Efficiently Factorizing Boolean Matrices using Proximal Gradient Descent

Anonymous Author(s)
Affiliation
Address
email

Abstract

Addressing the interpretability problem of NMF on Boolean data, Boolean Matrix Factorization (BMF) uses Boolean algebra to decompose the input into low-rank Boolean factor matrices. These matrices are highly interpretable and very useful in practice, but they come at the high computational cost of solving an NP-hard combinatorial optimization problem. To reduce the computational burden, we continuously relax BMF using a novel elastic-binary regularizer, from which we derive a proximal gradient algorithm. Through an extensive set of experiments, we demonstrate that our method works well in practice: On synthetic data, we show that our algorithm converges quickly, recovers the ground truth precisely, and estimates the true matrix rank robustly. On real-world data, we improve upon the state of the art in recall, loss, and runtime, and a case study from the medical domain confirms that our results are easily interpretable and semantically meaningful.

1 Introduction

Discovering groups in data and expressing them in terms of common concepts is a central problem in many scientific domains and business applications, including cancer genomics [24], neuroscience [14], and recommender systems [19, 9]. This problem is often addressed using variants of *matrix factorization*, a family of methods that decompose the target matrix into a set of typically low-rank factor matrices whose product approximates the input well. Prominent examples of matrix factorization are Singular Value Decomposition (SVD) [11], Principal Component Analysis (PCA) [11], and Nonnegative Matrix Factorization (NMF) [32, 22, 23]. These methods differ in how they constrain the matrices involved: SVD and PCA require orthogonal factors, while NMF constrains the target matrix and the factors to be nonnegative.

SVD, PCA, and NMF achieve interpretable results—unless the data is Boolean. In this case, which is ubiquitous in the real world, their results are hard to interpret directly, because the input domain differs from the output domain, thus requiring post-processing to extract useful information. *Boolean Matrix Factorization* (BMF) addresses this problem by seeking two low-rank *Boolean* factor matrices whose Boolean product is close to the Boolean target matrix [25]. The output matrices, now lying in the same domain as the input, are interpretable and useful, but they come at the computational cost of solving an NP-hard combinatorial optimization problem [31, 25, 26]. To make BMF applicable in practice, we need efficient approximation algorithms.

There are many ways to approximate BMF—for example, by exploiting its underlying combinatorial or spatial structure [25, 6, 5], using probabilistic inference [38, 37, 36], or solving the related Bi-Clustering problem [29, 30]. Although these approaches achieve impressive results, they fall short when the input data is large and noisy. Hence, we take a different approach to overcome BMF's computational barrier. Starting from an NMF-like optimization problem, we derive a continuous relaxation of the original BMF problem that allows intermediate solutions to be real-valued. Inspired

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

by the elastic-net regularizer [44], we introduce the novel *elastic binary (*ELB*) regularizer* to regularize towards Boolean factor matrices. We obtain an efficient-to-compute *proximal operator* from our ELB regularizer that projects relaxed real-valued factors towards being Boolean, which allows us to leverage fast gradient-based optimization procedures. In stark contrast to the state of the art [16, 17, 18], which requires heavy post-hoc post-processing to actually achieve Boolean factors, we ensure a Boolean outcome upon convergence, by gradually increasing the projection strength using a *regularization rate*. We combine our relaxation, efficient proximal operator, and regularization rate into an *Elastic Boolean Matrix Factorization* algorithm (ELBMF) that is scalable to large data, results in accurate reconstructions, and does so without relying on heavy post-processing procedures. Not confined to BMF, however, ELB and its rate can regularize, e.g., binary MF or bi-clustering [18].

In summary, our main contributions are as follows:

1. We introduce the ELB regularizer.
2. We overcome the computational hardness of BMF leveraging a novel relaxed BMF problem.
3. We efficiently solve the relaxed BMF problem using an optimization algorithm based on proximal gradient descent.

The remainder of the paper proceeds as follows. In Sec. 2, we formally introduce the BMF problem and its relaxation, define our ELB regularizer and its proximal point operator, and show how to ensure a Boolean outcome upon convergence. We discuss related work in Sec. 3, validate our method through an extensive set of experiments in Sec. 4, and conclude with a discussion in Sec. 5.

2 Theory

Our goal is to factorize a given Boolean target matrix into at least two smaller, low-rank Boolean factor matrices, whose product comes close to the target matrix. Since the factor matrices are Boolean, this product follows Boolean algebra, i.e., it is identical to the standard outer product on a field where addition obeys $1 + 1 = 1$. We define the Boolean product between two Boolean matrices as

$$[\mathrm{U} \circ \mathrm{V}]_{ij} = \bigvee_{l \in [k]} \mathrm{U}_{il} \mathrm{V}_{lk} \,, \tag{1}$$

where $\mathrm{U} \in \{0,1\}^{n \times k}$, $\mathrm{V} \in \{0,1\}^{k \times m}$, and $\mathrm{U} \circ \mathrm{V} \in \{0,1\}^{n \times m}$. This gives rise to the BMF problem.

Problem 1 (Boolean Matrix Factorization) *For a given target matrix* $\mathrm{A} \in \{0,1\}^{n \times m}$*, a given matrix rank* $\mathbb{N} \ni k \leq \min\{n, m\}$*, and let* $A \oplus B$ *be logical exclusive or, discover the factor matrices* $\mathrm{U} \in \{0,1\}^{n \times k}$ *and* $\mathrm{V} \in \{0,1\}^{k \times m}$ *that minimize*

$$\|\mathrm{A} - \mathrm{U} \circ \mathrm{V}\|_F^2 = \sum_{ij} \mathrm{A}_{ij} \oplus [\mathrm{U} \circ \mathrm{V}]_{ij} \,. \tag{2}$$

While beautiful in theory, we cannot solve this problem exactly for all but the smallest matrices, as it is NP-complete [26]. In practice, we have to rely on approximations. To do this, we relax the Boolean constraints of Eq. (2) to allow non-negative, *non-Boolean* 'intermediate' factor matrices during the optimization, allowing us to use linear algebra rather than Boolean algebra. In other words, we solve the non-negative matrix factorization (NMF) [32]

$$\|\mathrm{A} - \mathrm{U}\mathrm{V}\|_F^2 \,, \tag{3}$$

subject to $\mathrm{U} \in \mathbb{R}_+^{n \times k}$ and $\mathrm{V} \in \mathbb{R}_+^{k \times m}$. We can solve this problem efficiently, e.g., via a Gauss-Seidel scheme. Although efficient, using plain NMF, however, disregards the Boolean structure of our matrices and produces factor matrices from a different domain, which are consequently hard to interpret and potentially very dense. To benefit from efficient optimization and still arrive at Boolean outputs, we allow real-valued intermediate solutions and regularize them towards becoming Boolean.

To steer our optimization towards Boolean solutions, we penalize non-Boolean solutions using a regularizer. This idea has been explored in prior work. There exists the l_1-inspired PRIMP regularizer [16], which is $-\kappa[-|1 - 2x| + 1]$ for values inside $[0, 1]$ and ∞ otherwise, and the l_2-inspired bowl-shaped regularizer [43], which is $\lambda(x^2 - x)^2/2$ everywhere on the real line. Although

both have been successfully applied to BMF, both also have undesirable properties: The PRIMP regularizer penalizes well *inside* the interval $[0, 1]$ but is non-differentiable on the outside, while the bowl-shaped regularizer is differentiable and penalizes well *outside* the interval $[0, 1]$ but is almost flat on the inside. Hence, both regularizers are problematic if used individually. Combining them, however, yields a regularizer that penalizes non-Boolean values well across the full real line. To combine l_1- and l_2-regularization, we use the *elastic-net regularizer*,

$$r(x) = \kappa\|x\|_1 + \lambda\|x\|_2^2 \ ,$$

which, however, only penalizes *non-zero* solutions [44]. To penalize *non-Boolean* solutions, we combine two elastic-net regularizers into our (almost W-shaped) ELB *regularizer*,

$$R(X) = \sum_{x \in X} \min\{r(x), r(x-1)\} \ , \tag{4}$$

where $X \in \{\mathrm{U}, \mathrm{V}\}$. In Fig. 1, we show all three regularizers in the range of $[-1, 2]$, for $\lambda = \kappa = 0.5$. We see that only the ELB regularizer penalizes non-Boolean solutions across the full spectrum, and summarize our regularized relaxed BMF as follows.

Problem 2 (Elastic Boolean Matrix Factorization) *For a given target matrix* $\mathrm{A} \in \{0, 1\}^{n \times m}$ *and a given matrix rank* $\mathbb{N} \ni k \leq \min\{n, m\}$*, discover the factor matrices* $\mathrm{U} \in \mathbb{R}_+^{n \times k}$ *and* $\mathrm{V} \in \mathbb{R}_+^{k \times m}$ *that minimize*

$$\|\mathrm{A} - \mathrm{UV}\|_F^2 + R(\mathrm{U}) + R(\mathrm{V}) \ . \tag{5}$$

Although this is a relaxed problem, it is still non-convex. Therefore, we cannot straightforwardly solve it. However, the problem is suitable for the Gauss-Seidel optimization scheme. That is, we alternatingly fix one factor matrix to optimize the other. By doing so, we generate a sequence

$$\mathrm{U}_{t+1} \leftarrow \arg\min_{\mathrm{U}} \|\mathrm{A} - \mathrm{UV}_t\|_F^2 + R(\mathrm{U}) \ , \tag{6}$$

$$\mathrm{V}_{t+1} \leftarrow \arg\min_{\mathrm{V}} \|\mathrm{A} - \mathrm{U}_{t+1}\mathrm{V}\|_F^2 + R(\mathrm{V}) \ , \tag{7}$$

of simpler-to-solve sub-problems, until convergence. Now, each sub-problem is again a sum of two $f(X) + R(X)$ functions, where f is the loss $\|\cdot\|_F^2$, and $R(X)$ is the regularizer. This allows us to follow a proximal gradient approach, i.e., we use *Proximal Alternating Linear Minimization* (PALM) [7, 34]. In a nutshell, we minimize a sub-problem by following the gradient ∇f of f, to then use the proximal operator for R to nudge its outcome towards a Boolean solution. That is, for the gradients $\nabla_{\mathrm{U}} f = \mathrm{UVV}^\top - \mathrm{AV}^\top$ and $\nabla_{\mathrm{V}} f = \mathrm{U}^\top\mathrm{UV} - \mathrm{U}^\top\mathrm{A}$, we compute the step

$$\mathrm{prox}_R(X - \eta\nabla f) \ , \tag{8}$$

where η represents the step size, which we compute in terms of Lipschitz constant, rather than relying on a costly line-search [7]. To further improve the convergence properties, we make use of an *inertial* term that linearly combines X_t with X_{t-1} before applying Eq. (8) (see [34] for a detailed description). We now derive the proximal operator for the ELB regularizer, before discussing how we ensure that the factor matrices are Boolean, and summarizing our approach as an algorithm.

2.1 Proximal Mapping

Let $X \in \{\mathrm{U}, \mathrm{V}\}$, to solve the sub-problem from Eq. (7),

$$\arg\min_X \ell_{\kappa\lambda}(X) \text{ for } \ell_{\kappa\lambda}(X) = f(X) + R(X) \ , \tag{9}$$

we need a proximal operator [33] that projects values towards a regularized point. Formally in Appendix A, here in a nutshell: this operator is the solution to

$$\mathrm{prox}_R(X) = \arg\min_Y \frac{1}{2}\|X - Y\|_2^2 + R(X) \ . \tag{10}$$

As Eq. (10) is coordinate-wise solvable, we can reduce this equation to a scalar proximal operator

$$\mathrm{prox}_{\kappa\lambda}(x) = \arg\min_{y \in \mathbb{R}} \frac{1}{2}(x - y)^2 + R(y) \ . \tag{11}$$

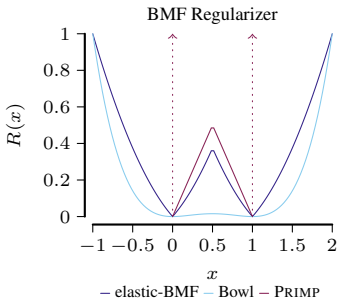


Figure 1: We show three BMF regularizer for $\lambda = \kappa = 0.5$.

Algorithm 1: ELBMF

Input: Matrix $\mathrm{A} \in \{0,1\}^{n \times m}$, rank $k \in \mathbb{N}$
Output: Factors $\mathrm{U} \in \{0,1\}^{n \times k}$, $\mathrm{V} \in \{0,1\}^{k \times m}$
initialize U, V uniformly at random
for $t = 1, 2, \ldots$ **until convergence do**
  $\mathrm{U} \leftarrow \arg\,\mathrm{reduce}_{\mathrm{U}}\, \ell_{\kappa\lambda_t}(\mathrm{A}, \mathrm{U}, \mathrm{V})$
  $\mathrm{V} \leftarrow \arg\,\mathrm{reduce}_{\mathrm{V}}\, \ell_{\kappa\lambda_t}(\mathrm{A}, \mathrm{U}, \mathrm{V})$
end

Although this scalar function may seem simple, due to R, it is non-convex, which—in general—rules out the existence of unique minima. We can, however, exploit the W-shape of R. That is, from the perspective of a fixed X_{ij}, the regularizer is locally convex after a case distinction. Starting with the case of $X_{ij} = x$, $x \leq 1/2$, we can simplify Eq. (11) to $\frac{1}{2}(x-y)^2 + \lambda'/2y^2 + \kappa|y|$, letting $\lambda' = 2\lambda$. Setting its derivative to 0, we get to $y = x - \kappa\,\mathrm{sign}(y)$. Asserting that a least-squares solution will always be of the same sign, we substitute the sign of x with $\mathrm{sign}(y)$. Repeating these steps analogously for $x > 1/2$, we obtain our proximal operator:

Definition 1 (Proximal Operator) *Given regularization coefficients κ and λ, our proximal operator for matrix X is $\mathrm{prox}_R(X) = [\mathrm{prox}_{\kappa\lambda}(X_{ij})]_{ij}$, where $\mathrm{prox}_{\kappa\lambda}$ is the scalar proximal operator*

$$\mathrm{prox}_{\kappa\lambda}(x) \equiv (1+\lambda)^{-1} \begin{cases} x - \kappa\,\mathrm{sign}(x) & \text{if } x \leq \frac{1}{2} \\ x - \kappa\,\mathrm{sign}(x-1) + \lambda & \text{otherwise}\,. \end{cases} \tag{12}$$

Although not strictly necessary, to improve the empirical convergence rate, we would like to constrain our factor matrices to be non-negative. As our proximal operator does not account for this, we impose non-negativity by using the *alternating projection* procedure to combine the non-negativity proximal operator [33] with Eq. (12) into $\max\{0, \mathrm{prox}_{\kappa\lambda}(x)\}$.

2.2 Ensuring Boolean Factors

Our proximal operator only nudges the factor matrices towards *becoming* Boolean. We, however, want to ensure that our results *are* Boolean. To this end, the state-of-the-art method PRIMP relies heavily on post-processing, performing a very expensive joint two-dimensional grid search to guess the 'best' pair of rounding thresholds, which are then used to produce Boolean matrices. Although this tends to work in practice, it is an inefficient post-hoc procedure—and thus, it would be highly desirable to have Boolean factors already upon convergence. To achieve this without rounding or clamping, we revisit our regularizer, which binarizes more strongly if we regularize more aggressively. Consequently, if we regularize too aggressively, we converge to a suboptimal solution, and if we regularize too mildly, we do not binarize our solutions. To prevent subpar solutions and still binarize our output, we start with a weak regularization and gradually increase its strength.

Considering Eq. (12), we see that a higher regularization strength increases the distance over which our proximal operator projects. Thus, if we set the l_1-distance controlling κ too high, we will immediately leap to a Boolean factor matrix, which will terminate the algorithm and yield a suboptimal solution. Regulating the l_2-distance controlling λ is a less delicate matter. Hence, we gradually increase λ to prevent a subpar solution and achieve a Boolean outcome, using a *regularization rate* $\lambda_t = \lambda \cdot \nu_t$ that gradually increases the proximal distance at a user-defined rate $\nu_t \geq 0\, \forall t \geq 0$. In case ELBMF has stopped without convergence, we bridge the remaining integrality gap by projecting the outcome onto their closed Boolean counterpart, using our proximal operator (see Fig. 2).

We summarize the considerations laid out above as ELBMF in Alg. 1. The computational complexity of ELBMF is bounded by the complexity of computing the gradient, which is identical to the complexity of matrix multiplication. Therefore, for all practical purposes, ELBMF is sub-cubic $\mathcal{O}\left(n^{2.807}\right)$ using Strassen's algorithm.

3 Related Work

Matrix factorization is a well-established family of methods, whose members, such as SVD, PCA, or NMF, are used everywhere in machine learning. Almost all matrix factorization methods operate on real-valued matrices, however, while BMF operates under Boolean algebra. *Boolean Matrix Factorization* originated in the combinatorics community [28] and was later introduced to the data mining community [25], where many cover-based BMF algorithms were developed [6, 5, 25, 27]. In recent years, BMF has gained traction in the machine learning community, which tends to tackle the problem differently. Here, *relaxation-based approaches* that optimize for a relaxed but regularized BMF [16, 15, 43] are related to our method, but they differ especially in their regularization. Hess et al. [16] introduce a regularizer that is only partially differentiable and rely heavily on post-processing to force a Boolean solution, and Zhang et al. [43] regularize only weakly between 0 and 1. In contrast, our regularizer penalizes well across the full spectrum and yields a Boolean outcome upon convergence. Building on a thresholding-based BMF formulation, Araujo et al. [3] also considers relaxations to benefit from gradient-based optimization. Other recent approaches build on *probabilistic inference*. Rukat et al. [38, 37, 36], for example, combine Bayesian Modeling and sampling into their logical factor machine. A similar direction is taken by Ravanbakhsh et al. [35], who use graphical models and message passing, and Liang et al [24], who combine MAP-inference and sampling. A different, *geometry-based approach* lies in locating dense submatrices by ordering the data to exploit the consecutive-ones property [41, 39]. Since BMF is essentially solving a bipartite graph partitioning problem, it is also closely related to Bi-Clustering and Co-Clustering [29, 18]. Neumann and Miettinen [30] use this relationship to efficiently solve BMF by means of a streaming algorithm. Although there are many different approaches to BMF, its biggest challenge to date remains scalability [26].

4 Experiments

We implement ELBMF in the Julia language and run experiments on 16 Cores of an AMD EPYC 7702 and a single NVIDIA A100 GPU, reporting wall clock time. We provide the source code, datasets, synthetic dataset generator, and other information needed for reproducibility.[1] We compare ELBMF against six methods: four dedicated BMF methods (ASSO [25], GRECOND [6], ORM [37], and PRIMP [16]), one streaming Bi-Clustering algorithm SOFA [30], one elastic-net-regularized NMF method leveraging proximal gradient descent (NMF [32, 22, 23]), and one interpretable Boolean autoencoder (BINAPS [10]). Since NMF outputs non-negative factor matrices, rather Boolean matrices, we cannot compare against NMF directly, so we clamp and round its solutions to the nearest Boolean outcome. To fairly compare against BINAPS, we task it with autoencoding the target matrix as a reconstruction, given the matrix ranks from our experiments as the number of latent dimensions. We perform three sets of experiments. First, we ascertain that ELBMF works reliably on synthetic data. Second, we verify that it generally performs well on real-world data. And third, we illustrate that its outputs are semantically meaningful through an exploratory analysis of a biomedical dataset.

4.1 Performance of ELBMF on Synthetic Data

In the following experiments, we ask three questions: (**1**) Does ELBMF converge?; (**2**) How well does ELBMF recover the information in the target matrix?; and (**3**) Does ELBMF estimate the true Boolean matrix rank correctly? To answer these questions, we generate synthetic data with known ground truth as follows. Starting with an all-zeros matrix, we randomly create rectangular, non-overlapping, consecutive areas of ones called *tiles*, each spanning between 5 and 10 rows and columns, thus inducing matrices with varying densities. We then add noise by setting each cell to 1, uniformly at random, with varying noise probabilities.

How does ELBMF converge? To study how our method converges to a Boolean solution, we quantify relevant properties of the sequence of intermediate solutions (cf. Eq. (7)). First, to understand how quickly and stably ELBMF converges to a Boolean solution, we quantify the *Boolean gap*,

$$\sum_{X \in \{U_t, V_t\}} |X|^{-1} \sum_{x \in X} \min\{|x|, |x-1|\} \ .$$

[1] Appendix B, dropbox.com/sh/gaoftnf0iyh27xq/AADRlhKxTV1S9LovwwnxVTCYa

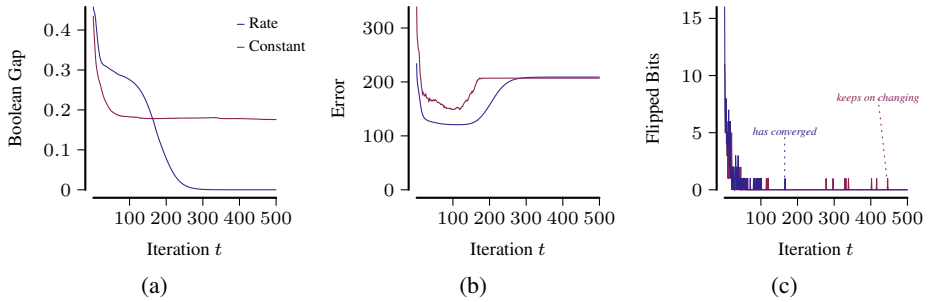


Figure 2: Our method converges quickly under a regularization rate. We report the progression over time of the *Boolean gap* (a), *loss* and *loss of rounded factors* (b), as well as the *Hamming process* (c) of ELBMF on synthetic 40×30 matrices with 10% noise, under a constant regularization of $\lambda_t = 1$ or a regularization rate of $\lambda_t = 1.05^t$.

Second, to understand when we can safely round intermediate almost-Boolean solutions without losing information, we compute for the *rounded* intermediate reconstruction B, the fraction of bits that flip from iteration t to iteration $t+1$ as the *Hamming process*

$$|A|^{-1}\|\mathrm{B}_t - \mathrm{B}_{t+1}\|_1 \ .$$

As shown in Fig. 2a, we achieve an almost-Boolean solution *without any rounding* after around 250 epochs, continuing until we reach a Boolean outcome. This is also the point at which the rounded intermediate solution and its relaxation are almost identical, as illustrated in Fig. 2b. Considering the *rounded* solution sequence in Fig. 2c, we observe that ELBMF goes through an erratic bit-flipping-phase in the beginning, followed by only minor changes in each iteration until iteration $t = 100$. Afterwards, ELBMF has settled on a solution—under our regularization rate. When using constant regularization instead, we continue to observe bit flips until the end of the experiment (Fig. 2c). The constant regularization fits the data slightly better than the regularization rate (Fig. 2b), which is unsurprising, given that its factors are less constrained than the Boolean factors we use otherwise. However, under constant regularization, the Boolean gap hardly decreases over time (Fig. 2a). In other words, our regularization works well, and it allows us to safely binarize almost-converged factors that are ϵ-far from being Boolean by means of, e.g., our proximal operator.

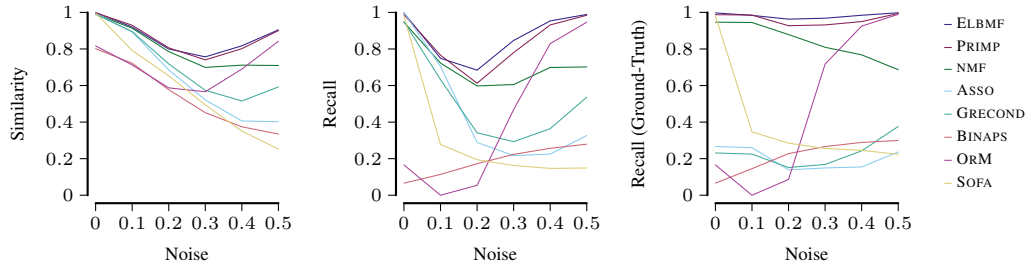
How well does ELBMF recover the information in the target matrix? Having ensured that our method converges stably and quickly, we would like to assess whether it also converges to a high-quality factorization. To this end, we generate synthetic 40×30 matrices containing 5 random tiles, under additive noise levels between 0% (no noise) and 50%. We then compute the fraction of *ones* in target A that is covered by the reconstruction $\mathrm{B} = \mathrm{U} \circ \mathrm{V}$, i.e., the *recall* (higher is better)

$$\|\mathrm{A}\|_1^{-1}\|\mathrm{A} \odot \mathrm{B}\|_1 \ .$$

To ensure that we fit the *signal* in the data, we additionally report the recall regarding the generating, noise-free ground-truth tiles A^*, denoted as recall*. Finally, to rate the overall quality of the reconstruction including *zeros*, we compute the *Hamming similarity* (higher is better) between the target matrix and its reconstruction

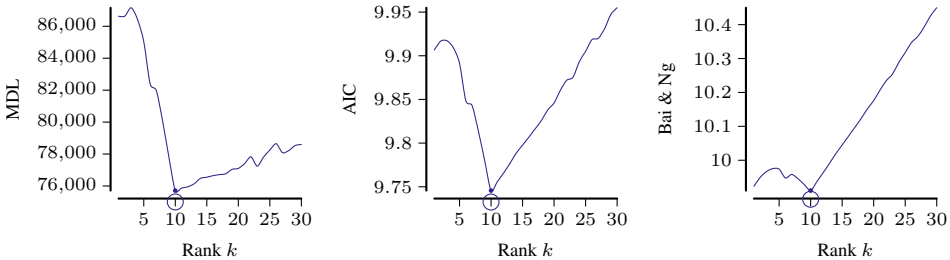
$$|\mathrm{A}|^{-1}\|\mathrm{A} - \mathrm{B}\|_1 \ .$$

We run each method on our synthetic datasets, targeting a matrix rank of 5. To account for random fluctuations, we average over 10 randomly drawn sets of 5 ground-truth tiles per 10% increment in noise probability. In Fig. 3, we show similarity, recall, and recall*. We observe that in the noiseless case (0%), all methods except BINAPS recover the 5 ground-truth tiles with high accuracy, but only ASSO and ELBMF do so with perfect recalls. Starting with as little as 10% noise, both recalls of ASSO, GRECOND, SOFA, NMF, and BINAPS deteriorate quickly, while the similarity and both recalls of PRIMP and ELBMF remain high. In fact, ELBMF and PRIMP perform similarly across the board—which is highly encouraging, as unlike PRIMP, ELBMF does not require post-processing. For ASSO and GRECOND, recall and similarity drop considerably, but they exhibit a slightly higher recall*. This means that these methods are robust against noise, but they fail to recover the remaining information. Starting low, ORM's recalls increase jointly with the noise level, suggesting that clean data is problematic for ORM.



(a) (b) (c)

Figure 3: Overall, ELBMF reconstructs the noisy synthetic data well and recovers the ground-truth tiles. On synthetic data for additive noise levels increasing from 0% to 50%, we show *similarity* (a) and *recall* (b) w.r.t. the target matrix, and recall* w.r.t. the noise-free ground-truth tiles, for ASSO, GRECOND, ORM, SOFA, NMF, PRIMP, and ELBMF.



(a) (b) (c)

Figure 4: ELBMF robustly detects the correct rank and does not overfit the data under different model selection criteria, such as MDL, AIC, or Bai & Ng's first information criterion.

Does ELBMF estimate the true Boolean matrix rank correctly? When the true matrix rank is known, we can immediately apply ELBMF to factorize the data. In the real world, however, the true matrix rank might be unknown. In this case, we need to estimate an appropriate choice from the data, and we use synthetic data to ensure that ELBMF does so correctly.

Since a higher matrix rank usually also means a better fit, selecting the best rank according to recall, loss, or similarity leads to overfitting—unless we properly account for the growth in model complexity. There are many *model selection criteria* that penalize complex models, such as AIC [1], Bai & Ng's criteria [4, 2], Nuclear-norm regularizing [21, 13], the information-theoretic Minimal Description Length principle (MDL) [12], or (Decomposed) Normalized Maximum Likelihood [20, 42]. Following common practice and motivated by its practical performance in preliminary experiments, we choose MDL. That is, we select the *minimizer* of sum the log binomial $l(X) = \log \binom{|X|}{\|X\|_1}$ of the error matrix and the rows and columns of our factorization (assumed to be i.i.d.) [27]

$$l(\mathrm{A} \oplus [\mathrm{U} \circ \mathrm{V}]) + \textstyle\sum_{i \in [k]} l(\mathrm{U}_i^\top) + l(\mathrm{V}_i) + k \log(n \cdot m)\,.$$

To validate whether ELBMF recovers the right rank, we synthetically generate a ground-truth 400×300 matrix of rank 10 with 10% additive noise. In Fig. 4, we show MDL, AIC, and Bai & Ng's first criterion for each rank up to 30, finding that our method precisely discovers the true matrix rank.

4.2 Performance of ELBMF on Real-World Data

Having ascertained that ELBMF works well on synthetic data, we turn to its performance in the real world. Here, we use nine publicly available datasets and anonymized[2] from different domains.

[2] GRAND.NETWORKMEDICINE.ORG STRING-DB.ORG CANCER.GOV/TCGA INTERNATIONALGENOME.ORG PATENTSVIEW.ORG GROUPLENS.ORG/DATASETS/MOVIELENS KAGGLE.COM/DATASETS/NETFLIX-INC/NETFLIX-PRIZE-DATA

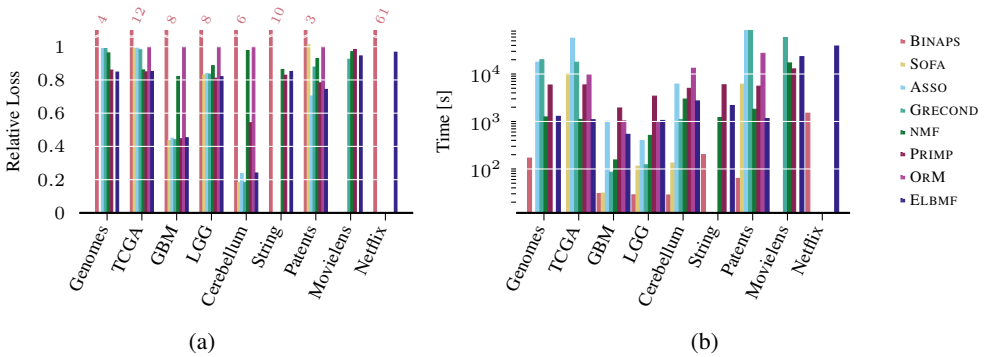


Figure 5: Our method factorizes real-world data with low relative loss and runtime. We report relative loss (a) and runtime (b) of real-world matrices and their reconstructions from ASSO, GRECOND, ORM, SOFA, BINAPS, NMF, and ELBMF. Tilted numbers in (a) report relative loss of BINAPS.

To cover the *biomedical domain*, we extract the network containing empirical evidence of protein-protein interactions in *homo sapiens* from the STRING database. From the GRAND repository, we take the gene regulatory networks sampled from *Glioblastoma (GBM)* and *Lower Grade Glioma (LGG)* brain cancer tissues, as well as from non-cancerous *Cerebellum* tissue. The *TCGA* dataset contains binarized gene expressions from cancer patients, and we further obtain the single nucleotide polymorphism (SNP) mutation data from the $1k$ *Genomes* project, following processing steps from the authors of BINAPS [10]. In the *entertainment domain*, we use the user-movie datasets *Movielens* and *Netflix*, binarizing the original 5-star-scale ratings by setting only reviews with more than 3.5 stars to 1. Finally, as data from the *innovation domain*, we derive a directed citation network between patent groups from patent citation and classification data provided by *PatentsView*. For each dataset with a given number of groups, such as cancer types or movie genres, we set the matrix rank k to 33 (TCGA), 28 (Genomes), 136 (Patents), 20 (Movielens), and 20 (Netflix). When the number of subgroups is unknown, we estimate the rank using MDL and ELBMF, resulting in 100 (GBM), 32 (LGG), 100 (String), and 450 (Cerebellum).

As we can achieve a high similarity with an all-zeros reconstruction (in the case of sparse data), or a perfect recall with an all-ones reconstruction, we also report the *relative loss* (lower is better),

$$\|\mathrm{A}\|_1 \|\mathrm{A} - \mathrm{U} \circ \mathrm{V}\|_1 \,,$$

between the target matrices and their reconstructions. In Fig. 5, we show recall and runtime of ASSO, GRECOND, ORM, SOFA, BINAPS, NMF, and ELBMF, applied to all datasets. We see in Fig. 5a that ELBMF often has the lowest loss, outperforming ASSO, GRECOND, ORM, SOFA, BINAPS, and NMF. In many cases, ELBMF also has a loss lower than PRIMP. We observe the largest gap between the two on the Cerebellum dataset, where PRIMP's grid search procedure fails to find suitable thresholds. This is a highly impressive result, as ELBMF does not necessitate heavy post-processing, which is required by PRIMP. In Fig. 5b (log scale), we see how post-processing degrades runtime, observing that PRIMP is always much slower than ELBMF. For example, PRIMP does not finish on the Netflix data, where only ELBMF and BINAPS succeeded. Although similarly fast as ELBMF, ORM results in a significantly higher loss. Notably, without much extra care, NMF's performance is surprisingly good: it often outperforms the dedicated BMF methods ASSO, GRECOND, ORM, and SOFA.

4.3 Exploratory Analysis of Gene Expression Data with ELBMF

Knowing that ELBMF performs well quantitatively, we ask whether its outputs are also interpretable. To this end, we take a closer look at the *TCGA* data, which contains the expression levels of $20\,530$ genes from $10\,459$ patients, who are labeled with 33 cancer types. Since we are interested in retaining high gene expression levels only, we set expression levels to one if their Z-scores fall into the top 5% quantile, and to zero otherwise [24]. We run ELBMF on this dataset, targeting a rank of 33.

To learn whether our method groups patients meaningfully, we visualize the target matrix and its reconstruction. As the target matrix is high-dimensional, we embed both the target matrix and the reconstruction into a two-dimensional space using t-SNE [40] as illustrated in Fig. 6, where each

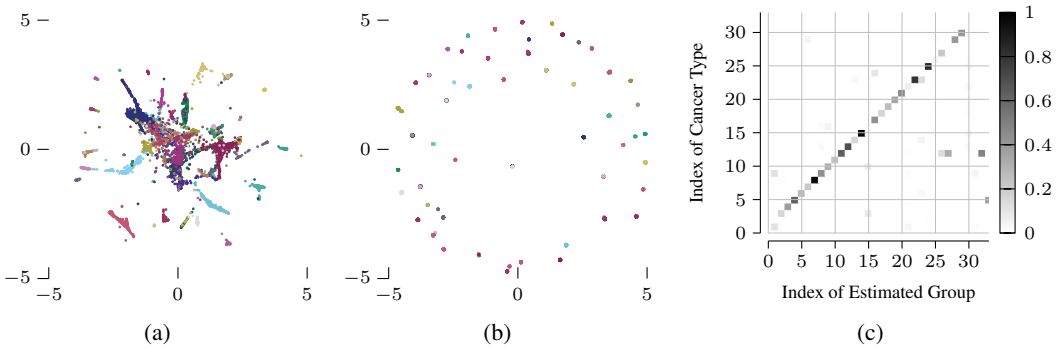


Figure 6: ELBMF discovers the hidden structure in gene expression data. We show the two-dimensional t-SNE embedding of the TCGA dataset (a) and the embedding of its reconstruction from ELBMF (b), where each point corresponds to one of the 10 459 patients, colored by cancer type. While the cancer types are hard to differentiate in the embedding of the original dataset (a), they are separated into easily distinguishable clusters in the embedding of our reconstruction (b). In (c), we show the renormalized mutual information between estimated groups and cancer types, and observe that there is an almost 1-to-1 correspondence between our estimated groups and the cancer types.

color corresponds to one cancer type. In Fig. 6a, we show that when embedding the target matrix directly, the cancer types are highly overlapping and hard to distinguish without the color coding. In contrast, when embedding our reconstruction, depicted in Fig. 6b, we see a clean segmentation into clusters that predominantly contain a single cancer type.

To better understand these results, we quantify the association between our 33 estimated groups and the ground-truth cancer types by computing the *mutual information* matrix. This matrix is noticeably sparse (Fig. 6c), leading to a clean segmentation. Upon closer inspection with ENRICHR [8], the associations we discover turn out to be biologically meaningful. For example, we find that ELBMF associates a group of 356 genes to patients with *thyroid carcinoma*. This group is associated with *thyroid hormone generation* and *thyroid gland development*, and *statistically significantly* so—even under a strict False Discovery Control, with p-values as low as 2.574×10^{-8} and 6.530×10^{-6}.

5 Conclusion

We introduced ELBMF to efficiently factorize Boolean matrices using an elegant and simple algorithm that, unlike its closest competitors, does not rely on heavy post-processing. ELBMF considers a novel relaxed BMF problem, which allows intermediate solutions to be real-valued. It solves this problem by leveraging an efficiently computable proximal operator, derived from the innovative ELB regularizer, and using a regularization rate to obtain Boolean factors upon convergence. Experimentally, we have shown that ELBMF works well in practice. It operates reliably on synthetic data, outperforms the discrete state of the art, is at least as good as the best relaxations on real data, and yields interpretable results even in difficult domains—without relying on post-processing.

Limitations Although ELBMF works overall, it has two bottlenecks. First, by randomly initializing factors, we start with highly dense matrices, thus prohibiting efficient sparse matrix operations. This is not ideal for sparse datasets that are too large to fit into memory, and future research on sparse initialization of *Boolean* factors will benefit not only ELBMF but also many other methods. Second, the larger the datasets, the higher the cost of computing gradients, and future work might adapt stochastic gradient methods for ELBMF to mitigate this problem.

Broader Impact ELBMF is a method for factorizing Boolean matrices, which permits interpretable, rather than black box data analysis. As such, it can help to make explicit any biases that may be present in the data. Just like any other method for interpretable data analysis, the insights they yield us can be used for good (e.g., insight into cancer-causing mutations) or for bad, and the ultimate quality of the outcome also depends on the behavior of the user. In our experiments we focus on *good* applications, and only consider anonymized open data.

References

[1] H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, 1974. Conference Name: IEEE Transactions on Automatic Control.

[2] Lucia Alessi, Matteo Barigozzi, and Marco Capasso. Improved penalization for determining the number of factors in approximate factor models. *Statistics & Probability Letters*, 80(23-24):1806–1813, 2010.

[3] Miguel Araujo, Pedro Manuel Pinto Ribeiro, and Christos Faloutsos. Faststep: Scalable boolean matrix decomposition. In *PAKDD*, pages 461–473. Springer, 2016.

[4] Jushan Bai and Serena Ng. Determining the number of factors in approximate factor models. *Econometrica*, 70(1):191–221, 2002.

[5] Radim Belohlavek and Martin Trnecka. A new algorithm for boolean matrix factorization which admits overcovering. *Discrete Appl Math*, 249:36–52, November 2018.

[6] Radim Belohlavek and Vilem Vychodil. Discovery of optimal factors in binary data via a novel method of matrix decomposition. *J Comput Syst Sci*, 76(1):3–20, February 2010.

[7] Jérôme Bolte, Shoham Sabach, and Marc Teboulle. Proximal alternating linearized minimization for nonconvex and nonsmooth problems. *Appl Manag Sci*, 146(1):459–494, August 2014.

[8] Edward Y Chen, Christopher M Tan, Yan Kou, Qiaonan Duan, Zichen Wang, Gabriela Vaz Meirelles, Neil R Clark, and Avi Ma'ayan. Enrichr: interactive and collaborative html5 gene list enrichment analysis tool. *BMC bioinformatics*, 14(1):1–14, 2013.

[9] Mamadou Diop, Sebastian Miron, Anthony Larue, and David Brie. Binary matrix factorization applied to netflix dataset analysis. *IFAC*, 52(24):13–17, 2019.

[10] Jonas Fischer and Jilles Vreeken. Differentiable pattern set mining. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, pages 383–392, New York, NY, USA, August 2021. Association for Computing Machinery.

[11] Gene H. Golub and Charles F. Van Loan. *Matrix Computations, Third Edition*. Johns Hopkins University Press, 1996.

[12] Peter D Grünwald. *The minimum description length principle*. MIT press, 2007.

[13] Suriya Gunasekar, Blake E Woodworth, Srinadh Bhojanapalli, Behnam Neyshabur, and Nati Srebro. Implicit regularization in matrix factorization. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[14] Ali Haddad, Foroogh Shamsi, Li Zhu, and Laleh Najafizadeh. Identifying dynamics of brain function via boolean matrix factorization. In *2018 52nd Asilomar Conference on Signals, Systems, and Computers*, pages 661–665, 2018. ISSN: 2576-2303.

[15] Sibylle Hess and Katharina Morik. C-SALT: Mining class-specific ALTerations in boolean matrix factorization. In Michelangelo Ceci, Jaakko Hollmén, Ljupčo Todorovski, Celine Vens, and Sašo Džeroski, editors, *Machine Learning and Knowledge Discovery in Databases*, Lecture Notes in Computer Science, pages 547–563, Cham, 2017. Springer International Publishing.

[16] Sibylle Hess, Katharina Morik, and Nico Piatkowski. The PRIMPING routine—tiling through proximal alternating linearized minimization. *Data Min Knowl Disc*, 31(4):1090–1131, July 2017.

[17] Sibylle Hess, Nico Piatkowski, and Katharina Morik. The trustworthy pal: Controlling the false discovery rate in boolean matrix factorization. *CoRR*, abs/1907.00697, 2019.

[18] Sibylle Hess, Gianvito Pio, Michiel E. Hochstenbach, and Michelangelo Ceci. BROCCOLI: overlapping and outlier-robust biclustering through proximal stochastic gradient descent. *Data Min. Knowl. Discov.*, 35(6):2542–2576, 2021.

[19] Dmitry I Ignatov, Elena Nenova, Natalia Konstantinova, and Andrey V Konstantinov. Boolean matrix factorisation for collaborative filtering: An fca-based approach. In *International Conference on Artificial Intelligence: Methodology, Systems, and Applications*, pages 47–58. Springer, 2014.

[20] Yu Ito, Shin-ichi Oeda, and Kenji Yamanishi. Rank selection for non-negative matrix factorization with normalized maximum likelihood coding. In *Proceedings of the 2016 SIAM International Conference on Data Mining*, pages 720–728. Society for Industrial and Applied Mathematics, June 2016.

[21] Martin Jaggi and Marek Sulovský. A simple algorithm for nuclear norm regularized problems. In *ICML*, January 2010.

[22] Daniel D. Lee and H. Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, October 1999. Number: 6755 Publisher: Nature Publishing Group.

[23] Daniel D. Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. In Todd K. Leen, Thomas G. Dietterich, and Volker Tresp, editors, *Advances in Neural Information Processing Systems 13, Papers from Neural Information Processing Systems (NIPS) 2000, Denver, CO, USA*, pages 556–562. MIT Press, 2000.

[24] Lifan Liang, Kunju Zhu, and Songjian Lu. BEM: Mining coregulation patterns in transcriptomics via boolean matrix factorization. *Bioinformatics (Oxford, England)*, 36(13):4030–4037, July 2020.

[25] Pauli Miettinen, Taneli Mielikäinen, Aristides Gionis, Gautam Das, and Heikki Mannila. The discrete basis problem. *IEEE transactions on knowledge and data engineering*, 20(10):1348–1362, 2008.

[26] Pauli Miettinen and Stefan Neumann. Recent developments in boolean matrix factorization. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, pages 4922–4928, Yokohama, Japan, July 2020. International Joint Conferences on Artificial Intelligence Organization.

[27] Pauli Miettinen and Jilles Vreeken. Mdl4bmf: Minimum description length for boolean matrix factorization. *ACM transactions on knowledge discovery from data (TKDD)*, 8(4):1–31, 2014.

[28] Sylvia D Monson, Norman J Pullman, and Rolf Rees. A survey of clique and biclique coverings and factorizations of (0, 1)-matrices. *Bull. Inst. Combin. Appl*, 14:17–86, 1995.

[29] Stefan Neumann. Bipartite stochastic block models with tiny clusters. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 3871–3881, 2018.

[30] Stefan Neumann and Pauli Miettinen. Biclustering and boolean matrix factorization in data streams. *Proceedings of the VLDB Endowment*, 13(10):1709–1722, June 2020.

[31] James Orlin. Contentment in graph theory: Covering graphs with cliques. *Indagationes Mathematicae (Proceedings)*, 80(5):406–424, January 1977.

[32] Pentti Paatero and Unto Tapper. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5(2):111–126, 1994. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/env.3170050203.

[33] Neal Parikh and Stephen Boyd. Proximal algorithms. *Foundations and Trends in optimization*, 1(3):127–239, 2014.

[34] Thomas Pock and Shoham Sabach. Inertial proximal alternating linearized minimization (iPALM) for nonconvex and nonsmooth problems. *SIAM Journal on Imaging Sciences*, 9(4):1756–1787, January 2016.

[35] Siamak Ravanbakhsh, Barnabas Poczos, and Russell Greiner. Boolean matrix factorization and noisy completion via message passing. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 945–954. PMLR, 2016. ISSN: 1938-7228.

[36] Tammo Rukat, Chris Holmes, and Christopher Yau. Probabilistic boolean tensor decomposition. In *Proceedings of the 35th International Conference on Machine Learning*, pages 4413–4422. PMLR, July 2018. ISSN: 2640-3498.

[37] Tammo Rukat, Chris C. Holmes, Michalis K. Titsias, and Christopher Yau. Bayesian boolean matrix factorisation. In *Proceedings of the 34th International Conference on Machine Learning*, pages 2969–2978. PMLR, July 2017. ISSN: 2640-3498.

[38] Tammo Rukat, Dustin Lange, and Cédric Archambeau. An interpretable latent variable model for attribute applicability in the amazon catalogue. *NeurIPS*, 2017.

[39] Nikolaj Tatti and Pauli Miettinen. Boolean matrix factorization meets consecutive ones property. In *Proceedings of the 2019 SIAM International Conference on Data Mining*, pages 729–737. SIAM, 2019.

[40] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[41] Changlin Wan, Wennan Chang, Tong Zhao, Mengya Li, Sha Cao, and Chi Zhang. Fast and efficient boolean matrix factorization by geometric segmentation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(4):6086–6093, April 2020.

[42] Tianyi Wu, Shinya Sugawara, and Kenji Yamanishi. Decomposed normalized maximum likelihood codelength criterion for selecting hierarchical latent variable models. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1165–1174, Halifax NS Canada, August 2017. ACM.

[43] Zhongyuan Zhang, Tao Li, Chris Ding, and Xiangsun Zhang. Binary matrix factorization with applications. In *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, pages 391–400, October 2007. ISSN: 2374-8486.

[44] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *J. R. Stat. Soc. B*, 67(2):301–320, 2005.

6 Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Sec. 5.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Sec. 5.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Sec. 2.
   (b) Did you include complete proofs of all theoretical results? [Yes] See Sec. 2 and appendix.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See Sec. 4, appendix, and under dropbox.com/sh/gaoftnf0iyh27xq/AADRlhKxTV1S9LovwwnxVTCYa.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Sec. 4 and appendix.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See appendix.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Sec. 4.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Sec. 4.
   (b) Did you mention the license of the assets? [Yes] See Sec. 4.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] See Sec. 4, appendix, and under dropbox.com/sh/gaoftnf0iyh27xq/AADRlhKxTV1S9LovwwnxVTCYa.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] See Sec. 4. All datasets are publicly available.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] See Sec. 4. All datasets have been anonymized prior to their publication.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Appendix

A Derivation of the Proximal Operator

In this section, we derive our proximal operator (given in Eq. (12), repeated below)

$$\text{prox}_{\kappa\lambda}(x) \equiv (1+\lambda)^{-1} \begin{cases} x - \kappa\,\text{sign}(x) & \text{if } x \leq \frac{1}{2} \\ x - \kappa\,\text{sign}(x-1) + \lambda & \text{otherwise} \end{cases}, \tag{13}$$

from the ELB (Eq. (4), repeated below)

$$R(X) = \sum_{x \in X} \min\{r(x), r(x-1)\} \; .$$

Starting with the definition [33] of the general proximal operator

$$\arg\min_{Y} \frac{1}{2}\|X - Y\|_2^2 + R(X) \; ,$$

we observe that this proximal operator is coordinate-wise solvable. This allows us to derive a *scalar* proximal operator, which we then apply to each value in the matrix independently. Substituting the regularizer R with its scalar version, we obtain a scalar proximal operator (Eq. (11))

$$\text{prox}_{\kappa\lambda}(x) = \arg\min_{y \in \mathbb{R}} \frac{1}{2}(x-y)^2 + \min\{r(y), r(y-1)\} \; ,$$

which is non-convex, has no unique minima, and, therefore, is not straightforwardly solvable. We can, however, separate this function into two *locally convex* V-shaped functions, which are straightforwardly solvable. By asserting that its least-squares solution is either less than $1/2$ (if $x \leq 1/2$), or greater than $1/2$ (if $x > 1/2$), we can address each case independently, and merge the outcome into a single piecewise proximal operator.

Case I In the first case, we address the operator for $x \leq 1/2$. For this, we start by simplifying our scalar proximal operator Eq. (11), by substituting $r(y)$ with its definition, and get

$$\text{prox}_{\kappa\lambda}^{\leq 1/2}(x) = \frac{1}{2}(y-x)^2 + \frac{\lambda'}{2}y^2 + \kappa|y| \; ,$$

for $\lambda'/2 = \lambda$. Then, we take its partial derivative for y

$$\frac{\partial}{\partial y}\,\text{prox}_{\kappa\lambda}^{\leq 1/2}(x) = (y-x) + \lambda' y + \kappa\,\text{sign}(y) \; ,$$

which we set to zero, obtaining

$$0 = (y-x) + \lambda' y + \kappa\,\text{sign}(y) \Leftrightarrow \; 0 = y(1+\lambda') - x + \kappa\,\text{sign}(y) \; .$$

By asserting that we can obtain a better least-squares solution if y has the same sign as x, we can substitute the sign of x with $\text{sign}(y)$, and get

$$0 = y(1+\lambda') - x + \kappa\,\text{sign}(x) \Leftrightarrow \; y = (1+\lambda')^{-1}[x - \kappa\,\text{sign}(x)] \; ,$$

which concludes the first case.

Case II Analogously, we now repeat the steps from above for $x > 1/2$. Again, we start by simplifying Eq. (11), substituting $r(y-1)$

$$\text{prox}_{\kappa\lambda}^{>1/2}(x) = \frac{1}{2}(y-x)^2 + \frac{\lambda'}{2}(y-1)^2 + \kappa|y-1| \; ,$$

for $\lambda'/2 = \lambda$. By taking its partial derivative for y

$$\frac{\partial}{\partial y}\,\text{prox}_{\kappa\lambda}^{>1/2}(x) = (y-x) + \lambda'(y-1) + \kappa\,\text{sign}(y-1) \; ,$$

and setting it to zero, we obtain

$$0 = (y-x) + \lambda'(y-1) + \kappa\,\text{sign}(y-1) \Leftrightarrow \; 0 = (1+\lambda')y - \lambda' - x + \kappa\,\text{sign}(y-1) \; .$$

Then, asserting that the least-squares solution does not get worse by using the same sign for $y-1$ and $x-1$, we can substitute the sign of $x-1$ with $\text{sign}(y-1)$, and get

$$0 = y(1+\lambda') - x - \lambda' + \kappa\,\text{sign}(x-1) \Leftrightarrow \; y = (1+\lambda')^{-1}[x - \kappa\,\text{sign}(x-1) + \lambda'] \; ,$$

which concludes the $x > 1/2$ case.

Combining Case I & Case II Combining the cases above yields our piecewise proximal operator (see Eq. (12))

$$\text{prox}_{\kappa\lambda}(x) \equiv (1+\lambda)^{-1} \begin{cases} x - \kappa\,\text{sign}(x) & \text{if } x \leq \frac{1}{2} \\ x - \kappa\,\text{sign}(x-1) + \lambda & \text{otherwise .} \end{cases} \tag{14}$$

Alternative Proximal Operator Considering Eq. (11), we notice that the term $y - x$ is squared, which means that there are multiple solutions to this equation. We derive the alternative operator analogously to the steps above, however, by switching the positions of y and x in f.

$$\text{prox}^{\text{alt.}}_{\kappa\lambda}(x) \equiv (\lambda-1)^{-1} \begin{cases} -x - \kappa\,\text{sign}(x) & \text{if } x \leq \frac{1}{2} \\ -x - \kappa\,\text{sign}(x-1) + \lambda & \text{otherwise .} \end{cases} \tag{15}$$

Since this operator is denominated by $\lambda' - 1$, we need to ensure that $\lambda' \neq 1$. Because our original proximal operator in Eq. (12) is denominated by $\lambda' + 1$, and since λ' is usually positive, we are not required to take extra precautions. Since this is more convenient, we select Eq. (12) as our proximal operator, rather than taking extra precautions when using Eq. (15).

Table 1: We provide an overview over the real-world datasets involved in this study, listing their dimensionalities, densities, and ranks k.

Dataset	Rank	Rows	Columns	Density
Genomes	28	2 504	226 623	0.1043
String	100	19 385	19 385	0.0318
GBM	100	650	10 701	0.0566
LGG	32	644	29 374	0.0729
Cerebellum	450	644	30 243	0.0823
TCGA	33	10 459	20 530	0.0501
Movielens 10M	20	71 567	65 133	0.0011
Netflix	20	17 770	480 189	0.0067
Patents	136	10 499	10 511	0.1305

B Reproducibility

In this addendum, we explain (**1**) how to obtain the datasets, (**2**) how we binarized them, (**3**) how to obtain the source code, and (**4**) how we parameterized the algorithms, starting with the dataset descriptions.

Broadly speaking, our datasets cover three domains: the *biomedical domain*, the *entertainment domain*, and the *innovation domain*. To cover the *biomedical domain*, we extract the network containing empirical evidence of protein-protein interactions in *homo sapiens* from the STRING database. For this, we remove all interactions in the protein-protein network, for which the empirical-evidence sub-score of the STRING database is zero, retaining only the protein-protein interactions discovered experimentally. From the GRAND repository, we take the gene regulatory networks sampled from *Glioblastoma (GBM)* and *Lower Grade Glioma (LGG)* brain cancer tissues, as well as from non-cancerous *Cerebellum* tissue. These networks all stem from a method, PANDA, which extracts gene-regulatory networks from data. The weights are between -20 and 20, where -20 corresponds to a low interaction likelihood, and 20 corresponds to a high interaction likelihood. To binarize, we set everything to zero except the weights whose Z-scores are in the upper 5% quantile, retaining the information about likely interactions. The *TCGA* dataset contains gene expressions from cancer patients. To binarize these logarithmic expression rates ($\log_{10}(x+1)$), we again only set those weights to one whose Z-scores lie in the upper 5% quantile [24], retaining high gene expressions. We further obtain the single nucleotide polymorphism (SNP) mutation data from the 1*k Genomes* project, and follow the data retrieval steps from the authors of BINAPS [10], which immediately produces a binary dataset. In the *entertainment domain*, we use the user-movie datasets *Movielens* and *Netflix*. Since we are only interested in recommending good movies, we binarize the original 5-star-scaled ratings, by setting reviews with more than 3.5 stars to one, and everything else to zero. Finally, as data from the *innovation domain*, we derive a directed citation network between patent groups from patent citation and classification data provided by *PatentsView*. We binarize this weighted network simply by setting every non-zero weight to one, retaining all edges in the network. For each of these publicly[3] available dataset, we give its dimensionality, density, and the matrix rank used in our experiments in Table 1.

[3] GRAND.NETWORKMEDICINE.ORG STRING-DB.ORG CANCER.GOV/TCGA INTERNATIONALGENOME.ORG PATENTSVIEW.ORG GROUPLENS.ORG/DATASETS/MOVIELENS KAGGLE.COM/DATASETS/NETFLIX-INC/NETFLIX-PRIZE-DATA

In our experiments in Sec. 4, we compare ELBMF against six methods: three dedicated BMF methods (ASSO [25], GRECOND [6], and PRIMP [16]), one streaming Bi-Clustering algorithm SOFA [30], one elastic-net-regularized NMF method leveraging proximal gradient descent (NMF [32, 22, 23]), and one interpretable Boolean autoencoder (BINAPS [10]). The code for ASSO, GRECOND, PRIMP, SOFA, and BINAPS was written by their respective authors and is publicly available. We implement NMF and ELBMF in the Julia programming language and provide their source code for reproducibility.[4] On *TCGA*, *Genomes*, *Movielens*, *Netflix*, and *Patents*, we set the l_2-regularizer $\lambda = 0.001$, the l_1-regularizer $\kappa = 0.005$, and the regularization rate to $\nu_t = 1.0033^t$. On *GBM*, *LGG*, and *Cerebellum*, we set the l_2-regularizer $\lambda = 0.001$, the l_1-regularizer $\kappa = 0.001$, and the regularization rate to $\nu_t = 1.0015^t$. We run NMF, ELBMF, PRIMP for at most 1 500 epochs on each dataset. In the case that ELBMF reaches its maximum number of iterations without convergence, we bridge the remaining integrality gap simply by applying our proximal operator (see Fig. 2). To obtain a good reconstruction for PRIMP, we use a grid-width of 0.01. To obtain a binary solution from NMF, we first clamp and then round its factor matrices upon convergence. We set ASSO's threshold, gain for covering, and penalty for over-covering each to 1. To achieve a better performance with ASSO, we parallelize ASSO on 16 CPU cores. Further, because ASSO's runtime scales with the number of columns, we reconstruct the *transposed* target whenever it has more columns than rows (see Table 1). For example, transposing *GBM*, *LGG*, *Cerebellum*, and *Genomes* is particularly beneficial for ASSO, as these datasets have orders-of-magnitude more columns than rows.

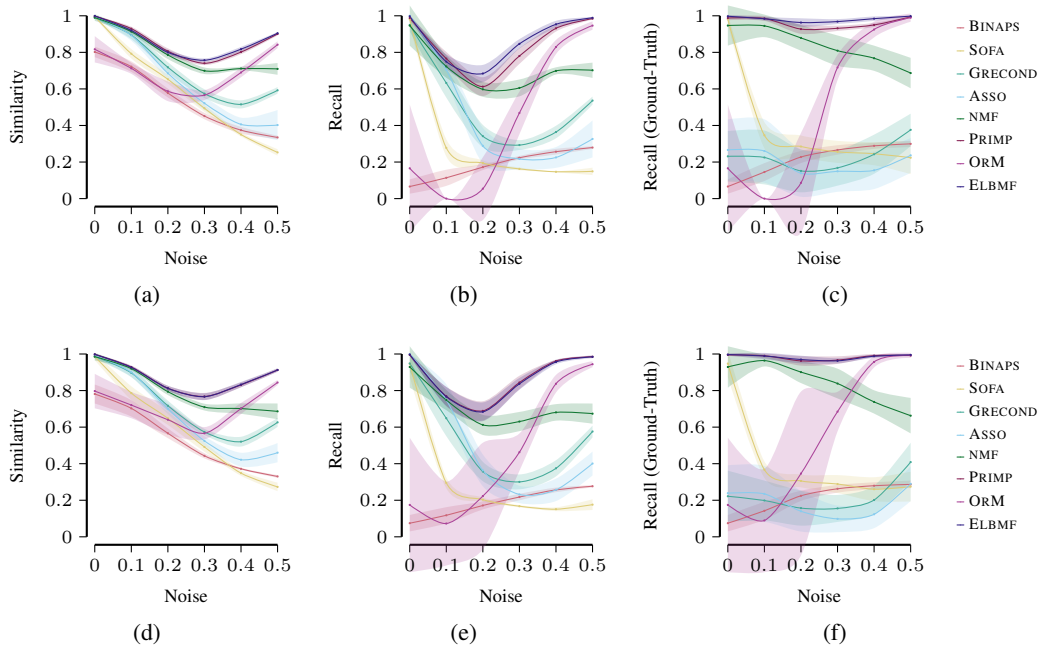


Figure 7: Overall, ELBMF reconstructs the noisy synthetic data well and recovers the ground-truth tiles which are *non-overlapping* in (a)-(c) and *overlapping* in (d)-(f). On synthetic data for additive noise levels increasing from 0% to 50%, we show *similarity* (a) and *recall* (b) w.r.t. the target matrix, and recall* w.r.t. the noise-free ground-truth tiles (c), for BINAPS, ASSO, GRECOND, ORM, SOFA, NMF, PRIMP, and ELBMF.

C Additional Results on the Performance of ELBMF on Synthetic Data

In this section, we provide additional results on our synthetic experiments in Sec. 4.

Supplementary to our synthetic experiments in Fig. 3, we report their standard deviations, depicted in Fig. 7 (a)-(c) as the shaded region around each line representing the mean. In Fig. 7a, we see that the standard deviation of the similarity score is low, except for ASSO and NMF in the highest noise regime, where they exhibit higher variance. In the other two figures, we observe that the standard deviation in Fig. 7b is lower than in Fig. 7c, for ASSO, GRECOND, NMF, and SOFA. Overall in Fig. 7, we see that ELBMF, PRIMP, and BINAPS perform most consistently.

[4] CS.UEF.FI/~PAULI/BASSO GITHUB.COM/MARTIN-TRNECKA/MATRIX-FACTORIZATION-ALGORITHMS BITBUCKET.ORG/NP84/PALTILING CS.UEF.FI/~PAULI/BMF/SOFA EDA.MMCI.UNI-SAARLAND.DE/PRJ/BINAPS dropbox.com/sh/gaoftnf0iyh27xq/AADRlhKxTV1S9LovwwnxVTCYa

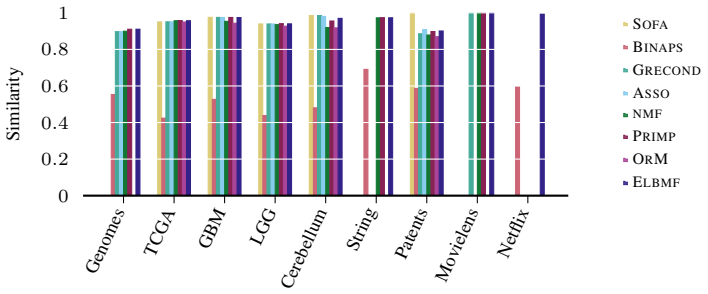


Figure 8: Our method factorizes real-world data with high similarity. We report similarity between real-world target matrices and their reconstructions from ASSO, GRECOND, ORM, SOFA, BINAPS, NMF, and ELBMF.

In our synthetic experiments in Sec. 4, we simulated data by generating non-overlapping tiles. In the experiments without tile overlap, we perform rejection sampling. That is, before we place the next randomly drawn tile into our matrix, we check for overlap with past placements. If we detect an overlap, we reject, redraw, and repeat, until we placed the desired number of tiles into our matrix. In the following we allow the overlap of tiles, depicting results on harder to separate data. For this, we do not check for overlap, thus inserting each draw into the matrix. To this end, we generate synthetic matrices as described in Sec. 4, however, this time, allowing tiles to overlap arbitrarily. In Fig. 7 we show similarity (d), recall (e), and recall*(f) for the overlapping case, observing a similar behavior to Fig. 7 and Fig. 3 across the board. Figures (a)-(f) show that ELBMF, which does not use any post-processing, achieves highly competitive results for *overlapping* and *non-overlapping* tiles, on par with the strongest competitor, which relies heavily on post-processing.

D Additional Results on the Performance of ELBMF on Real-World Data

In this section, we provide additional results on our real-world experiments in Sec. 4.

In particular, we want to answer the question whether ASSO, SOFA, NMF, PRIMP, BINAPS, and ELBMF are sparsity-inducing. For this, we additionally show similarity scores in Fig. 8. Since our data is sparse (see Table 1), we can expect that sparsity-inducing methods would result in a high similarity, even under a high loss. There, we observe that ASSO, GRECOND, ORM, SOFA, SNMF, PRIMP, and ELBMF yield reconstructions that are highly similar to the target, which means that they are sparsity-inducing methods. Although PRIMP exhibits a high loss (Fig. 5a) on, for example, *Cerebellum*, it also has a high similarity (Fig. 8). Hence, PRIMP still recovers a sparse and interpretable result. Since BINAPS covers many zeros with its patterns, it results in dense reconstructions of sparse targets, thus exhibiting lower similarity and high loss. That, however, does not mean that the patterns are necessarily wrong, but rather that they are in the wrong place.

E Extended Pseudo Code for ELBMF

Extending Alg. 1, we provide the long version of ELBMF as pseudo code in Alg. 2.

Algorithm 2: Long version of ELBMF in terms of iPALM [34]

Input:

Target Matrix	$\mathrm{A} \in \{0,1\}^{n \times m}$
Rank	$k \in \mathbb{N}$,
l_1 Regularizer Coefficients	$\kappa \in \mathbb{R}$,
l_2 Regularizer Coefficients	$\lambda \in \mathbb{R}$,
Regularization Rate	$\nu_t \in \mathbb{N} \to \mathbb{R}$,
optional Inertial Parameter	$\beta \in \mathbb{R}_+$

Output: Factors $\mathrm{U} \in \{0,1\}^{n \times k}$, $\mathrm{V} \in \{0,1\}^{k \times m}$

$\mathrm{U}_0 = \mathrm{U}_1 \leftarrow \mathrm{rand}(n, k)$
$\mathrm{V}_0 = \mathrm{V}_1 \leftarrow \mathrm{rand}(k, m)$

for $t = 1, 2, \ldots$ **until convergence do**
  $\lambda_t \leftarrow \lambda \cdot \nu_t$

  $\mathrm{U} \leftarrow \mathrm{U}_{t-1} + \beta(\mathrm{U}_{t-1} - \mathrm{U}_{t-2})$
  $\nabla_{\mathrm{U}} f = \mathrm{U}\mathrm{V}\mathrm{V}^\top - \mathrm{A}\mathrm{V}^\top$
  $L \leftarrow \|\mathrm{V}\mathrm{V}^\top\|_2$
  $\mathrm{U} \leftarrow \mathrm{prox}_{\kappa L^{-1}, \lambda_t L^{-1}}\left(\mathrm{U} - L^{-1}\nabla_{\mathrm{U}} f\right)$
  $\mathrm{U}_t \leftarrow \mathrm{U}$

  $\mathrm{V} \leftarrow \mathrm{V}_{t-1} + \beta(\mathrm{V}_{t-1} - \mathrm{V}_{t-2})$
  $\nabla_{\mathrm{V}} f = \mathrm{U}^\top\mathrm{U}\mathrm{V} - \mathrm{U}^\top\mathrm{A}$
  $L \leftarrow \|\mathrm{U}^\top\mathrm{U}\|_2$
  $\mathrm{V} \leftarrow \mathrm{prox}_{\kappa L^{-1}, \lambda_t L^{-1}}\left(\mathrm{V} - L^{-1}\nabla_{\mathrm{V}} f\right)$
  $\mathrm{V}_t \leftarrow \mathrm{V}$
end

if U or V not Boolean *(i.e., if the above was aborted early (cf. Fig. 2))*
  let $\lambda' \in \mathbb{R}$ be huge
  $\mathrm{U} \leftarrow \lfloor \mathrm{prox}_{0.5, \lambda'}(\mathrm{U}) \rceil$
  $\mathrm{V} \leftarrow \lfloor \mathrm{prox}_{0.5, \lambda'}(\mathrm{V}) \rceil$
end

return U, V