf{A} \oplus (\mathbf{B} \boxtimes \mathbf{c})| = \|\mathbf{A} - \mathbf{B} \boxtimes \mathbf{c}\|_F^2$

Back to the Basis Usage

- But what has the Basis Usage problem to do with \pm PSC?
 - ▶ They're also almost equivalent problems
- To see the equivalence, consider the one-column problem: given \mathbf{a} and \mathbf{B} , find \mathbf{c} such that $\|\mathbf{a} - \mathbf{B} \boxtimes \mathbf{c}\|_F^2$ is minimized
 - ▶ $\mathbf{a}_i \in P$ if $\mathbf{a}_i = 1$, o/w $\mathbf{a}_i \in N$
 - ▶ Sets in \mathcal{S} are defined by the columns of \mathbf{B} : $\mathbf{a}_i \in S_j$ if $\mathbf{B}_{ij} = 1$
 - ▶ If set S_j is selected to \mathcal{C} , then $\mathbf{c}_j = 1$ (o/w $\mathbf{c}_j = 0$)
 - ▶ And $|P \setminus (\cup \mathcal{C})| + |N \cap (\cup \mathcal{C})| = |\mathbf{A} \oplus (\mathbf{B} \boxtimes \mathbf{c})| = \|\mathbf{A} - \mathbf{B} \boxtimes \mathbf{c}\|_F^2$
- So while Basis Usage and \pm PSC look different, they actually are essentially the same problem
 - ▶ Unfortunately this is also a hard problem, making algorithm development complicated

Example of \pm PSC and Basis Usage



Computational complexity

- Computing the Boolean rank is as hard as solving the Set Basis problem, i.e. NP-hard
 - ▶ Approximating the Boolean rank is as hard as approximating the minimum chromatic number of a graph, i.e. very hard
 - ▶ Compare to normal rank, which is easy save for precision issues
- Finding the least-error approximate BMF is NP-hard
 - ▶ And we cannot get any multiplicative approximation factors, as recognizing the case with zero error is also NP-hard
 - ▶ The problem is also hard to approximate within additive error
- Solving the \pm PSC problem is NP-hard and it is NP-hard to approximate within a superpolylogarithmic factor
 - ▶ Therefore, the Basis Usage and Component Selection problems are also NP-hard even to approximate

Outline

- 1 Warm-Up
- 2 What is BMF
- 3 BMF vs. other three-letter abbreviations
- 4 Binary matrices, tiles, graphs, and sets
- 5 Computational Complexity
- 6 Algorithms**
- 7 Wrap-Up

Two simple ideas

- **Idea 1:** Alternating updates
 - ▶ Start with random \mathbf{B} , find new \mathbf{C} , update \mathbf{B} , etc. until convergence
 - ▶ Guaranteed to converge in nm steps for $m \times n$ matrices

- **Idea 2:** Find many dense submatrices (quasi-bicliques) and select from them
 - ▶ Existing algorithms find the dense submatrices

Two simple ideas

- **Idea 1:** Alternating updates
 - ▶ Start with random \mathbf{B} , find new \mathbf{C} , update \mathbf{B} , etc. until convergence
 - ▶ Guaranteed to converge in nm steps for $m \times n$ matrices
 - ▶ Problem: requires solving the BU problem
 - ★ But it can be approximated

- **Idea 2:** Find many dense submatrices (quasi-bicliques) and select from them
 - ▶ Existing algorithms find the dense submatrices

Two simple ideas

- **Idea 1:** Alternating updates
 - ▶ Start with random \mathbf{B} , find new \mathbf{C} , update \mathbf{B} , etc. until convergence
 - ▶ Guaranteed to converge in nm steps for $m \times n$ matrices
 - ▶ Problem: requires solving the BU problem
 - ★ But it can be approximated
 - ▶ Problem: Converges too fast
 - ★ The optimization landscape is bumpy (many local optima)
- **Idea 2:** Find many dense submatrices (quasi-bicliques) and select from them
 - ▶ Existing algorithms find the dense submatrices

Two simple ideas

- **Idea 1:** Alternating updates
 - ▶ Start with random \mathbf{B} , find new \mathbf{C} , update \mathbf{B} , etc. until convergence
 - ▶ Guaranteed to converge in nm steps for $m \times n$ matrices
 - ▶ Problem: requires solving the BU problem
 - ★ But it can be approximated
 - ▶ Problem: Converges too fast
 - ★ The optimization landscape is bumpy (many local optima)
- **Idea 2:** Find many dense submatrices (quasi-bicliques) and select from them
 - ▶ Existing algorithms find the dense submatrices
 - ▶ Finding the dense submatrices is slow

Two simple ideas

- **Idea 1:** Alternating updates
 - ▶ Start with random \mathbf{B} , find new \mathbf{C} , update \mathbf{B} , etc. until convergence
 - ▶ Guaranteed to converge in nm steps for $m \times n$ matrices
 - ▶ Problem: requires solving the BU problem
 - ★ But it can be approximated
 - ▶ Problem: Converges too fast
 - ★ The optimization landscape is bumpy (many local optima)
- **Idea 2:** Find many dense submatrices (quasi-bicliques) and select from them
 - ▶ Existing algorithms find the dense submatrices
 - ▶ Finding the dense submatrices is slow
 - ▶ Problem: requires solving the BU problem

Using association accuracy: the Asso algorithm

- The Asso algorithm uses the correlations between rows to define *candidate factors*, from which it selects the final (column) factors
 - ▶ Assume two rows of \mathbf{A} share the same factor
 - ▶ Then both of these rows have 1s in the same subset of columns (assuming no noise)
 - ▶ Therefore the probability of seeing 1 in the other row on a column we've observed 1 on the other row is high
- Asso computes the empirical probabilities of seeing 1 in row i if it's seen in row j into $m \times m$ matrix
 - ▶ This matrix is rounded to binary
 - ▶ A greedy search selects a column of this matrix and its corresponding row factor to create the next component

Using association accuracy: the Asso algorithm

- The Asso algorithm uses the correlations between rows to define *candidate factors*, from which it selects the final (column) factors
 - ▶ Assume two rows of \mathbf{A} share the same factor
 - ▶ Then both of these rows have 1s in the same subset of columns (assuming no noise)
 - ▶ Therefore the probability of seeing 1 in the other row on a column we've observed 1 on the other row is high
- Asso computes the empirical probabilities of seeing 1 in row i if it's seen in row j into $m \times m$ matrix
 - ▶ This matrix is rounded to binary
 - ▶ A greedy search selects a column of this matrix and its corresponding row factor to create the next component
- Problem: requires solving the BU problem
 - ▶ Greedy heuristic works well in practice

Using association accuracy: the Asso algorithm

- The Asso algorithm uses the correlations between rows to define *candidate factors*, from which it selects the final (column) factors
 - ▶ Assume two rows of \mathbf{A} share the same factor
 - ▶ Then both of these rows have 1s in the same subset of columns (assuming no noise)
 - ▶ Therefore the probability of seeing 1 in the other row on a column we've observed 1 on the other row is high
- Asso computes the empirical probabilities of seeing 1 in row i if it's seen in row j into $m \times m$ matrix
 - ▶ This matrix is rounded to binary
 - ▶ A greedy search selects a column of this matrix and its corresponding row factor to create the next component
- Problem: requires solving the BU problem
 - ▶ Greedy heuristic works well in practice
- Problem: introduces a parameter to round the probabilities

Using association accuracy: the Asso algorithm

- The Asso algorithm uses the correlations between rows to define *candidate factors*, from which it selects the final (column) factors
 - ▶ Assume two rows of \mathbf{A} share the same factor
 - ▶ Then both of these rows have 1s in the same subset of columns (assuming no noise)
 - ▶ Therefore the probability of seeing 1 in the other row on a column we've observed 1 on the other row is high
- Asso computes the empirical probabilities of seeing 1 in row i if it's seen in row j into $m \times m$ matrix
 - ▶ This matrix is rounded to binary
 - ▶ A greedy search selects a column of this matrix and its corresponding row factor to create the next component
- Problem: requires solving the BU problem
 - ▶ Greedy heuristic works well in practice
- Problem: introduces a parameter to round the probabilities
- Problem: noisy or badly overlapping factors do not appear on the rounded matrix

Selecting the parameters: The MDL principle

- Typical matrix factorization methods require the user to pre-specify the rank
 - ▶ Also SVD is usually computed only up to some top- k factors
- With BMF, the minimum description length (MDL) principle gives a powerful way to automatically select the rank
- Intuition: data consists of structure and noise
 - ▶ Structure can be explained well using the factors
 - ▶ Noise cannot be explained well using the factors
- Goal: find the size of the factorization that explains all the structure but doesn't explain the noise
- Idea: Quantify how well we explain the data by how well we can compress it
 - ▶ If a component explains many 1s of the data, it's easier to compress the factors than each of the 1s

The MDL principle

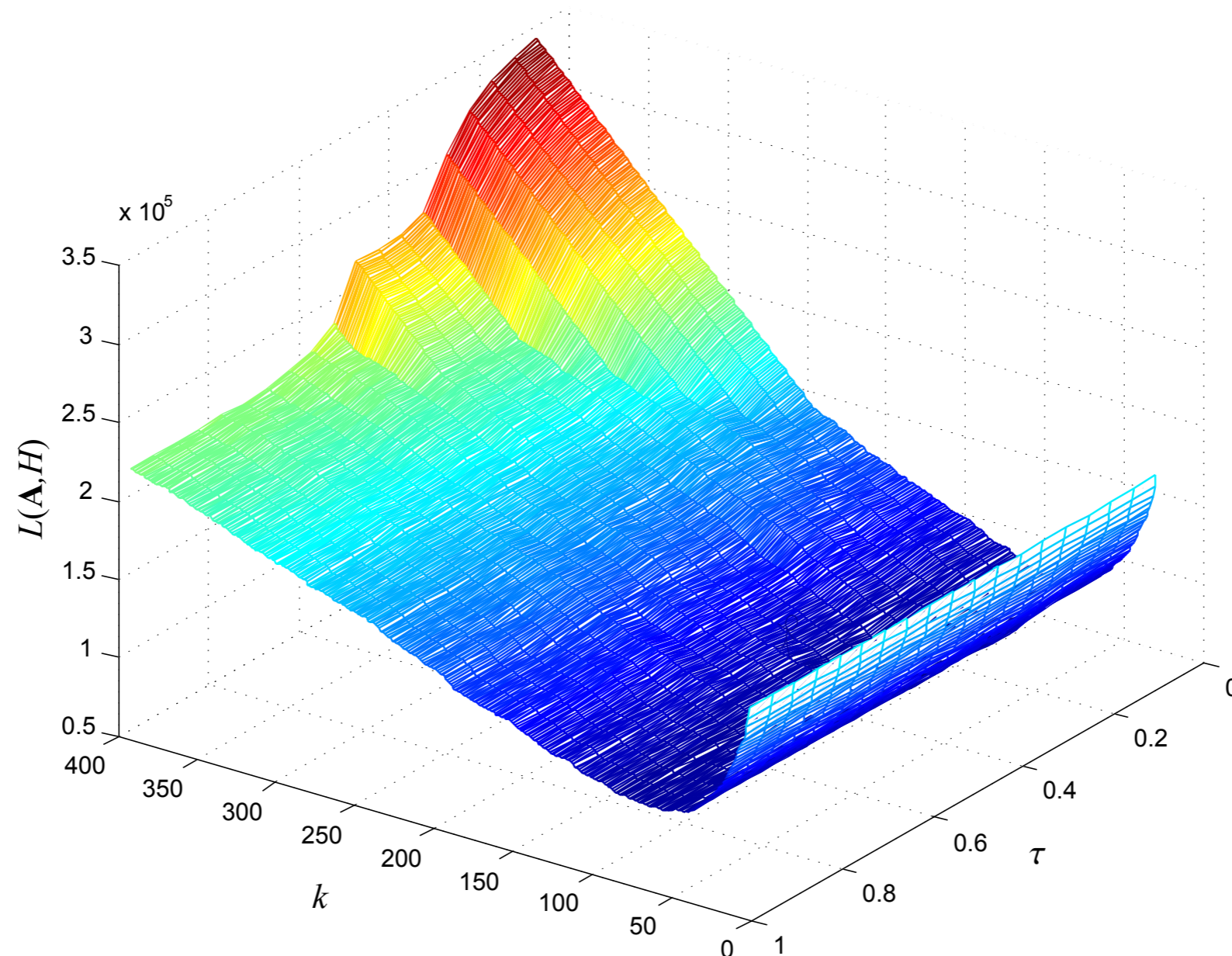
The best rank is the one that lets us to express the data with the least number of bits

MDL for BMF: Specifics

- We compress our data by compressing the factor matrices and the residual matrix
 - ▶ The residual is the exclusive or of the data and the factorization,
$$\mathbf{R} = \mathbf{A} \oplus (\mathbf{B} \boxtimes \mathbf{C})$$
 - ▶ The residual is needed because the compression must be lossless
- In MDL parlance, \mathbf{B} and \mathbf{C} constitute the **hypothesis** and \mathbf{R} explains the data given the hypothesis
- Question: how do we encode the matrices?
 - ▶ One idea: consider each column of \mathbf{B} separately
 - ▶ Encode the number of 1s in the column, call it b ($\log_2(m)$ bits when m is already known)
 - ▶ Enumerate every m -bit binary vector with b 1s in lexicographical order and send the number
 - ★ There are $\binom{m}{b}$ such vectors, so we can encode the number with $\log_2\left(\binom{m}{b}\right)$ bits
 - ★ We don't really need to do the enumeration, just to know how many (fractional) bits it would take

MDL for BMF: An Example

- MDL can be used to find all parameters for the algorithm, not just one
- To use MDL, run the algorithm with different values of k and select the one that gives the smallest description length
 - ▶ Usually approximately convex, so no need to try all values of k



Outline

- 1 Warm-Up
- 2 What is BMF
- 3 BMF vs. other three-letter abbreviations
- 4 Binary matrices, tiles, graphs, and sets
- 5 Computational Complexity
- 6 Algorithms
- 7 Wrap-Up**

Lessons learned

- BMF finds binary factors for binary data yielding binary approximation
 - easier interpretation, different structure than normal algebra
- Many problems associated with BMF are hard even to approximate
 - ▶ Boolean rank, minimum-error BMF, Basis Usage, ...
- BMF has very combinatorial flavour
 - algorithms are less like other matrix factorization algorithms
- MDL can be used to automatically find the rank of the factorization

Suggested reading

- Slides at http://www.mpi-inf.mpg.de/~pmiettinen/bmf_tutorial/material.html
- Miettinen et al. *The Discrete Basis Problem*, IEEE Trans. Knowl. Data Eng. 20(10), 2008.
 - ▶ Explains the Asso algorithm and the use of BMF (called DBP in the paper) in data mining
- Lucchese et al. *A unifying framework for mining approximate top-k binary patterns*. IEEE Trans. Knowl. Data Eng.
 - ▶ Explains the Panda+ algorithm
- Miettinen & Vreeken *MDL4BMF: Minimum Description Length for Boolean Matrix Factorization*, ACM Trans. Knowl. Discov. Data 8(4), 2014
 - ▶ Explains the use of MDL with BMF