

Fast $(1 + \varepsilon)$ -Approximation Algorithms for Binary Matrix Factorization

Ameya Velingker* Maximilian Vötsch† David P. Woodruff‡ Samson Zhou§

June 6, 2023

Abstract

We introduce efficient $(1 + \varepsilon)$ -approximation algorithms for the binary matrix factorization (BMF) problem, where the inputs are a matrix $\mathbf{A} \in \{0, 1\}^{n \times d}$, a rank parameter $k > 0$, as well as an accuracy parameter $\varepsilon > 0$, and the goal is to approximate \mathbf{A} as a product of low-rank factors $\mathbf{U} \in \{0, 1\}^{n \times k}$ and $\mathbf{V} \in \{0, 1\}^{k \times d}$. Equivalently, we want to find \mathbf{U} and \mathbf{V} that minimize the Frobenius loss $\|\mathbf{UV} - \mathbf{A}\|_F^2$. Before this work, the state-of-the-art for this problem was the approximation algorithm of Kumar *et al.* [ICML 2019], which achieves a C -approximation for some constant $C \geq 576$. We give the first $(1 + \varepsilon)$ -approximation algorithm using running time singly exponential in k , where k is typically a small integer. Our techniques generalize to other common variants of the BMF problem, admitting bicriteria $(1 + \varepsilon)$ -approximation algorithms for L_p loss functions and the setting where matrix operations are performed in \mathbb{F}_2 . Our approach can be implemented in standard big data models, such as the streaming or distributed models.

1 Introduction

Low-rank approximation is a fundamental tool for factor analysis. The goal is to decompose several observed variables stored in the matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$ into a combination of k unobserved and uncorrelated variables called factors, represented by the matrices $\mathbf{U} \in \mathbb{R}^{n \times k}$ and $\mathbf{V} \in \mathbb{R}^{k \times d}$. In particular, we want to solve the problem

$$\min_{\mathbf{U} \in \mathbb{R}^{n \times k}, \mathbf{V} \in \mathbb{R}^{k \times d}} \|\mathbf{UV} - \mathbf{A}\|,$$

for some predetermined norm $\|\cdot\|$. Identifying the factors can often decrease the number of relevant features in an observation and thus significantly improve interpretability. Another benefit is that low-rank matrices allow us to approximate the matrix \mathbf{A} with its factors \mathbf{U} and \mathbf{V} using only $(n+d)k$

*Google Research. E-mail: ameyav@google.com.

†Faculty of Computer Science, Univie Doctoral School Computer Science DoCS, University of Vienna. Email: maximilian.voetsch@univie.ac.at.

‡Carnegie Mellon University. E-mail: dwoodruf@andrew.cmu.edu. Work done while at Google Research.

§UC Berkeley and Rice University. E-mail: samsonzhou@gmail.com.

M. Vötsch: This project has received funding from the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (Grant agreement No. 101019564 “The Design of Modern Fully Dynamic Data Structures (MoDynStruct)”).



parameters rather than the nd parameters needed to represent \mathbf{A} . Moreover, for a vector $\mathbf{x} \in \mathbb{R}^d$, we can approximate the matrix-vector multiplication $\mathbf{A}\mathbf{x} \approx \mathbf{U}\mathbf{V}\mathbf{x}$ in time $(n+d)k$, while computing $\mathbf{A}\mathbf{x}$ requires nd time. These benefits make low-rank approximation one of the most widely used tools in machine learning, recommender systems, data science, statistics, computer vision, and natural language processing. In many of these applications, discrete or categorical datasets are typical. In this case, restricting the underlying factors to a discrete domain for interpretability often makes sense. For example, [KPRW19] observed that nearly half of the data sets in the UCI repository [DG17] are categorical and thus can be represented as binary matrices, possibly using multiple binary variables to represent each category.

In the binary matrix factorization (BMF) problem, the input matrix $\mathbf{A} \in \{0, 1\}^{n \times d}$ is binary. Additionally, we are given an integer range parameter k , with $0 < k \ll n, d$. The goal is to approximate \mathbf{A} by the factors $\mathbf{U} \in \{0, 1\}^{n \times k}$ and $\mathbf{V} \in \{0, 1\}^{k \times d}$ such that $\mathbf{A} \approx \mathbf{U}\mathbf{V}$. The BMF problem restricts the general low-rank approximation problem to a discrete space, making finding good factors more challenging (see Section 1.3).

1.1 Our Contributions

We present $(1 + \varepsilon)$ -approximation algorithms for the binary low-rank matrix factorization problem for several standard loss functions used in the general low-rank approximation problem. Table 1 summarizes our results.

Reference	Approximation	Runtime	Other
[KPRW19]	$C \geq 576$	$2^{\tilde{O}(k^2)} \text{poly}(n, d)$	Frobenius loss
[FGL+20]	$1 + \varepsilon$	$2^{\frac{2^{\tilde{O}(k)}}{\varepsilon^2} \log^2 \frac{1}{\varepsilon}} \text{poly}(n, d)$	Frobenius loss
Our work	$1 + \varepsilon$	$2^{\tilde{O}(k^2/\varepsilon^4)} \text{poly}(n, d)$	Frobenius loss
[KPRW19]	$C \geq 122^{2p-2} + 2^{p-1}$	$2^{\text{poly}(k)} \text{poly}(n, d)$	L_p loss, $p \geq 1$
Here	$1 + \varepsilon$	$2^{\text{poly}(k/\varepsilon)} \text{poly}(n, d)$	L_p loss, $p \geq 1$, bicriteria
[FGL+20]	$1 + \varepsilon$	$2^{\frac{2^{\tilde{O}(k)}}{\varepsilon^2} \log^2 \frac{1}{\varepsilon}} \text{poly}(n, d)$	Binary field
[BBB+19]	$1 + \varepsilon$	$2^{\frac{2^{\tilde{O}(k)}}{\varepsilon^2} \log \frac{1}{\varepsilon}} \text{poly}(n, d)$	Binary field
[KPRW19]	$C \geq 40001$	$2^{\text{poly}(k)} \text{poly}(n, d)$	Binary field, bicriteria
Our work	$1 + \varepsilon$	$2^{\text{poly}(k/\varepsilon)} \text{poly}(n, d)$	Binary field, bicriteria

Table 1: Summary of related work on binary matrix factorization

Binary matrix factorization. We first consider the minimization of the Frobenius norm, defined by $\|\mathbf{A} - \mathbf{U}\mathbf{V}\|_F^2 = \sum_{i \in [n]} \sum_{j \in [d]} |\mathbf{A}_{i,j} - (\mathbf{U}\mathbf{V})_{i,j}|^2$, where $[n] := \{1, \dots, n\}$ and $\mathbf{A}_{i,j}$ denotes the entry in the i -th row and the j -th column of \mathbf{A} . Intuitively, we can view this as finding a least-squares approximation of \mathbf{A} .

We introduce the first $(1 + \varepsilon)$ -approximation algorithm for the BMF problem that runs in singly exponential time. That is, we present an algorithm that, for any $\varepsilon > 0$, returns $\mathbf{U}' \in \{0, 1\}^{n \times k}$, $\mathbf{V}' \in \{0, 1\}^{k \times d}$ with

$$\|\mathbf{A} - \mathbf{U}'\mathbf{V}'\|_F^2 \leq (1 + \varepsilon) \min_{\mathbf{U} \in \{0,1\}^{n \times k}, \mathbf{V} \in \{0,1\}^{k \times d}} \|\mathbf{A} - \mathbf{U}\mathbf{V}\|_F^2.$$

For $\varepsilon \in (0, 1)$, our algorithm uses $2^{\tilde{\mathcal{O}}(k^2/\varepsilon^4)}$ $\text{poly}(n, d)$ runtime and for $\varepsilon \geq 1$, our algorithm uses $2^{\tilde{\mathcal{O}}(k^2)}$ $\text{poly}(n, d)$ runtime, where $\text{poly}(n, d)$ denotes a polynomial in n and d .

By comparison, [KPRW19] gave a C -approximation algorithm for the BMF problem also using runtime $2^{\tilde{\mathcal{O}}(k^2)}$ $\text{poly}(n, d)$, but for some constant $C \geq 576$. Though they did not attempt to optimize for C , their proofs employ multiple triangle inequalities that present a constant lower bound of at least 2 on C . See Section 1.2 for a more thorough discussion of the limitations of their approach. [FGL⁺20] introduced a $(1 + \varepsilon)$ -approximation algorithm for the BMF problem with rank- k factors. However, their algorithm uses time doubly exponential in k , specifically $2^{\frac{2^{\mathcal{O}(k)}}{\varepsilon^2} \log^2 \frac{1}{\varepsilon}}$ $\text{poly}(n, d)$, which [BBB⁺19] later improved to doubly exponential runtime $2^{\frac{2^{\mathcal{O}(k)}}{\varepsilon^2} \log \frac{1}{\varepsilon}}$ $\text{poly}(n, d)$, while also showing that time $2^{k^{\Omega(1)}}$ is necessary even for constant-factor approximation, under the Small Set Expansion Hypothesis and the Exponential Time Hypothesis.

BMF with L_p loss. We also consider the more general problem of minimizing for L_p loss for a given p , defined as the optimization problem of minimizing $\|\mathbf{A} - \mathbf{UV}\|_p^p = \sum_{i \in [n]} \sum_{j \in d} |\mathbf{A}_{i,j} - (\mathbf{UV})_{i,j}|^p$. Of particular interest is the case $p = 1$, which corresponds to robust principal component analysis, and which has been proposed as an alternative to Frobenius norm low-rank approximation that is more robust to outliers, i.e., values that are far away from the majority of the data points [KK03, KK05, Kwa08, ZLS⁺12, BDB13, MKP14, SWZ17, PK18, BBB⁺19, MW21]. On the other hand, for $p > 2$, low-rank approximation with L_p error increasingly places higher priority on outliers, i.e., the larger entries of \mathbf{UV} .

We present the first $(1 + \varepsilon)$ -approximation algorithm for the BMF problem that runs in singly exponential time, albeit at the cost of incurring logarithmic increases in the rank k , making it a bicriteria algorithm. Specifically, for any $\varepsilon > 0$, our algorithm returns $\mathbf{U}' \in \{0, 1\}^{n \times k'}$, $\mathbf{V}' \in \{0, 1\}^{k' \times d}$ with

$$\|\mathbf{A} - \mathbf{U}'\mathbf{V}'\|_p^p \leq (1 + \varepsilon) \min_{\mathbf{U} \in \{0, 1\}^{n \times k}, \mathbf{V} \in \{0, 1\}^{k \times d}} \|\mathbf{A} - \mathbf{UV}\|_p^p,$$

where $k' = \mathcal{O}\left(\frac{k \log^2 n}{\varepsilon^2}\right)$. For $\varepsilon \in (0, 1)$, our algorithm uses $2^{\text{poly}(k/\varepsilon)}$ $\text{poly}(n, d)$ runtime and for $\varepsilon \geq 1$, our algorithm uses $2^{\text{poly}(k)}$ $\text{poly}(n, d)$ runtime.

Previous work [KPRW19] gave a C -approximation algorithm for this problem, using singly exponential runtime $2^{\text{poly}(k)}$ $\text{poly}(n, d)$, without incurring a bicriteria loss in the rank k . However, their constant $C \geq 122^{2p-2} + 2^{p-1}$ is large and depends on p . Again, their use of multiple triangle inequalities in their argument bars this approach from being able to achieve a $(1 + \varepsilon)$ -approximation. To our knowledge, no prior works achieved $(1 + \varepsilon)$ -approximation to BMF with L_p loss in singly exponential time.

BMF on binary fields. Finally, we consider the case where all arithmetic operations are performed modulo two, i.e., in the finite field \mathbb{F}_2 . Specifically, the (i, j) -th entry of \mathbf{UV} is the inner product $\langle \mathbf{U}_i, \mathbf{V}^{(j)} \rangle$ of the i -th row of \mathbf{U} and the j -th column of \mathbf{V} , taken over \mathbb{F}_2 . This model has been frequently used for dimensionality reduction for high-dimensional data with binary attributes [KG03, SJY09, JPHY14, DHJ⁺18] and independent component analysis, especially in the context of signal processing [Yer11, GGYT12, PRF15, PRF18]. This problem is also known as bipartite clique cover, the discrete basis problem, or minimal noise role mining and

has been well-studied in applications to association rule mining, database tiling, and topic modeling [SBM03, SH06, VAG07, MMG⁺08, BV10, LVAH12, CIK16, CSTZ22].

We introduce the first bicriteria $(1 + \varepsilon)$ -approximation algorithm for the BMF problem on binary fields that runs in singly exponential time. Specifically, for any $\varepsilon > 0$, our algorithm returns $\mathbf{U}' \in \{0, 1\}^{n \times k'}$, $\mathbf{V}' \in \{0, 1\}^{k' \times d}$ with

$$\|\mathbf{A} - \mathbf{U}'\mathbf{V}'\|_p^p \leq (1 + \varepsilon) \min_{\mathbf{U} \in \{0, 1\}^{n \times k}, \mathbf{V} \in \{0, 1\}^{k \times d}} \|\mathbf{A} - \mathbf{UV}\|_p^p,$$

where $k' = \mathcal{O}\left(\frac{k \log n}{\varepsilon}\right)$ and all arithmetic operations are performed in \mathbb{F}_2 . For $\varepsilon \in (0, 1)$, our algorithm has running time $2^{\text{poly}(k/\varepsilon)} \text{poly}(n, d)$ and for $\varepsilon \geq 1$, our algorithm has running time $2^{\text{poly}(k)} \text{poly}(n, d)$.

By comparison, [KPRW19] gave a bicriteria C -approximation algorithm for the BMF problem on binary fields with running time $2^{\text{poly}(k)} \text{poly}(n, d)$, for some constant $C \geq 40001$. Even though their algorithm also gives a bicriteria guarantee, their approach, once again, inherently cannot achieve $(1 + \varepsilon)$ -approximation. On the other hand, [FGL⁺20] achieved a $(1 + \varepsilon)$ -approximation without a bicriteria guarantee, but their algorithm uses doubly exponential running time $2^{\frac{2^{\mathcal{O}(k)}}{\varepsilon^2} \log^2 \frac{1}{\varepsilon}} \text{poly}(n, d)$, which [BBB⁺19] later improved to doubly exponential running time $2^{\frac{2^{\mathcal{O}(k)}}{\varepsilon^2} \log \frac{1}{\varepsilon}} \text{poly}(n, d)$, while also showing that running time doubly exponential in k is necessary for $(1 + \varepsilon)$ -approximation on \mathbb{F}_2 .

Applications to big data models. We remark that our algorithms are conducive to big data models. Specifically, our algorithmic ideas facilitate a two-pass algorithm in the streaming model, where either the rows or the columns of the input matrix arrive sequentially, and the goal is to perform binary low-rank approximation while using space sublinear in the size of the input matrix. Similarly, our approach can be used to achieve a two-round protocol in the distributed model, where either the rows or the columns of the input matrix are partitioned among several players, and the goal is to perform binary low-rank approximation while using total communication sublinear in the size of the input matrix. See Section 5 for a formal description of the problem settings and additional details.

1.2 Overview of Our Techniques

This section briefly overviews our approaches to achieving $(1 + \varepsilon)$ -approximation to the BMF problem. Alongside our techniques, we discuss why prior approaches for BMF fail to achieve $(1 + \varepsilon)$ -approximation.

The BMF problem under the Frobenius norm is stated as follows: Let $\mathbf{U}^* \in \{0, 1\}^{n \times k}$ and $\mathbf{V}^* \in \{0, 1\}^{k \times d}$ be optimal low-rank factors, so that

$$\|\mathbf{U}^*\mathbf{V}^* - \mathbf{A}\|_F^2 = \min_{\mathbf{U} \in \{0, 1\}^{n \times k}, \mathbf{V} \in \{0, 1\}^{k \times d}} \|\mathbf{UV} - \mathbf{A}\|_F^2. \quad (1)$$

Our approach relies on the sketch-and-solve paradigm, and we ask of our sketch matrix \mathbf{S} that it is an *affine embedding*, that is, given \mathbf{U}^* and \mathbf{A} , for all $\mathbf{V} \in \{0, 1\}^{k \times d}$,

$$(1 - \varepsilon)\|\mathbf{U}^*\mathbf{V} - \mathbf{A}\|_F^2 \leq \|\mathbf{SU}^*\mathbf{V} - \mathbf{SA}\|_F^2 \leq (1 + \varepsilon)\|\mathbf{U}^*\mathbf{V} - \mathbf{A}\|_F^2.$$

Observe that if \mathbf{S} is an affine embedding, then we obtain a $(1 + \varepsilon)$ -approximation by solving for the minimizer \mathbf{V}^* in the sketched space. That is, given \mathbf{S} and \mathbf{U}^* , instead of solving Equation 1 for \mathbf{V}^* , it suffices to solve

$$\operatorname{argmin}_{\mathbf{V} \in \{0,1\}^{k \times d}} \|\mathbf{S}\mathbf{U}^*\mathbf{V} - \mathbf{S}\mathbf{A}\|_F^2.$$

Guessing the sketch matrix \mathbf{S} . A general approach taken by [RSW16, KPRW19, BWZ19] for various low-rank approximation problems is first to choose \mathbf{S} in a way so that there are not too many possibilities for the matrices $\mathbf{S}\mathbf{U}^*$ and $\mathbf{S}\mathbf{A}$ and then find the minimizer \mathbf{V}^* for all guesses of $\mathbf{S}\mathbf{U}^*$ and $\mathbf{S}\mathbf{A}$. Note that this approach is delicate because it depends on the choice of the sketch matrix \mathbf{S} . For example, if we chose \mathbf{S} to be a dense matrix with random Gaussian entries, then since there are 2^{nk} possibilities for the matrix $\mathbf{U}^* \in \{0,1\}^{n \times k}$, we cannot enumerate the possible matrices $\mathbf{S}\mathbf{U}^*$. Prior work [RSW16, KPRW19, BWZ19] made the key observation that if \mathbf{A} (and thus \mathbf{U}^*) has a small number of unique rows, then a matrix \mathbf{S} that samples a small number of rows of \mathbf{A} has only a small number of possibilities for $\mathbf{S}\mathbf{A}$.

To ensure that \mathbf{A} has a small number of unique rows for the BMF problem, [KPRW19] first find a 2^k -means clustering solution $\tilde{\mathbf{A}}$ for the rows of \mathbf{A} . Instead of solving the problem on \mathbf{A} , they then solve BMF on the matrix $\tilde{\mathbf{A}}$, where each row is replaced by the center the point is assigned to, yielding at most 2^k unique rows. Finally, they note that $\|\mathbf{U}^*\mathbf{V}^* - \mathbf{A}\|_F^2$ is at least the 2^k -means cost, as $\mathbf{U}^*\mathbf{V}^*$ has at most 2^k unique rows. Now that $\tilde{\mathbf{A}}$ has 2^k unique rows, they can make all possible guesses for both $\mathbf{S}\mathbf{U}^*$ and $\mathbf{S}\tilde{\mathbf{A}}$ in time $2^{\tilde{O}(k^2)}$. By using an algorithm of [KMN⁺04] that achieves roughly a 9-approximation to k -means clustering, [KPRW19] ultimately obtain a C -approximation to the BMF problem, for some $C \geq 576$.

Shortcomings of previous work for $(1 + \varepsilon)$ -approximation. While [KPRW19] do not optimize for C , their approach fundamentally cannot achieve $(1 + \varepsilon)$ -approximation for BMF for the following reasons. First, they use a k -means clustering subroutine [KMN⁺04], (achieving roughly a 9-approximation) which due to hardness-of-approximation results [CK19, LSW17] can never achieve $(1 + \varepsilon)$ -approximation, as there cannot exist a 1.07-approximation algorithm for k -means clustering unless $P=NP$. Moreover, even if a $(1 + \varepsilon)$ -approximate k -means clustering could be found, there is no guarantee that the cluster centers obtained by this algorithm are binary. That is, while $\mathbf{U}\mathbf{V}$ has a specific form induced by the requirement that each factor must be binary, a solution to k -means clustering offers no such guarantee and may return Steiner points. Finally, [KPRW19] achieves a matrix \mathbf{S} that roughly preserves $\mathbf{S}\mathbf{U}^*$ and $\mathbf{S}\mathbf{A}$. By generalizations of the triangle inequality, one can show that $\|\mathbf{S}\mathbf{U}^*\mathbf{V}^* - \mathbf{S}\mathbf{A}\|_F^2$ preserves a constant factor approximation to $\|\mathbf{U}^*\mathbf{V}^* - \mathbf{A}\|_F^2$, but not necessarily a $(1 + \varepsilon)$ -approximation.

Another related work, [FGL⁺20], reduces instances of BMF to constrained k -means clustering instances, where the constraints demand that the selected centers are linear combinations of binary vectors. The core part of their work is to design a sampling-based algorithm for solving binary-constrained clustering instances, and the result on BMF is a corollary. Constrained clustering is a harder problem than BMF with Frobenius loss, so it is unclear how one might improve the doubly exponential running time using this approach.

Our approach: computing a strong coresets. We first reduce the number of unique rows in \mathbf{A} by computing a strong coresets $\tilde{\mathbf{A}}$ for \mathbf{A} . The strong coresets has the property that for any choices

of $\mathbf{U} \in \{0, 1\}^{n \times k}$ and $\mathbf{V} \in \{0, 1\}^{k \times d}$, there exists $\mathbf{X} \in \{0, 1\}^{n \times k}$ such that

$$(1 - \varepsilon) \|\mathbf{UV} - \mathbf{A}\|_F^2 \leq \|\mathbf{XV} - \tilde{\mathbf{A}}\|_F^2 \leq (1 + \varepsilon) \|\mathbf{UV} - \mathbf{A}\|_F^2.$$

Therefore, we instead first solve the low-rank approximation problem on $\tilde{\mathbf{A}}$ first. Crucially, we choose $\tilde{\mathbf{A}}$ to have $2^{\text{poly}(k/\varepsilon)}$ unique rows so then for a matrix \mathbf{S} that samples $\text{poly}(k/\varepsilon)$ rows, there are $2^{\text{poly}(k/\varepsilon)}$ possibilities for $\tilde{\mathbf{S}}\tilde{\mathbf{A}}$, so we can make all possible guesses for both \mathbf{SU}^* and $\tilde{\mathbf{S}}\tilde{\mathbf{A}}$. Unfortunately, we still have the problem that $\|\mathbf{SU}^*\mathbf{V}^* - \tilde{\mathbf{S}}\tilde{\mathbf{A}}\|_F^2$ does not even necessarily give a $(1 + \varepsilon)$ -approximation to $\|\mathbf{U}^*\mathbf{V}^* - \tilde{\mathbf{A}}\|_F^2$.

Binary matrix factorization. To that end, we show that when \mathbf{S} is a leverage score sampling matrix, then \mathbf{S} also satisfies an approximate matrix multiplication property. Therefore \mathbf{S} can effectively be used for an affine embedding. That is, the minimizer to $\|\mathbf{SU}^*\mathbf{V}^* - \tilde{\mathbf{S}}\tilde{\mathbf{A}}\|_F^2$ produces an $(1 + \varepsilon)$ -approximation to the cost of the optimal factors $\|\mathbf{U}^*\mathbf{V}^* - \tilde{\mathbf{A}}\|_F^2$. Thus, we can then solve

$$\begin{aligned} \mathbf{V}' &= \operatorname{argmin}_{\mathbf{V} \in \{0,1\}^{k \times d}} \|\mathbf{SU}^*\mathbf{V} - \tilde{\mathbf{S}}\tilde{\mathbf{A}}\|_F^2 \\ \mathbf{U}' &= \operatorname{argmin}_{\mathbf{U} \in \{0,1\}^{n \times k}} \|\mathbf{UV}' - \mathbf{A}\|_F^2, \end{aligned}$$

where the latter optimization problem can be solved by iteratively optimizing over each row so that the total computation time is $\mathcal{O}(2^k n)$ rather than 2^{kn} .

BMF on binary fields. We again form the matrix $\tilde{\mathbf{A}}$ by taking a strong coreset of \mathbf{A} , constructed using an algorithm that gives integer weight w_i to each point, and then duplicating the rows to form $\tilde{\mathbf{A}}$. That is, if the i -th row \mathbf{A}_i of \mathbf{A} is sampled with weight w_i in the coreset, then $\tilde{\mathbf{A}}$ will contain w_i repetitions of the row \mathbf{A}_i . We want to use the same approach for binary fields to make guesses for \mathbf{SU}^* and $\tilde{\mathbf{S}}\tilde{\mathbf{A}}$. However, it is no longer true that \mathbf{S} will provide an affine embedding over \mathbb{F}_2 , in part because the subspace embedding property of \mathbf{S} computes leverage scores of each row of \mathbf{U}^* and \mathbf{A} with respect to general integers. Hence, we require a different approach for matrix operations over \mathbb{F}_2 .

Instead, we group the rows of $\tilde{\mathbf{A}}$ by their number of repetitions, so that group \mathbf{G}_j consists of the rows of $\tilde{\mathbf{A}}$ that are repeated $[(1 + \varepsilon)^j, (1 + \varepsilon)^{j+1})$ times. That is, if \mathbf{A}_i appears w_i times in $\tilde{\mathbf{A}}$, then it appears a single time in group \mathbf{G}_j for $j = \lfloor \log w_i \rfloor$. We then perform entrywise L_0 low-rank approximation over \mathbb{F}_2 for each of the groups \mathbf{G}_j , which gives low-rank factors $\mathbf{U}^{(j)}$ and $\mathbf{V}^{(j)}$. We then compute $\widetilde{\mathbf{U}}^{(j)}$ by duplicating rows appropriately so that if \mathbf{A}_i is in \mathbf{G}_j , then we place the row of $\mathbf{U}^{(j)}$ corresponding to \mathbf{A}_i into the i -th row of $\widetilde{\mathbf{U}}^{(j)}$, for all $i \in [n]$. Otherwise if \mathbf{A}_i is not in \mathbf{G}_j , then we set i -th row of $\widetilde{\mathbf{U}}^{(j)}$ to be the all zeros row. We compute $\mathbf{V}^{(j)}$ by padding accordingly and then collect

$$\tilde{\mathbf{U}} = \left[\widetilde{\mathbf{U}}^{(0)} \mid \dots \mid \widetilde{\mathbf{U}}^{(\ell)} \right], \quad \tilde{\mathbf{V}} \leftarrow \widetilde{\mathbf{V}}^{(0)} \circ \dots \circ \widetilde{\mathbf{V}}^{(\ell)},$$

where $\left[\widetilde{\mathbf{U}}^{(0)} \mid \dots \mid \widetilde{\mathbf{U}}^{(\ell)} \right]$ denotes horizontal concatenation of matrices and $\widetilde{\mathbf{V}}^{(0)} \circ \dots \circ \widetilde{\mathbf{V}}^{(\ell)}$ denotes vertical concatenation (stacking) of matrices, to achieve bicriteria low-rank approximations $\tilde{\mathbf{U}}$ and $\tilde{\mathbf{V}}$ to $\tilde{\mathbf{A}}$. Finally, to achieve bicriteria factors \mathbf{U}' and \mathbf{V}' to \mathbf{A} , we ensure that \mathbf{U}' achieves the same block structure as $\tilde{\mathbf{U}}$.

BMF with L_p loss. We would again like to use the same approach as our $(1 + \varepsilon)$ -approximation algorithm for BMF with Frobenius loss. To that end, we observe that a coresets construction for clustering under L_p metrics rather than Euclidean distance is known, which we can use to construct $\tilde{\mathbf{A}}$. However, the challenge is that no known sampling matrix \mathbf{S} guarantees an affine embedding. One might hope that recent results on active L_p regression [CP19, PPP21, MMWY22, MMM+22, MMM+23] can provide such a tool. Unfortunately, adapting these techniques would still require taking a union bound over a number of columns, which would result in the sampling matrix having too many rows for our desired runtime.

Instead, we invoke the coresets construction on the rows and the columns so that $\tilde{\mathbf{A}}$ has a small number of distinct rows and columns. We again partition the rows of $\tilde{\mathbf{A}}$ into groups based on their frequency, but now we further partition the groups based on the frequency of the columns. Thus, it remains to solve BMF with L_p loss on the partition, each part of which has a small number of rows and columns. Since the contribution of each row toward the overall loss is small (because there is a small number of columns), we show that there exists a matrix that samples $\text{poly}(k/\varepsilon)$ rows of each partition that finally achieves the desired affine embedding. Therefore, we can solve the problem on each partition, pad the factors accordingly, and build the bicriteria factors as in the binary field case.

1.3 Motivation and Related Work

Low-rank approximation is one of the fundamental problems of machine learning and data science. Therefore, it has received extensive attention, e.g., see the surveys [KV09, Mah11, Woo14]. When the underlying loss function is the Frobenius norm, the low-rank approximation problem can be optimally solved via singular value decomposition (SVD). However, when we restrict both the observed input \mathbf{A} and the factors \mathbf{U}, \mathbf{V} to binary matrices, the SVD no longer guarantees optimal factors. In fact, many restricted variants of low-rank approximation are NP-hard [RSW16, SWZ17, KPRW19, BBB+19, BWZ19, FGL+20, MW21].

Motivation and background for BMF. The BMF problem has applications to graph partitioning [CIK16], low-density parity-check codes [RPG16], and optimizing passive organic LED (OLED) displays [KPRW19]. Observe that we can use \mathbf{A} to encode the incidence matrix of the bipartite graph with n vertices on the left side of the bipartition and d vertices on the right side so that $\mathbf{A}_{i,j} = 1$ if and only if there exists an edge connecting the i -th vertex on the left side with the j -th vertex on the right side. Then \mathbf{UV} can be written as the sum of k rank-1 matrices, each encoding a different bipartite clique of the graph, i.e., a subset of vertices on the left and a subset of vertices on the right such that there exists an edge between every vertex on the left and every vertex on the right. It then follows that the BMF problem solves the bipartite clique partition problem [Orl77, FMPS09, CHHK14, Neu18], in which the goal is to find the smallest integer k such that the graph can be represented as a union of k bipartite cliques.

[KPRW19] also present the following motivation for the BMF problem to improve the performance of passive OLED displays, which rapidly and sequentially illuminate rows of lights to render an image in a manner so that the human eye integrates this sequence of lights into a complete image. However, [KPRW19] observed that passive OLED displays could illuminate many rows simultaneously, provided the image being shown is a rank-1 matrix and that the apparent brightness of an image is inversely proportional to the rank of the decomposition. Thus [KPRW19] notes that BMF can be used to not only find a low-rank decomposition that illuminates pixels in a way that

seems brighter to the viewer but also achieves binary restrictions on the decomposition in order to use simple and inexpensive voltage drivers on the rows and columns, rather than a more expensive bank of video-rate digital to analog-to-digital converters.

BMF with Frobenius loss. [KPRW19] first gave a constant factor approximation algorithm for the BMF problem using runtime $2^{\tilde{O}(k^2)} \text{poly}(n, d)$, i.e., singly exponential time. [FGL⁺20] introduced a $(1 + \varepsilon)$ -approximation to the BMF problem with rank- k factors, but their algorithm uses doubly exponential time, specifically runtime $2^{\frac{2^{\mathcal{O}(k)}}{\varepsilon^2} \log^2 \frac{1}{\varepsilon}} \text{poly}(n, d)$, which was later improved to doubly exponential runtime $2^{\frac{2^{\mathcal{O}(k)}}{\varepsilon^2} \log \frac{1}{\varepsilon}} \text{poly}(n, d)$ by [BBB⁺19], who also showed that $2^{k^{\Omega(1)}}$ runtime is necessary even for constant-factor approximation, under the Small Set Expansion Hypothesis and the Exponential Time Hypothesis. By introducing sparsity constraints on the rows of \mathbf{U} and \mathbf{V} , [CSTZ22] provide an alternate parametrization of the runtime, though, at the cost of runtime quasipolynomial in n and d .

BMF on binary fields. Binary matrix factorization is particularly suited for datasets involving binary data. Thus, the problem is well-motivated for binary fields when performing dimensionality reduction on high-dimension datasets [KG03]. To this end, many heuristics have been developed for this problem [KG03, SJY09, FJS10, JPHY14], due to its NP-hardness [GV18, DHJ⁺18].

For the special case of $k = 1$, [SJY09] first gave a 2-approximation algorithm that uses polynomial time through a relaxation of integer linear programming. Subsequently, [JPHY14] produced a simpler approach, and [BKW17] introduced a sublinear time algorithm. For general k , [KPRW19] gave a constant factor approximation algorithm using runtime $2^{\text{poly}(k)} \text{poly}(n, d)$, i.e., singly exponential time, at the expense of a bicriteria solution, i.e., factors with rank $k' = \mathcal{O}(k \log n)$. [FGL⁺20] introduced a $(1 + \varepsilon)$ -approximation to the BMF problem with rank- k factors, but their algorithm uses doubly exponential time, specifically runtime $2^{\frac{2^{\mathcal{O}(k)}}{\varepsilon^2} \log^2 \frac{1}{\varepsilon}} \text{poly}(n, d)$, which was later improved to doubly exponential runtime $2^{\frac{2^{\mathcal{O}(k)}}{\varepsilon^2} \log \frac{1}{\varepsilon}} \text{poly}(n, d)$ by [BBB⁺19], who also showed that doubly exponential runtime is necessary for $(1 + \varepsilon)$ -approximation without bicriteria relaxation under the Exponential Time Hypothesis.

BMF with L_p loss. Using more general L_p loss functions can result in drastically different behaviors of the optimal low-rank factors for the BMF problem. For example, the low-rank factors for $p > 2$ are penalized more when the corresponding entries of \mathbf{UV} are large, and thus may choose to prioritize a larger number of small entries that do not match \mathbf{A} rather than a single large entry. On the other hand, $p = 1$ corresponds to robust principal component analysis, which yields factors that are more robust to outliers in the data [KK03, KK05, Kwa08, ZLS⁺12, BDB13, MKP14, SWZ17, PK18, BBB⁺19, MW21]. The first approximation algorithm with provable guarantees for L_1 low-rank approximation on the reals was given by [SWZ17]. They achieved $\text{poly}(k) \cdot \log d$ -approximation in roughly $\mathcal{O}(nd)$ time. For constant k , [SWZ17] further achieved constant-factor approximation in polynomial time.

When we restrict the inputs and factors to be binary, [KPRW19] observed that $p = 1$ corresponds to minimizing the number of edges in the symmetric difference between an unweighted bipartite graph G and its approximation H , which is the multiset union of k bicliques. Here we represent the graph G with n and d vertices on the bipartition's left- and right-hand side, respectively, through

its edge incidence matrix \mathbf{A} . Similarly, we have $\mathbf{U}_{i,j} = 1$ if and only if the i -th vertex on the left bipartition is in the j -th biclique and $\mathbf{V}_{i,j} = 1$ if and only if the j -th vertex on the right bipartition is in the i -th biclique. Then we have $\|\mathbf{UV} - \mathbf{A}\|_1 = |E(G) \triangle E(H)|$. [CIK16] showed how to solve the exact version of the problem, i.e., to recover \mathbf{U}, \mathbf{V} under the promise that $\mathbf{A} = \mathbf{UV}$, using $2^{\mathcal{O}(k^2)}$ poly(n, d) time. [KPRW19] recently gave the first constant-factor approximation algorithm for this problem, achieving a C -approximation using $2^{\text{poly}(k)}$ poly(n, d) time, for some constant $C \geq 122^{2p-2} + 2^{p-1}$.

1.4 Preliminaries

For an integer $n > 0$, we use $[n]$ to denote the set $\{1, 2, \dots, n\}$. We use poly(n) to represent a fixed polynomial in n and more generally, poly(n_1, \dots, n_k) to represent a fixed multivariate polynomial in n_1, \dots, n_k . For a function $f(n_1, \dots, n_k)$, we use $\tilde{\mathcal{O}}(f(n_1, \dots, n_k))$ to denote $f(n_1, \dots, n_k) \cdot \text{poly}(\log f(n_1, \dots, n_k))$.

We generally use bold-font variables to denote matrices. For a matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$, we use \mathbf{A}_i to denote the i -th row of \mathbf{A} and $\mathbf{A}^{(j)}$ to denote the j -th column of \mathbf{A} . We use $A_{i,j}$ to denote the entry in the i -th row and j -th column of \mathbf{A} . For $p \geq 1$, we write the entrywise L_p norm of \mathbf{A} as

$$\|\mathbf{A}\|_p = \left(\sum_{i \in [n]} \sum_{j \in [d]} A_{i,j}^p \right)^{1/p}.$$

The Frobenius norm of \mathbf{A} , denoted $\|\mathbf{A}\|_F$ is simply the entrywise L_2 norm of \mathbf{A} :

$$\|\mathbf{A}\|_F = \left(\sum_{i \in [n]} \sum_{j \in [d]} A_{i,j}^2 \right)^{1/2}.$$

The entrywise L_0 norm of \mathbf{A} is

$$\|\mathbf{A}\|_0 = |\{(i, j) \mid i \in [n], j \in [d] : A_{i,j} \neq 0\}|.$$

We use \circ to denote vertical stacking of matrices, so that

$$\mathbf{A}^{(1)} \circ \dots \circ \mathbf{A}^{(m)} = \begin{bmatrix} \mathbf{A}^{(1)} \\ \vdots \\ \mathbf{A}^{(m)} \end{bmatrix}.$$

For a set X of n points in \mathbb{R}^d weighted by a function w , the k -means clustering cost of X with respect to a set S of k centers is defined as

$$\text{Cost}(X, S, w) := \sum_{x \in X} w(x) \cdot \min_{s \in S} \|x - s\|_2^2.$$

When the weights w are uniformly unit across all points in X , we simply write $\text{Cost}(X, S) = \text{Cost}(X, S, w)$.

One of the core ingredients for avoiding the triangle inequality and achieving $(1+\varepsilon)$ -approximation is our use of coresets for k -means clustering:

Definition 1.1 (Strong coreset). *Given an accuracy parameter $\varepsilon > 0$ and a set X of n points in \mathbb{R}^d , we say that a subset C of X with weights w is a strong ε -coreset of X for the k -means clustering problem if for any set S of k points in \mathbb{R}^d , we have*

$$(1 - \varepsilon)\text{Cost}(X, S) \leq \text{Cost}(C, S, w) \leq (1 + \varepsilon)\text{Cost}(X, S).$$

Many coreset construction exist in the literature, and the goal is to minimize $|C|$, the size of the coreset, while preserving $(1 \pm \varepsilon)$ -approximate cost for all sets of k centers. If the points lie in \mathbb{R}^d , we can find coresets of size $\tilde{O}(\text{poly}(k, d, \varepsilon^{-1}))$, i.e., the size is independent of n .

Leverage scores. Finally, we recall the notion of a leverage score sampling matrix. For a matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$, the leverage score of row \mathbf{a}_i with $i \in [n]$ is defined as $\mathbf{a}_i(\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{a}_i^\top$. We can use the leverage scores to generate a random leverage score sampling matrix as follows:

Theorem 1.2 (Leverage score sampling matrix). *[DMM06a, DMM06b, Mag10, Woo14] Let $C > 1$ be a universal constant and $\alpha > 1$ be a parameter. Given a matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$, let ℓ_i be the leverage score of the i -th row of \mathbf{A} . Suppose $p_i \in \left[\min\left(1, \frac{C\ell_i \log k}{\varepsilon^2}\right), \min\left(1, \frac{C\alpha\ell_i \log k}{\varepsilon^2}\right) \right]$ for all $i \in [n]$.*

For $m := \mathcal{O}\left(\frac{\alpha}{\varepsilon^2} d \log d\right)$, let $\mathbf{S} \in \mathbb{R}^{m \times n}$ be generated so that each row of \mathbf{S} randomly selects row $j \in [n]$ with probability proportional to p_j and rescales the row by $\frac{1}{\sqrt{mp_j}}$. Then with probability at least 0.99, we have that simultaneously for all vectors $\mathbf{x} \in \mathbb{R}^d$,

$$(1 - \varepsilon)\|\mathbf{A}\mathbf{x}\|_2 \leq \|\mathbf{S}\mathbf{A}\mathbf{x}\|_2 \leq (1 + \varepsilon)\|\mathbf{A}\mathbf{x}\|_2.$$

The main point of [Theorem 1.2](#) is that given constant-factor approximations p_i to the leverage scores ℓ_i , it suffices to sample $\mathcal{O}(d \log d)$ rows of \mathbf{A} to achieve a constant-factor subspace embedding of \mathbf{A} , and similar bounds can be achieved for $(1 + \varepsilon)$ -approximate subspace embeddings. Finally, we remark that \mathbf{S} can be decomposed as the product of matrices $\mathbf{D}\mathbf{T}$, where $\mathbf{T} \in \mathbb{R}^{m \times n}$ is a sparse matrix with a single one per row, denoting the selection of a row for the purposes of leverage score sampling and \mathbf{D} is the diagonal matrix with the corresponding scaling factor, i.e., the i -th diagonal entry of \mathbf{D} is set to $\frac{1}{\sqrt{mp_j}}$ if the j -th row of \mathbf{A} is selected for the i -th sample.

2 Binary Low-Rank Approximation

In this section, we present a $(1 + \varepsilon)$ -approximation algorithm for binary low-rank approximation with Frobenius norm loss, where the goal is to find matrices $\mathbf{U} \in \{0, 1\}^{n \times k}$ and $\mathbf{V} \in \{0, 1\}^{k \times d}$ to minimize $\|\mathbf{U}\mathbf{V} - \mathbf{A}\|_F^2$. Suppose optimal low-rank factors are $\mathbf{U}^* \in \{0, 1\}^{n \times k}$ and $\mathbf{V}^* \in \{0, 1\}^{k \times d}$, so that

$$\|\mathbf{U}^*\mathbf{V}^* - \mathbf{A}\|_F^2 = \min_{\mathbf{U} \in \{0, 1\}^{n \times k}, \mathbf{V} \in \{0, 1\}^{k \times d}} \|\mathbf{U}\mathbf{V} - \mathbf{A}\|_F^2.$$

Observe that if we knew matrices $\mathbf{S}\mathbf{U}^*$ and $\mathbf{S}\mathbf{A}$ so that for all $\mathbf{V} \in \{0, 1\}^{k \times d}$,

$$(1 - \varepsilon)\|\mathbf{U}^*\mathbf{V} - \mathbf{A}\|_F^2 \leq \|\mathbf{S}\mathbf{U}^*\mathbf{V} - \mathbf{S}\mathbf{A}\|_F^2 \leq (1 + \varepsilon)\|\mathbf{U}^*\mathbf{V} - \mathbf{A}\|_F^2,$$

then we could find a $(1 + \varepsilon)$ -approximate solution for \mathbf{V}^* by solving the problem

$$\underset{\mathbf{V} \in \{0, 1\}^{k \times d}}{\text{argmin}} \|\mathbf{S}\mathbf{U}^*\mathbf{V} - \mathbf{S}\mathbf{A}\|_F^2$$

instead.

We would like to make guesses for the matrices $\mathbf{S}\mathbf{U}^*$ and $\mathbf{S}\mathbf{A}$, but first we must ensure there are not too many possibilities for these matrices. For example, if we chose \mathbf{S} to be a dense matrix with random gaussian entries, then $\mathbf{S}\mathbf{U}^*$ could have too many possibilities because without additional information, there are 2^{nk} possibilities for the matrix $\mathbf{U}^* \in \{0, 1\}^{n \times k}$. We can instead choose \mathbf{S} to be a leverage score sampling matrix, which samples rows from \mathbf{U}^* and \mathbf{A} . Since each row of \mathbf{U}^* has dimension k , then there are at most 2^k distinct possibilities for each of the rows of \mathbf{U}^* . On the other hand, $\mathbf{A} \in \{0, 1\}^{n \times d}$, so there may be 2^d distinct possibilities for the rows of \mathbf{A} , which is too many to guess.

Thus we first reduce the number of unique rows in \mathbf{A} by computing a strong coreset $\tilde{\mathbf{A}}$ for \mathbf{A} . The strong coreset has the property that for any choices of $\mathbf{U} \in \{0, 1\}^{n \times k}$ and $\mathbf{V} \in \{0, 1\}^{k \times d}$, there exists $\mathbf{X} \in \{0, 1\}^{n \times k}$ such that

$$(1 - \varepsilon)\|\mathbf{UV} - \mathbf{A}\|_F^2 \leq \|\mathbf{XV} - \tilde{\mathbf{A}}\|_F^2 \leq (1 + \varepsilon)\|\mathbf{UV} - \mathbf{A}\|_F^2.$$

Therefore, we instead first solve the low-rank approximation problem on $\tilde{\mathbf{A}}$ first. Crucially, $\tilde{\mathbf{A}}$ has $2^{\text{poly}(k/\varepsilon)}$ unique rows so then for a matrix \mathbf{S} that samples $\text{poly}(k/\varepsilon)$ rows, there are $\binom{2^{\text{poly}(k/\varepsilon)}}{\text{poly}(k/\varepsilon)} = 2^{\text{poly}(k/\varepsilon)}$ possible choices of $\mathbf{S}\tilde{\mathbf{A}}$, so we can enumerate all of them for both $\mathbf{S}\mathbf{U}^*$ and $\mathbf{S}\tilde{\mathbf{A}}$. We can then solve

$$\mathbf{V}' = \underset{\mathbf{V} \in \{0, 1\}^{k \times d}}{\text{argmin}} \|\mathbf{S}\mathbf{U}^*\mathbf{V} - \mathbf{S}\tilde{\mathbf{A}}\|_F^2$$

and

$$\mathbf{U}' = \underset{\mathbf{U} \in \{0, 1\}^{n \times k}}{\text{argmin}} \|\mathbf{UV}' - \mathbf{A}\|_F^2,$$

where the latter optimization problem can be solved by iteratively optimizing over each row, so that the total computation time is $\mathcal{O}(2^k n)$ rather than 2^{kn} . We give the full algorithm in [Algorithm 4](#) and the subroutine for optimizing with respect to $\tilde{\mathbf{A}}$ in [Algorithm 3](#). We give the subroutines for solving for \mathbf{V}' and \mathbf{U}' in [Algorithm 1](#) and [Algorithm 2](#), respectively.

Algorithm 1 Algorithm for computing optimal \mathbf{V} given \mathbf{U}

Input: $\tilde{\mathbf{A}} \in \{0, 1\}^{N \times d}$, $\mathbf{U} \in \{0, 1\}^{N \times k}$

Output: $\mathbf{V}' = \underset{\mathbf{V} \in \{0, 1\}^{k \times d}}{\text{argmin}} \|\mathbf{UV} - \tilde{\mathbf{A}}\|_F$

- 1: **for** $i = 1$ to $i = d$ **do** ▷Optimize for each column individually
 - 2: Set $\mathbf{V}'^{(i)} = \underset{\mathbf{V}^{(i)} \in \{0, 1\}^{k \times 1}}{\text{argmin}} \|\mathbf{UV}^{(i)} - \tilde{\mathbf{A}}^{(i)}\|_2$ ▷Enumerate over all 2^k possible binary vectors
 - 3: **return** $\mathbf{V}' = [\mathbf{V}'^{(1)} | \dots | \mathbf{V}'^{(d)}]$
-

First, we recall that leverage score sampling matrices preserve approximate matrix multiplication.

Lemma 2.1 (Lemma 32 in [\[CW13\]](#)). *Let $\mathbf{U} \in \mathbb{R}^{N \times k}$ have orthonormal columns, $\tilde{\mathbf{A}} \in \{0, 1\}^{N \times d}$, and $\mathbf{S} \in \mathbb{R}^{m \times N}$ be a leverage score sampling matrix for \mathbf{U} with $m = \mathcal{O}(\frac{1}{\varepsilon^2})$ rows. Then,*

$$\Pr \left[\|\mathbf{U}^\top \mathbf{S}^\top \tilde{\mathbf{S}}\tilde{\mathbf{A}} - \mathbf{U}^\top \tilde{\mathbf{A}}\|_F^2 < \varepsilon^2 \|\mathbf{U}\|_F^2 \|\tilde{\mathbf{A}}\|_F^2 \right] \geq 0.99.$$

Next, we recall that leverage score sampling matrices give subspace embeddings.

Algorithm 2 Algorithm for computing optimal \mathbf{U} given \mathbf{V}

Input: $\tilde{\mathbf{A}} \in \{0, 1\}^{N \times d}$, $\mathbf{V} \in \{0, 1\}^{k \times d}$ **Output:** $\mathbf{U}' = \operatorname{argmin}_{\mathbf{U} \in \{0, 1\}^{N \times k}} \|\mathbf{U}\mathbf{V} - \tilde{\mathbf{A}}\|_F$

- 1: **for** $i = 1$ to $i = N$ **do** ▷Optimize for each row individually
 - 2: Set $\mathbf{U}'_i = \operatorname{argmin}_{\mathbf{U}_i \in \{0, 1\}^{1 \times k}} \|\mathbf{U}_i \mathbf{V} - \tilde{\mathbf{A}}_i\|_2$ ▷Enumerate over all 2^k possible binary vectors
 - 3: **return** $\mathbf{U}' = \mathbf{U}'_1 \circ \dots \circ \mathbf{U}'_N$
-

Algorithm 3 Low-rank approximation for matrix $\tilde{\mathbf{A}}$ with t distinct rows

Input: $\tilde{\mathbf{A}} \in \{0, 1\}^{N \times d}$ with at most t distinct rows, rank parameter k , accuracy parameter $\varepsilon > 0$ **Output:** $\mathbf{U}' \in \{0, 1\}^{n \times k}$, $\mathbf{V}' \in \{0, 1\}^{k \times d}$ satisfying the property that $\|\mathbf{U}'\mathbf{V}' - \tilde{\mathbf{A}}\|_F^2 \leq (1 + \varepsilon) \min_{\mathbf{U} \in \{0, 1\}^{n \times k}, \mathbf{V} \in \{0, 1\}^{k \times d}} \|\mathbf{U}\mathbf{V} - \tilde{\mathbf{A}}\|_F^2$

- 1: $V \leftarrow \emptyset$
 - 2: **for** each guess of $\mathbf{S}\mathbf{U}^*$ and $\tilde{\mathbf{S}}\tilde{\mathbf{A}}$, where \mathbf{S} is a leverage score sampling matrix with $m = \mathcal{O}\left(\frac{k \log k}{\varepsilon^2}\right)$ rows with weights that are powers of two up to $\operatorname{poly}(N)$ **do**
 - 3: $V \leftarrow V \cup \operatorname{argmin}_{\mathbf{V} \in \{0, 1\}^{k \times d}} \|\mathbf{S}\mathbf{U}^*\mathbf{V} - \tilde{\mathbf{S}}\tilde{\mathbf{A}}\|_F^2$ ▷Algorithm 1
 - 4: **for** each $\mathbf{V} \in V$ **do**
 - 5: Let $\mathbf{U}_{\mathbf{V}} = \operatorname{argmin}_{\mathbf{U} \in \{0, 1\}^{N \times k}} \|\mathbf{U}\mathbf{V} - \tilde{\mathbf{A}}\|_F^2$ ▷Algorithm 2
 - 6: $\mathbf{V}' \leftarrow \operatorname{argmin}_{\mathbf{V} \in \{0, 1\}^{k \times d}} \|\mathbf{S}\mathbf{U}_{\mathbf{V}}\mathbf{V} - \tilde{\mathbf{S}}\tilde{\mathbf{A}}\|_F^2$
 - 7: $\mathbf{U}' \leftarrow \mathbf{U}_{\mathbf{V}'}$
 - 8: **return** $(\mathbf{U}', \mathbf{V}')$
-

Theorem 2.2 (Theorem 42 in [CW13]). For $\mathbf{U} \in \mathbb{R}^{N \times k}$, let $\mathbf{S} \in \mathbb{R}^{m \times N}$ be a leverage score sampling matrix for $\mathbf{U} \in \{0, 1\}^{N \times k}$ with $m = \mathcal{O}\left(\frac{k \log k}{\varepsilon^2}\right)$ rows. Then with probability at least 0.99, we have for all $\mathbf{V} \in \mathbb{R}^{k \times d}$,

$$(1 - \varepsilon)\|\mathbf{U}\mathbf{V}\|_F^2 \leq \|\mathbf{S}\mathbf{U}\mathbf{V}\|_F^2 \leq (1 + \varepsilon)\|\mathbf{U}\mathbf{V}\|_F^2.$$

Finally, we recall that approximate matrix multiplication and leverage score sampling suffices to achieve an affine embedding.

Theorem 2.3 (Theorem 39 in [CW13]). Let $\mathbf{U} \in \mathbb{R}^{N \times k}$ have orthonormal columns. Let \mathbf{S} be a sampling matrix that satisfies Lemma 2.1 with error parameter $\frac{\varepsilon}{\sqrt{k}}$ and also let $\tilde{\mathbf{S}}$ be a subspace embedding for \mathbf{U} with error parameter ε . Let $\mathbf{V}^* = \operatorname{argmin}_{\mathbf{V}} \|\mathbf{U}\mathbf{V} - \tilde{\mathbf{A}}\|_F$ and $\mathbf{X} = \mathbf{U}\mathbf{V}^* - \tilde{\mathbf{A}}$. Then for all $\mathbf{V} \in \mathbb{R}^{k \times d}$,

$$(1 - 2\varepsilon)\|\mathbf{U}\mathbf{V} - \tilde{\mathbf{A}}\|_F^2 - \|\mathbf{X}\|_F^2 \leq \|\mathbf{S}\mathbf{U}\mathbf{V} - \tilde{\mathbf{S}}\tilde{\mathbf{A}}\|_F^2 - \|\mathbf{S}\mathbf{X}\|_F^2 \leq (1 + 2\varepsilon)\|\mathbf{U}\mathbf{V} - \tilde{\mathbf{A}}\|_F^2 - \|\mathbf{X}\|_F^2.$$

We first show that Algorithm 3 achieves a good approximation to the optimal low-rank factors for the coresset $\tilde{\mathbf{A}}$.

Lemma 2.4. Suppose $\varepsilon < \frac{1}{10}$. Then with probability at least 0.97, the output of Algorithm 3 satisfies

$$\|\mathbf{U}'\mathbf{V}' - \tilde{\mathbf{A}}\|_F^2 \leq (1 + 6\varepsilon)\|\mathbf{U}^*\mathbf{V}^* - \tilde{\mathbf{A}}\|_F^2.$$

Proof. Let $\mathbf{V}'' = \operatorname{argmin}_{\mathbf{V} \in \{0,1\}^{k \times d}} \|\mathbf{S}\mathbf{U}^*\mathbf{V} - \tilde{\mathbf{A}}\|_F^2$ and let $\mathbf{U}'' = \operatorname{argmin}_{\mathbf{U} \in \{0,1\}^{N \times k}} \|\mathbf{S}\mathbf{U}\mathbf{V}'' - \tilde{\mathbf{A}}\|_F^2$. Since the algorithm chooses \mathbf{U}' and \mathbf{V}' over \mathbf{U}'' and \mathbf{V}'' , then

$$\|\mathbf{U}'\mathbf{V}' - \tilde{\mathbf{A}}\|_F^2 \leq \|\mathbf{U}''\mathbf{V}'' - \tilde{\mathbf{A}}\|_F^2.$$

Due to the optimality of \mathbf{U}'' ,

$$\|\mathbf{U}''\mathbf{V}'' - \tilde{\mathbf{A}}\|_F^2 \leq \|\mathbf{U}^*\mathbf{V}'' - \tilde{\mathbf{A}}\|_F^2.$$

Let $\mathbf{X} = \mathbf{U}^*\mathbf{V}^* - \tilde{\mathbf{A}}$. Note that since \mathbf{U}^* has orthonormal columns, then by [Lemma 2.1](#), the leverage score sampling matrix \mathbf{S} achieves approximate matrix multiplication with probability at least 0.99. By [Theorem 2.2](#), the matrix \mathbf{S} also is a subspace embedding for \mathbf{U} . Thus, \mathbf{S} meets the criteria for applying [Theorem 2.3](#). Then for the correct guess \mathbf{DT} of matrix \mathbf{S} corresponding to \mathbf{U}^* and conditioning on the correctness of \mathbf{S} in [Theorem 2.3](#),

$$\|\mathbf{U}^*\mathbf{V}'' - \tilde{\mathbf{A}}\|_F^2 \leq \frac{1}{1-2\varepsilon} [\|\mathbf{S}\mathbf{U}^*\mathbf{V}'' - \mathbf{S}\tilde{\mathbf{A}}\|_F^2 - \|\mathbf{S}\mathbf{X}\|_F^2 + \|\mathbf{X}\|_F^2].$$

Due to the optimality of \mathbf{V}'' ,

$$\frac{1}{1-2\varepsilon} [\|\mathbf{S}\mathbf{U}^*\mathbf{V}'' - \mathbf{S}\tilde{\mathbf{A}}\|_F^2 - \|\mathbf{S}\mathbf{X}\|_F^2 + \|\mathbf{X}\|_F^2] \leq \frac{1}{1-2\varepsilon} [\|\mathbf{S}\mathbf{U}^*\mathbf{V}^* - \mathbf{S}\tilde{\mathbf{A}}\|_F^2 - \|\mathbf{S}\mathbf{X}\|_F^2 + \|\mathbf{X}\|_F^2].$$

Then again conditioning on the correctness of \mathbf{S} ,

$$\begin{aligned} & \frac{1}{1-2\varepsilon} [\|\mathbf{S}\mathbf{U}^*\mathbf{V}^* - \mathbf{S}\tilde{\mathbf{A}}\|_F^2 - \|\mathbf{S}\mathbf{X}\|_F^2 + \|\mathbf{X}\|_F^2] \\ & \leq \frac{1}{1-2\varepsilon} [(1+2\varepsilon)\|\mathbf{U}^*\mathbf{V}^* - \tilde{\mathbf{A}}\|_F^2 + \|\mathbf{S}\mathbf{X}\|_F^2 - \|\mathbf{X}\|_F^2 - \|\mathbf{S}\mathbf{X}\|_F^2 + \|\mathbf{X}\|_F^2] \\ & \leq (1+6\varepsilon)\|\mathbf{U}^*\mathbf{V}^* - \tilde{\mathbf{A}}\|_F^2, \end{aligned}$$

for sufficiently small ε , e.g., $\varepsilon < \frac{1}{10}$. Thus, putting things together, we have that conditioned on the correctness of \mathbf{S} in [Theorem 2.3](#),

$$\|\mathbf{U}'\mathbf{V}' - \tilde{\mathbf{A}}\|_F^2 \leq (1+6\varepsilon)\|\mathbf{U}^*\mathbf{V}^* - \tilde{\mathbf{A}}\|_F^2.$$

Since the approximate matrix multiplication property of [Lemma 2.1](#), the subspace embedding property of [Theorem 2.2](#), and the affine embedding property of [Theorem 2.3](#) all fail with probability at most 0.01, then by a union bound, \mathbf{S} succeeds with probability at least 0.97. \square

We now analyze the runtime of the subroutine [Algorithm 3](#).

Lemma 2.5. *Algorithm 3 uses $2^{\mathcal{O}(m^2+m \log t)}$ poly(N, d) runtime for $m = \mathcal{O}\left(\frac{k \log k}{\varepsilon^2}\right)$.*

Proof. We analyze the number of possible guesses \mathbf{D} and \mathbf{T} corresponding to guesses of $\mathbf{S}\tilde{\mathbf{A}}$ (see the remark after [Theorem 1.2](#)). There are at most $\binom{t}{m} = 2^{\mathcal{O}(m \log t)}$ distinct subsets of $m = \mathcal{O}\left(\frac{k \log k}{\varepsilon^2}\right)$ rows of $\tilde{\mathbf{A}}$. Thus there are $2^{\mathcal{O}(m \log t)}$ possible matrices \mathbf{T} that selects m rows of $\tilde{\mathbf{A}}$, for the purposes of leverage score sampling. Assuming the leverage score sampling matrix does not sample any rows with leverage score less than $\frac{1}{\operatorname{poly}(N)}$, then there are $\mathcal{O}(\log N)^m = 2^{\mathcal{O}(m \log \log N)}$ total guesses for

the matrix \mathbf{D} . Note that $\log n \leq 2^m$ implies that $2^{\mathcal{O}(m \log \log N)} \leq 2^{\mathcal{O}(m^2)}$ while $\log N > 2^m$ implies that $2^{\mathcal{O}(m \log \log N)} \leq 2^{\mathcal{O}(\log^2 \log N)} \leq N$. Therefore, there are at most $2^{\mathcal{O}(m^2 + m \log t)} N$ total guesses for all combinations of \mathbf{T} and \mathbf{D} , corresponding to all guesses of $\widetilde{\mathbf{S}}\mathbf{A}$.

For each guess of \mathbf{S} and $\widetilde{\mathbf{S}}\mathbf{A}$, we also need to guess $\mathbf{S}\mathbf{U}^*$. Since $\mathbf{U}^* \in \{0, 1\}^{N \times k}$ is binary and \mathbf{T} samples m rows before weighting each row with one of $\mathcal{O}(\log N)$ possible weights, the number of total guesses for $\mathbf{S}\mathbf{U}^*$ is $(2 \cdot \mathcal{O}(\log N))^{mk}$.

Given guesses for $\mathbf{S}\mathbf{A}$ and $\mathbf{S}\mathbf{U}^*$, we can then compute $\operatorname{argmin}_{\mathbf{V} \in \{0, 1\}^{k \times d}} \|\mathbf{S}\mathbf{U}^*\mathbf{V} - \mathbf{S}\mathbf{A}\|_F^2$ using $\mathcal{O}(2^{kd})$ time through the subroutine [Algorithm 1](#), which enumerates through all possible 2^k binary vectors for each column. For a fixed \mathbf{V} , we can then compute $\mathbf{U}\mathbf{V} = \operatorname{argmin}_{\mathbf{U} \in \{0, 1\}^{N \times k}} \|\mathbf{U}\mathbf{V} - \mathbf{A}\|_F^2$ using $\mathcal{O}(2^k N)$ time through the subroutine [Algorithm 2](#), which enumerates through all possible 2^k binary vectors for each row of $\mathbf{U}\mathbf{V}$. Therefore, the total runtime of [Algorithm 3](#) is $2^{\mathcal{O}(m^2 + m \log t)} \operatorname{poly}(N, d)$. \square

We recall the following construction for a strong ε -coreset for k -means clustering.

Theorem 2.6 (Theorem 36 in [\[FSS20\]](#)). *Let $X \subset \mathbb{R}^d$ be a subset of n points, $\varepsilon \in (0, 1)$ be an accuracy parameter, and let $t = \mathcal{O}\left(\frac{k^3 \log^2 k}{\varepsilon^4}\right)$. There exists an algorithm that uses $\mathcal{O}\left(nd^2 + n^2 d + \frac{nk d}{\varepsilon^2} + \frac{nk^2}{\varepsilon^2}\right)$ time and outputs a set of t weighted points that is a strong ε -coreset for k -means clustering with probability at least 0.99. Moreover, each point has an integer weight that is at most $\operatorname{poly}(n)$.*

Algorithm 4 Low-rank approximation for matrix \mathbf{A}

Input: $\mathbf{A} \in \{0, 1\}^{n \times d}$, rank parameter k , accuracy parameter $\varepsilon > 0$

Output: $\mathbf{U}' \in \{0, 1\}^{n \times k}$, $\mathbf{V}' \in \{0, 1\}^{k \times d}$ satisfying the property that $\|\mathbf{U}'\mathbf{V}' - \mathbf{A}\|_F^2 \leq (1 + \varepsilon) \min_{\mathbf{U} \in \{0, 1\}^{n \times k}, \mathbf{V} \in \{0, 1\}^{k \times d}} \|\mathbf{U}\mathbf{V} - \mathbf{A}\|_F^2$

- 1: $t \leftarrow \mathcal{O}\left(\frac{2^{3k} k^2}{\varepsilon^4}\right)$ \triangleright [Theorem 2.6](#) for 2^k -means clustering
 - 2: Compute a strong coreset C for 2^k -means clustering of \mathbf{A} , with size t and total weight $N = \operatorname{poly}(n)$
 - 3: Let $\widetilde{\mathbf{A}} \in \{0, 1\}^{N \times d}$ be the matrix representation of C , where weighted points are duplicated appropriately
 - 4: Let $(\widetilde{\mathbf{U}}, \widetilde{\mathbf{V}})$ be the output of [Algorithm 3](#) on input $\widetilde{\mathbf{A}}$
 - 5: $\mathbf{U}' \leftarrow \operatorname{argmin}_{\mathbf{U} \in \{0, 1\}^{n \times k}} \|\mathbf{U}\widetilde{\mathbf{V}} - \mathbf{A}\|_F^2$, $\mathbf{V}' \leftarrow \widetilde{\mathbf{V}}$ \triangleright [Algorithm 2](#)
 - 6: **return** $(\mathbf{U}', \mathbf{V}')$
-

We now justify the correctness of [Algorithm 4](#).

Lemma 2.7. *With probability at least 0.95, [Algorithm 4](#) returns \mathbf{U}', \mathbf{V}' such that*

$$\|\mathbf{U}'\mathbf{V}' - \mathbf{A}\|_F^2 \leq (1 + \varepsilon) \min_{\mathbf{U} \in \{0, 1\}^{n \times k}, \mathbf{V} \in \{0, 1\}^{k \times d}} \|\mathbf{U}\mathbf{V} - \mathbf{A}\|_F^2.$$

Proof. Let $\widetilde{\mathbf{M}}$ be the indicator matrix that selects a row of $\widetilde{\mathbf{U}}\widetilde{\mathbf{V}} = \widetilde{\mathbf{U}}\mathbf{V}'$ to match to each row of \mathbf{A} , so that by the optimality of \mathbf{U}' ,

$$\|\mathbf{U}'\mathbf{V}' - \mathbf{A}\|_F^2 \leq \|\widetilde{\mathbf{M}}\widetilde{\mathbf{U}}\widetilde{\mathbf{V}} - \mathbf{A}\|_F^2.$$

Note that any \mathbf{V} is a set of k points in $\{0, 1\}^d$ and so each row \mathbf{U}_i of \mathbf{U} induces one of at most 2^k possible points $\mathbf{U}_i \mathbf{V} \in \{0, 1\}^d$. Hence $\|\mathbf{U}\mathbf{V} - \mathbf{A}\|_F^2$ is the objective value of a constrained 2^k -means clustering problem. Thus by the choice of t in [Theorem 2.6](#), we have that $\tilde{\mathbf{A}}$ is a strong coresnet, so that

$$\|\tilde{\mathbf{M}}\tilde{\mathbf{U}}\tilde{\mathbf{V}} - \mathbf{A}\|_F^2 \leq (1 + \varepsilon)\|\tilde{\mathbf{U}}\tilde{\mathbf{V}} - \tilde{\mathbf{A}}\|_F^2.$$

Let $\mathbf{U}^* \in \{0, 1\}^{n \times k}$ and $\mathbf{V}^* \in \{0, 1\}^{k \times d}$ such that

$$\|\mathbf{U}^* \mathbf{V}^* - \mathbf{A}\|_F^2 = \min_{\mathbf{U} \in \{0, 1\}^{n \times k}, \mathbf{V} \in \{0, 1\}^{k \times d}} \|\mathbf{U}\mathbf{V} - \mathbf{A}\|_F^2.$$

Let \mathbf{M}^* be the indicator matrix that selects a row of $\mathbf{U}^* \mathbf{V}^*$ to match to each row of $\tilde{\mathbf{A}}$, so that by [Lemma 2.4](#),

$$(1 + \varepsilon)\|\tilde{\mathbf{U}}\tilde{\mathbf{V}} - \tilde{\mathbf{A}}\|_F^2 \leq (1 + \varepsilon)^2 \|\mathbf{M}^* \mathbf{U}^* \mathbf{V}^* - \tilde{\mathbf{A}}\|_F^2.$$

Then by the choice of t in [Theorem 2.6](#), we have that

$$(1 + \varepsilon)^2 \|\mathbf{M}^* \mathbf{U}^* \mathbf{V}^* - \tilde{\mathbf{A}}\|_F^2 \leq (1 + \varepsilon)^3 \|\mathbf{U}^* \mathbf{V}^* - \mathbf{A}\|_F^2.$$

The desired claim then follows from rescaling ε . \square

We now analyze the runtime of [Algorithm 4](#).

Lemma 2.8. *Algorithm 4 uses $2^{\tilde{\mathcal{O}}(k^2/\varepsilon^4)}$ poly(n, d) runtime.*

Proof. By [Theorem 2.6](#), it follows that [Algorithm 4](#) uses $\mathcal{O}\left(nd^2 + n^2d + \frac{nk d}{\varepsilon^2} + \frac{nk^2}{\varepsilon^2}\right)$ time to compute $\tilde{\mathbf{A}} \in \{0, 1\}^{N \times d}$ with $N = \text{poly}(n)$. By [Lemma 2.5](#), it follows that [Algorithm 3](#) on input $\tilde{\mathbf{A}}$ thus uses runtime $2^{\mathcal{O}(m^2 + m \log t)}$ poly(N, d) for $m = \mathcal{O}\left(\frac{k \log k}{\varepsilon^2}\right)$ and $t = \mathcal{O}\left(\frac{2^{3k} k^2}{\varepsilon^4}\right)$. Finally, computing \mathbf{U}' via [Algorithm 2](#) takes $\mathcal{O}(2^k n)$ time after enumerating through all possible 2^k binary vectors for each row of \mathbf{U}' . Therefore, the total runtime of [Algorithm 4](#) is $2^{\tilde{\mathcal{O}}\left(\frac{k^2 \log^2 k}{\varepsilon^4}\right)} \text{poly}(n, d) = 2^{\tilde{\mathcal{O}}(k^2/\varepsilon^4)} \text{poly}(n, d)$. \square

Combining [Lemma 2.7](#) and [Lemma 2.8](#), we have:

Theorem 2.9. *There exists an algorithm that uses $2^{\tilde{\mathcal{O}}(k^2/\varepsilon^4)}$ poly(n, d) runtime and with probability at least $\frac{2}{3}$, outputs $\mathbf{U}' \in \{0, 1\}^{n \times k}$ and $\mathbf{V}' \in \{0, 1\}^{k \times d}$ such that*

$$\|\mathbf{U}' \mathbf{V}' - \mathbf{A}\|_F^2 \leq (1 + \varepsilon) \min_{\mathbf{U} \in \{0, 1\}^{n \times k}, \mathbf{V} \in \{0, 1\}^{k \times d}} \|\mathbf{U}\mathbf{V} - \mathbf{A}\|_F^2.$$

3 \mathbb{F}_2 Low-Rank Approximation

In this section, we present a $(1 + \varepsilon)$ -approximation algorithm for binary low-rank approximation on \mathbb{F}_2 , where the goal is to find matrices $\mathbf{U} \in \{0, 1\}^{n \times k}$ and $\mathbf{V} \in \{0, 1\}^{k \times d}$ to minimize the Frobenius norm loss $\|\mathbf{U}\mathbf{V} - \mathbf{A}\|_F^2$, but now all operations are performed in \mathbb{F}_2 . We would like to use the same approach as in [Section 2](#), i.e., to make guesses for the matrices $\mathbf{S}\mathbf{U}^*$ and $\mathbf{S}\mathbf{A}$ while ensuring there are not too many possibilities for these matrices. To do so for matrix operations over general integers, we chose \mathbf{S} to be a leverage score sampling matrix that samples rows from \mathbf{U}^* and \mathbf{A} . We then

used the approximate matrix multiplication property in [Lemma 2.1](#) and the subspace embedding property in [Theorem 2.2](#) to show that \mathbf{S} provides an affine embedding in [Theorem 2.3](#) over general integers. However, it no longer necessarily seems true that \mathbf{S} will provide an affine embedding over \mathbb{F}_2 , in part because the subspace embedding property of \mathbf{S} computes leverage scores of each row of \mathbf{U}^* and \mathbf{A} with respect to general integers. Thus we require an alternate approach for matrix operations over \mathbb{F}_2 .

Instead, we form the matrix $\tilde{\mathbf{A}}$ by taking a strong coreset of \mathbf{A} and then duplicating the rows according to their weight w_i to form $\tilde{\mathbf{A}}$. That is, if the i -th row \mathbf{A}_i of \mathbf{A} is sampled with weight w_i in the coreset, then $\tilde{\mathbf{A}}$ will contain w_i repetitions of the row \mathbf{A}_i , where we note that w_i is an integer. We then group the rows of $\tilde{\mathbf{A}}$ by their repetitions, so that group \mathbf{G}_j consists of the rows of $\tilde{\mathbf{A}}$ that are repeated $[(1 + \varepsilon)^j, (1 + \varepsilon)^{j+1})$ times. Thus if \mathbf{A}_i appears w_i times in $\tilde{\mathbf{A}}$, then it appears a single time in group \mathbf{G}_j for $j = \lfloor \log w_i \rfloor$.

We perform entrywise L_0 low-rank approximation over \mathbb{F}_2 for each of the groups \mathbf{G}_j , which gives low-rank factors $\mathbf{U}^{(j)}$ and $\mathbf{V}^{(j)}$. We then compute $\widetilde{\mathbf{U}}^{(j)} \in \mathbb{R}^{n \times d}$ from $\mathbf{U}^{(j)}$ by following procedure. If \mathbf{A}_i is in \mathbf{G}_j , then we place the row of $\mathbf{U}^{(j)}$ corresponding to \mathbf{A}_i into the i -th row of $\widetilde{\mathbf{U}}^{(j)}$, for all $i \in [n]$. Note that the row of $\mathbf{U}^{(j)}$ corresponding to \mathbf{A}_i may not be the i -th row of $\mathbf{U}^{(j)}$, e.g., since \mathbf{A}_i will appear only once in \mathbf{G}_j even though it appears $w_i \in [(1 + \varepsilon)^j, (1 + \varepsilon)^{j+1})$ times in $\tilde{\mathbf{A}}$. Otherwise if \mathbf{A}_i is not in \mathbf{G}_j , then we set i -th row of $\widetilde{\mathbf{U}}^{(j)}$ to be the all zeros row. We then achieve $\widetilde{\mathbf{V}}^{(j)}$ by padding accordingly. Finally, we collect

$$\tilde{\mathbf{U}} = \left[\widetilde{\mathbf{U}}^{(0)} \mid \dots \mid \widetilde{\mathbf{U}}^{(\ell)} \right], \quad \tilde{\mathbf{V}} \leftarrow \widetilde{\mathbf{V}}^{(0)} \circ \dots \circ \widetilde{\mathbf{V}}^{(\ell)}$$

to achieve bicriteria low-rank approximations $\tilde{\mathbf{U}}$ and $\tilde{\mathbf{V}}$ to $\tilde{\mathbf{A}}$. Finally, to achieve bicriteria low-rank approximations \mathbf{U}' and \mathbf{V}' to \mathbf{A} , we require that \mathbf{U}' achieves the same block structure as $\tilde{\mathbf{U}}$. We describe this subroutine in [Algorithm 5](#) and we give the full low-rank approximation bicriteria algorithm in [Algorithm 6](#).

We first recall the following subroutine to achieve entrywise L_0 low-rank approximation over \mathbb{F}_2 . Note that for matrix operations over \mathbb{F}_2 , we have that the entrywise L_0 norm is the same as the entrywise L_p norm for all p .

Lemma 3.1 (Theorem 3 in [\[BBB⁺19\]](#)). *For $\varepsilon \in (0, 1)$, there exists a $(1 + \varepsilon)$ -approximation algorithm to entrywise L_0 rank- k approximation over \mathbb{F}_2 running in $d \cdot n^{\text{poly}(k/\varepsilon)}$ time.*

Algorithm 5 Algorithm for computing optimal \mathbf{U} given $\mathbf{V}^{(1)}, \dots, \mathbf{V}^{(\ell)}$

Input: $\tilde{\mathbf{A}} \in \{0, 1\}^{N \times d}$, $\mathbf{V}^{(1)}, \dots, \mathbf{V}^{(\ell)} \in \{0, 1\}^{k \times d}$

Output: $\mathbf{U}' = \operatorname{argmin}_{\mathbf{U} \in \{0, 1\}^{N \times \ell k}} \|\mathbf{U}\mathbf{V} - \tilde{\mathbf{A}}\|_F$, where \mathbf{U} is restricted to one nonzero block of k coordinates

- 1: **for** $i = 1$ to $i = N$ **do**
 - 2: Set $(\mathbf{U}'_i, j') = \operatorname{argmin}_{\mathbf{U}_i \in \{0, 1\}^{1 \times k}, j \in [\ell]} \|\mathbf{U}_i \mathbf{V}^{(j)} - \tilde{\mathbf{A}}_i\|_2$ ▷Enumerate over all 2^k possible binary vectors, all ℓ indices
 - 3: Pad \mathbf{U}'_i with length ℓk , as the j' -th block of k coordinates
 - 4: **return** $\mathbf{U}' = \mathbf{U}'_1 \circ \dots \circ \mathbf{U}'_N$
-

We first justify the correctness of [Algorithm 6](#).

Algorithm 6 Bicriteria low-rank approximation on \mathbb{F}_2 for matrix \mathbf{A}

Input: $\mathbf{A} \in \{0, 1\}^{n \times d}$, rank parameter k , accuracy parameter $\varepsilon > 0$

Output: $\mathbf{U}' \in \{0, 1\}^{n \times k}$, $\mathbf{V}' \in \{0, 1\}^{k \times d}$ satisfying the property that $\|\mathbf{U}'\mathbf{V}' - \mathbf{A}\|_F^2 \leq (1 + \varepsilon) \min_{\mathbf{U} \in \{0, 1\}^{n \times k}, \mathbf{V} \in \{0, 1\}^{k \times d}} \|\mathbf{U}\mathbf{V} - \mathbf{A}\|_F^2$, where all matrix operations are performed in \mathbb{F}_2

- 1: $\ell \leftarrow \mathcal{O}\left(\frac{\log n}{\varepsilon}\right)$, $t \leftarrow \mathcal{O}\left(\frac{(2^k \ell)^3 k^2}{\varepsilon^4}\right)$, $k' \leftarrow \ell k$ ▷ [Theorem 2.6](#) for 2^k -means clustering
 - 2: Compute a strong coreset C for 2^k -means clustering of \mathbf{A} , with size t and total weight $N = \text{poly}(n)$
 - 3: Let $\tilde{\mathbf{A}} \in \{0, 1\}^{N \times d}$ be the matrix representation of C , where weighted points are duplicated appropriately
 - 4: For $i \in [\ell]$, let $\mathbf{G}^{(i)}$ be the group of rows (removing multiplicity) of $\tilde{\mathbf{A}}$ with frequency $[(1 + \varepsilon)^i, (1 + \varepsilon)^{i+1})$
 - 5: Let $(\widetilde{\mathbf{U}}^{(i)}, \widetilde{\mathbf{V}}^{(i)})$ be the output of [Lemma 3.1](#) on input $\mathbf{G}^{(i)}$, padded to $\mathbb{R}^{n \times k}$ and $\mathbb{R}^{k \times d}$, respectively
 - 6: $\widetilde{\mathbf{V}} \leftarrow \widetilde{\mathbf{V}}^{(0)} \circ \dots \circ \widetilde{\mathbf{V}}^{(\ell)}$
 - 7: Use [Algorithm 5](#) with $\widetilde{\mathbf{V}}^{(0)}, \dots, \widetilde{\mathbf{V}}^{(\ell)}$ and \mathbf{A} to find \mathbf{U}'
 - 8: **return** $(\mathbf{U}', \mathbf{V}')$ with $\mathbf{V}' = \widetilde{\mathbf{V}}$
-

Lemma 3.2. *With probability at least 0.95, [Algorithm 6](#) returns \mathbf{U}', \mathbf{V}' such that*

$$\|\mathbf{U}'\mathbf{V}' - \mathbf{A}\|_F^2 \leq (1 + \varepsilon) \min_{\mathbf{U} \in \{0, 1\}^{n \times k}, \mathbf{V} \in \{0, 1\}^{k \times d}} \|\mathbf{U}\mathbf{V} - \mathbf{A}\|_F^2,$$

where all matrix operations are performed in \mathbb{F}_2 .

Proof. Let $\widetilde{\mathbf{U}} \leftarrow [\widetilde{\mathbf{U}}^{(0)} | \dots | \widetilde{\mathbf{U}}^{(\ell)}]$ in [Algorithm 6](#). Let $\widetilde{\mathbf{M}}$ be the indicator matrix that selects a row of $\widetilde{\mathbf{U}}\widetilde{\mathbf{V}} = \widetilde{\mathbf{U}}\mathbf{V}'$ to match to each row of \mathbf{A} , so that by the optimality of \mathbf{U}' ,

$$\|\mathbf{U}'\mathbf{V}' - \mathbf{A}\|_F^2 \leq \|\widetilde{\mathbf{M}}\widetilde{\mathbf{U}}\widetilde{\mathbf{V}} - \mathbf{A}\|_F^2.$$

Since \mathbf{V} is a set of k points in $\{0, 1\}^d$ and each row \mathbf{U}_i of \mathbf{U} induces one of at most 2^k possible points $\mathbf{U}_i\mathbf{V} \in \{0, 1\}^d$, then $\|\mathbf{U}\mathbf{V} - \mathbf{A}\|_F^2$ is the objective value of a constrained 2^k -means clustering problem, even when all operations performed are on \mathbb{F}_2 . Similarly, $\mathbf{V}^{(j)}$ is a set of k points in $\{0, 1\}^d$ for each $j \in [\ell]$. Each row \mathbf{U}_i of \mathbf{U} induces one of at most 2^k possible points $\mathbf{U}_i\mathbf{V}^{(j)} \in \{0, 1\}^d$ for a fixed $j \in [\ell]$, so that $\|\mathbf{U}\mathbf{V}' - \mathbf{A}\|_F^2$ is the objective value of a constrained $2^k \ell$ -means clustering problem, even when all operations performed are on \mathbb{F}_2 .

Hence by the choice of t in [Theorem 2.6](#), it follows that $\tilde{\mathbf{A}}$ is a strong coreset, and so

$$\|\widetilde{\mathbf{M}}\widetilde{\mathbf{U}}\widetilde{\mathbf{V}} - \mathbf{A}\|_F^2 \leq (1 + \varepsilon) \|\widetilde{\mathbf{U}}\widetilde{\mathbf{V}} - \tilde{\mathbf{A}}\|_F^2.$$

We decompose the rows of $\tilde{\mathbf{A}}$ into $\mathbf{G}^{(0)}, \dots, \mathbf{G}^{(\ell)}$ for $\ell = \mathcal{O}\left(\frac{\log n}{\varepsilon}\right)$. Let G_i be the corresponding indices in $[n]$ so that $j \in G_i$ if and only if $\tilde{\mathbf{A}}_j$ is nonzero in \mathbf{G}_i . Then we have

$$\|\widetilde{\mathbf{U}}\widetilde{\mathbf{V}} - \tilde{\mathbf{A}}\|_F^2 = \sum_{i \in [\ell]} \sum_{j \in G_i} \|\mathbf{U}'_j \mathbf{V}' - \tilde{\mathbf{A}}_j\|_F^2.$$

Since each row in G_i is repeated a number of times in $[(1 + \varepsilon)^i, (1 + \varepsilon)^{i+1})$, then

$$\sum_{j \in G_i} \|\mathbf{U}'_j \mathbf{V}' - \widetilde{\mathbf{A}}_j\|_F^2 \leq (1 + \varepsilon)^2 \min_{\mathbf{U}^{(i)} \in \{0,1\}^{n \times k}, \mathbf{V}^{(i)} \in \{0,1\}^{k \times d}} \|\mathbf{U}^{(i)} \mathbf{V}^{(i)} - \mathbf{G}^{(i)}\|_F^2,$$

where the first factor of $(1 + \varepsilon)$ is from the $(1 + \varepsilon)$ -approximation guarantee of $\mathbf{U}^{(i)}$ and $\mathbf{V}^{(i)}$ by [Lemma 3.1](#) and the second factor of $(1 + \varepsilon)$ is from the number of each row in $\mathbf{G}^{(i)}$ varying by at most a $(1 + \varepsilon)$ factor. Therefore,

$$\begin{aligned} \|\mathbf{U}' \mathbf{V}' - \mathbf{A}\|_F^2 &\leq (1 + \varepsilon)^3 \sum_{i \in [\ell]} \min_{\mathbf{U}^{(i)} \in \{0,1\}^{n \times k}, \mathbf{V}^{(i)} \in \{0,1\}^{k \times d}} \|\mathbf{U}^{(i)} \mathbf{V}^{(i)} - \mathbf{G}^{(i)}\|_F^2 \\ &\leq (1 + \varepsilon)^3 \min_{\mathbf{U} \in \{0,1\}^{n \times k}, \mathbf{V} \in \{0,1\}^{k \times d}} \|\mathbf{U} \mathbf{V} - \widetilde{\mathbf{A}}\|_F^2. \end{aligned}$$

Let $\mathbf{U}^* \in \{0,1\}^{n \times k}$ and $\mathbf{V}^* \in \{0,1\}^{k \times d}$ such that

$$\|\mathbf{U}^* \mathbf{V}^* - \mathbf{A}\|_F^2 = \min_{\mathbf{U} \in \{0,1\}^{n \times k}, \mathbf{V} \in \{0,1\}^{k \times d}} \|\mathbf{U} \mathbf{V} - \mathbf{A}\|_F^2,$$

where all operations are performed in \mathbb{F}_2 . Let \mathbf{M}^* be the indicator matrix that selects a row of $\mathbf{U}^* \mathbf{V}^*$ to match to each row of $\widetilde{\mathbf{A}}$, so that by [Lemma 2.4](#),

$$\min_{\mathbf{U} \in \{0,1\}^{n \times k}, \mathbf{V} \in \{0,1\}^{k \times d}} \|\mathbf{U} \mathbf{V} - \widetilde{\mathbf{A}}\|_F^2 \leq (1 + \varepsilon) \|\mathbf{M}^* \mathbf{U}^* \mathbf{V}^* - \widetilde{\mathbf{A}}\|_F^2.$$

Then by the choice of t in [Theorem 2.6](#) so that $\widetilde{\mathbf{A}}$ is a strong coreset of \mathbf{A} ,

$$\|\mathbf{M}^* \mathbf{U}^* \mathbf{V}^* - \widetilde{\mathbf{A}}\|_F^2 \leq (1 + \varepsilon) \|\mathbf{U}^* \mathbf{V}^* - \mathbf{A}\|_F^2.$$

Therefore, we have

$$\|\mathbf{U}' \mathbf{V}' - \mathbf{A}\|_F^2 \leq (1 + \varepsilon)^5 \|\mathbf{U}^* \mathbf{V}^* - \mathbf{A}\|_F^2$$

and the desired claim then follows from rescaling ε . \square

It remains to analyze the runtime of [Algorithm 6](#).

Lemma 3.3. *Algorithm 6 uses $2^{\text{poly}(k/\varepsilon)}$ poly(n, d) runtime.*

Proof. By [Theorem 2.6](#), we have that [Algorithm 6](#) uses $\mathcal{O}\left(nd^2 + n^2d + \frac{nk d}{\varepsilon^2} + \frac{nk^2}{\varepsilon^2}\right)$ time to compute $\widetilde{\mathbf{A}} \in \{0,1\}^{N \times d}$ with $N = \text{poly}(n)$. By [Lemma 3.1](#), it takes $d \cdot (2^k)^{\text{poly}(k/\varepsilon)}$ time to compute $\widetilde{\mathbf{U}}^{(i)}, \widetilde{\mathbf{V}}^{(i)}$ for each $i \in [\ell]$ for $\ell = \mathcal{O}\left(\frac{\log n}{\varepsilon}\right)$. Hence, it takes $2^{\text{poly}(k/\varepsilon)}$ poly(n, d) runtime to compute $\widetilde{\mathbf{U}}$ and $\widetilde{\mathbf{V}}$. Finally, computing \mathbf{U}' via [Algorithm 5](#) takes $\mathcal{O}(2^{k'} n)$ time after enumerating through all possible $2^k \ell$ binary vectors for each row of \mathbf{U}' . Therefore, the total runtime of [Algorithm 4](#) is $2^{\text{poly}(k/\varepsilon)}$ poly(n, d). \square

By [Lemma 3.2](#) and [Lemma 3.3](#), we thus have:

Theorem 3.4. *There exists an algorithm that uses $2^{\text{poly}(k/\varepsilon)}$ poly(n, d) runtime and with probability at least $\frac{2}{3}$, outputs $\mathbf{U}' \in \{0,1\}^{n \times k'}$ and $\mathbf{V}' \in \{0,1\}^{k' \times d}$ such that*

$$\|\mathbf{U}' \mathbf{V}' - \mathbf{A}\|_F^2 \leq (1 + \varepsilon) \min_{\mathbf{U} \in \{0,1\}^{n \times k}, \mathbf{V} \in \{0,1\}^{k \times d}} \|\mathbf{U} \mathbf{V} - \mathbf{A}\|_F^2,$$

where $k' = \mathcal{O}\left(\frac{k \log k}{\varepsilon}\right)$.

4 L_p Low-Rank Approximation

In this section, we present a $(1 + \varepsilon)$ -approximation algorithm for binary low-rank approximation with L_p loss, where the goal is to find matrices $\mathbf{U} \in \{0, 1\}^{n \times k}$ and $\mathbf{V} \in \{0, 1\}^{k \times d}$ to minimize $\|\mathbf{UV} - \mathbf{A}\|_p^p$. We would like to use the same approach as in [Section 2](#), where we first compute a weighted matrix $\tilde{\mathbf{A}}$ from a strong coresets for \mathbf{A} , and then we make guesses for the matrices \mathbf{SU}^* and \mathbf{SA} and solve for $\min_{\mathbf{V} \in \{0, 1\}^{k \times d}} \|\mathbf{SU}^* \mathbf{V} - \mathbf{SA}\|_F^2$, while ensuring there are not too many possibilities for the matrices \mathbf{SU}^* and \mathbf{SA} . Thus to adapt this approach to L_p loss, we first require the following strong coreset construction for discrete metrics:

Theorem 4.1 (Theorem 1 in [\[CSS21\]](#)). *Let $X \subset \mathbb{R}^d$ be a subset of n points, $\varepsilon \in (0, 1)$ be an accuracy parameter, $p \geq 1$ be a constant, and let*

$$t = \mathcal{O}(\min(\varepsilon^{-2} + \varepsilon^{-p}, k\varepsilon^{-2}) \cdot k \log n).$$

There exists an algorithm that uses $\text{poly}(n, d, k)$ runtime and outputs a set of t weighted points that is a strong ε -coresets for k -clustering on discrete L_p metrics with probability at least 0.99. Moreover, each point has an integer weight that is at most $\text{poly}(n)$.

For Frobenius error, we crucially require the affine embedding property that

$$(1 - \varepsilon)\|\mathbf{U}^* \mathbf{V} - \mathbf{A}\|_F^2 \leq \|\mathbf{SU}^* \mathbf{V} - \mathbf{SA}\|_F^2 \leq (1 + \varepsilon)\|\mathbf{U}^* \mathbf{V} - \mathbf{A}\|_F^2,$$

for all $\mathbf{V} \in \{0, 1\}^{k \times d}$. Unfortunately, it is not known whether there exists an efficient sampling-based affine embedding for L_p loss.

We instead invoke the coreset construction of [Theorem 4.1](#) on the rows and the columns so that $\tilde{\mathbf{A}}$ has a small number of distinct rows and columns. We again use the idea from [Section 3](#) to partition the rows of $\tilde{\mathbf{A}}$ into groups based on their frequency, but now we further partition the groups based on the frequency of the columns. It then remains to solve BMF with L_p loss on the partition, each part of which has a small number of rows and columns. Because the contribution of each row toward the overall loss is small (because there is a small number of columns), it turns out that there exists a matrix that samples $\text{poly}(k/\varepsilon)$ rows of each partition that finally achieves the desired affine embedding. Thus, we can solve the problem on each partition, pad the factors accordingly, and build the bicriteria factors as in the binary field case. The algorithm appears in full in [Algorithm 9](#), with subroutines appearing in [Algorithm 7](#) and [Algorithm 8](#).

Algorithm 7 Algorithm for computing optimal \mathbf{U} given $\mathbf{V}^{(1)}, \dots, \mathbf{V}^{(\ell)}$

Input: $\tilde{\mathbf{A}} \in \{0, 1\}^{N \times d}$, $\mathbf{V}^{(1)}, \dots, \mathbf{V}^{(\ell)} \in \{0, 1\}^{k \times d}$

Output: $\mathbf{U}' = \text{argmin}_{\mathbf{U} \in \{0, 1\}^{N \times \ell k}} \|\mathbf{UV} - \tilde{\mathbf{A}}\|_p^p$, where \mathbf{U} is restricted to one nonzero block of k coordinates

- 1: **for** $i = 1$ to $i = N$ **do**
 - 2: Set $(\mathbf{U}'_i, j') = \text{argmin}_{\mathbf{U}_i \in \{0, 1\}^{1 \times k}, j \in [\ell]} \|(\mathbf{U}_i \mathbf{V}^{(j)} - \tilde{\mathbf{A}}_i)\|_p^p$ ▷Enumerate over all 2^k possible binary vectors, all ℓ indices
 - 3: Pad \mathbf{U}'_i with length ℓk , as the j' -th block of k coordinates
 - 4: **return** $\mathbf{U}' = \mathbf{U}'_1 \circ \dots \circ \mathbf{U}'_N$
-

We first justify the correctness of [Algorithm 8](#) by showing the existence of an L_0 sampling matrix \mathbf{S} that achieves a subspace embedding for binary inputs.

Algorithm 8 Low-rank approximation for matrix $\tilde{\mathbf{A}}$ with t distinct rows and t' distinct columns

Input: $\tilde{\mathbf{A}} \in \{0, 1\}^{N \times D}$ with at most t distinct rows and r distinct columns

Output: \mathbf{U}', \mathbf{V}' with $\|\mathbf{U}\mathbf{V} - \tilde{\mathbf{A}}\|_p \leq (1 + \varepsilon) \min_{\mathbf{U} \in \{0, 1\}^{N \times k}, \mathbf{V} \in \{0, 1\}^{k \times D}} \|\mathbf{U}\mathbf{V} - \tilde{\mathbf{A}}\|_p$

- 1: $V \leftarrow \emptyset$
 - 2: **for** each guess of $\mathbf{S}\mathbf{U}^*$ and $\mathbf{S}\mathbf{A}$, where \mathbf{S} is a L_0 sampling matrix with $m = \mathcal{O}\left(\frac{k^{p+1}}{\varepsilon^2} \log r\right)$ rows with weights that are powers of two up to $\text{poly}(N)$ **do**
 - 3: $V \leftarrow V \cup \text{argmin}_{\mathbf{V} \in \{0, 1\}^{k \times D}} \|\mathbf{S}\mathbf{U}^*\mathbf{V} - \mathbf{S}\mathbf{A}\|_p^p$ ▷ Algorithm 1 with L_p loss
 - 4: **for** each $\mathbf{V} \in V$ **do**
 - 5: Let $\mathbf{U}_{\mathbf{V}} = \text{argmin}_{\mathbf{U} \in \{0, 1\}^{N \times k}} \|\mathbf{U}\mathbf{V} - \mathbf{A}\|_p^p$ ▷ Algorithm 2 with L_p loss
 - 6: $\mathbf{V}' \leftarrow \text{argmin}_{\mathbf{V} \in \{0, 1\}^{k \times d}} \|\mathbf{S}\mathbf{U}_{\mathbf{V}}\mathbf{V} - \mathbf{S}\mathbf{A}\|_p^p$
 - 7: $\mathbf{U}' \leftarrow \mathbf{U}_{\mathbf{V}'}$
 - 8: **return** $(\mathbf{U}', \mathbf{V}')$
-

Algorithm 9 Bicriteria low-rank approximation with L_p loss for matrix \mathbf{A}

Input: $\mathbf{A} \in \{0, 1\}^{n \times d}$, rank parameter k , accuracy parameter $\varepsilon > 0$

Output: $\mathbf{U}' \in \{0, 1\}^{n \times k}, \mathbf{V}' \in \{0, 1\}^{k \times d}$ satisfying the property that $\|\mathbf{U}'\mathbf{V}' - \mathbf{A}\|_p^p \leq (1 + \varepsilon) \min_{\mathbf{U} \in \{0, 1\}^{n \times k}, \mathbf{V} \in \{0, 1\}^{k \times d}} \|\mathbf{U}\mathbf{V} - \mathbf{A}\|_p^p$

- 1: $t \leftarrow \mathcal{O}\left(\min(\varepsilon^{-2} + \varepsilon^{-p}, k\varepsilon^{-2}) \cdot k \log n\right)$ ▷ Theorem 4.1
 - 2: $\ell \leftarrow \mathcal{O}\left(\frac{\log n}{\varepsilon}\right), k' \leftarrow \ell k$
 - 3: Compute a strong coreset C for 2^k -means clustering of \mathbf{A} , with t rows, with weights $N = \text{poly}(n)$
 - 4: Compute a strong coreset C' for 2^k -means clustering of C , with t rows and columns, with weights $N, D = \text{poly}(n)$
 - 5: Let $\tilde{\mathbf{A}} \in \{0, 1\}^{N \times D}$ be the matrix representation of C , where weighted points are duplicated appropriately
 - 6: For $i \in [\ell]$, let $\mathbf{G}^{(i)}$ be the group of rows (removing multiplicity) of $\tilde{\mathbf{A}}$ with frequency $[(1 + \varepsilon)^i, (1 + \varepsilon)^{i+1})$
 - 7: For $i, j \in [\ell]$, let $\mathbf{G}^{(i,j)}$ be the group of columns (removing multiplicity) of $\mathbf{G}^{(i,j)}$ with frequency $[(1 + \varepsilon)^j, (1 + \varepsilon)^{j+1})$
 - 8: Compute the low-rank minimizers $(\widetilde{\mathbf{U}}^{(i,j)}, \widetilde{\mathbf{V}}^{(i,j)})$ on input $\mathbf{G}^{(i,j)}$ using Algorithm 8, padded to $\mathbb{R}^{n \times k}$ and $\mathbb{R}^{k \times D}$, respectively
 - 9: $\tilde{\mathbf{U}} \leftarrow \left[\widetilde{\mathbf{U}}^{(0,0)} \mid \widetilde{\mathbf{U}}^{(1,0)} \mid \dots \mid \widetilde{\mathbf{U}}^{(\ell,\ell)} \right], \tilde{\mathbf{V}} \leftarrow \widetilde{\mathbf{V}}^{(0,0)} \circ \widetilde{\mathbf{V}}^{(1,0)} \dots \circ \widetilde{\mathbf{V}}^{(\ell,\ell)}$
 - 10: Use Algorithm 7 with $\widetilde{\mathbf{U}}^{(0,0)}, \widetilde{\mathbf{U}}^{(1,0)}, \dots, \widetilde{\mathbf{U}}^{(\ell,\ell)}$ and C to find \mathbf{V}'
 - 11: Use \mathbf{V}' and \mathbf{A} to find \mathbf{U}' , i.e., Algorithm 2 with dimension k' and L_p loss
 - 12: **return** $(\mathbf{U}', \mathbf{V}')$
-

Lemma 4.2. Given matrices $\mathbf{A} \in \{0, 1\}^{n \times k}$ and $\mathbf{B} \in \{0, 1\}^{n \times r}$, there exists a matrix $\mathbf{S} \in \mathbb{R}^{m \times n}$ with $m = \mathcal{O}\left(\frac{k^{p+1}}{\varepsilon^2} \log r\right)$ such that with probability at least 0.99, we have that simultaneously for all $\mathbf{X} \in \{0, 1\}^{k \times r}$,

$$(1 - \varepsilon) \|\mathbf{A}\mathbf{X} - \mathbf{B}\|_p^p \leq \|\mathbf{S}\mathbf{A}\mathbf{X} - \mathbf{S}\mathbf{B}\|_p^p \leq (1 + \varepsilon) \|\mathbf{A}\mathbf{X} - \mathbf{B}\|_p^p.$$

Proof. Let $\mathbf{M} \in \{0, 1, \dots, k\}^{n \times 1}$ be an arbitrary matrix and let S be a set that contains the nonzero

rows of \mathbf{M} and has cardinality that is a power of two. That is, $|S| = 2^i$ for some integer $i \geq 0$. Let \mathbf{z} be a random element of S , i.e., a random non-zero row of \mathbf{M} , so that we have

$$\mathbb{E} [2^i \cdot \|\mathbf{z}\|_p^p] = \|\mathbf{M}\|_p^p.$$

Similarly, we have

$$\text{Var}(2^i \cdot \|\mathbf{z}\|_p^p) \leq 2^i k^p \leq 2k^p \|\mathbf{M}\|_p^p.$$

Hence if we repeat take the mean of $\mathcal{O}(\frac{k^p}{\varepsilon^2})$ estimators, we have that with probability at least 0.99,

$$(1 - \varepsilon)\|\mathbf{M}\|_p^p \leq \|\mathbf{SM}\|_p^p \leq (1 + \varepsilon)\|\mathbf{M}\|_p^p.$$

We can further improve the probability of success to $1 - \delta$ for $\delta \in (0, 1)$ by repeating $\mathcal{O}(\log \frac{1}{\delta})$ times. By setting $\mathbf{M} = \mathbf{Ax} - \mathbf{B}^{(i)}$ for fixed $\mathbf{A} \in \{0, 1\}^{n \times k}$, $\mathbf{x} \in \{0, 1\}^k$, and $\mathbf{B} \in \{0, 1\}^{n \times r}$ with $i \in [r]$, we have that the sketch matrix gives a $(1 + \varepsilon)$ -approximation to $\|\mathbf{Ax} - \mathbf{B}^{(i)}\|_p^p$. The result then follows from setting $\delta = \frac{1}{2^{k_r}}$, taking a union bound over all $\mathbf{x} \in \{0, 1\}^k$, and then a union bound over all $i \in [r]$. \square

We then justify the correctness of [Algorithm 9](#).

Lemma 4.3. *With probability at least 0.95, [Algorithm 9](#) returns \mathbf{U}' , \mathbf{V}' such that*

$$\|\mathbf{U}'\mathbf{V}' - \mathbf{A}\|_p^p \leq (1 + \varepsilon) \min_{\mathbf{U} \in \{0, 1\}^{n \times k}, \mathbf{V} \in \{0, 1\}^{k \times d}} \|\mathbf{UV} - \mathbf{A}\|_p^p.$$

Proof. Let \mathbf{M}_1 and \mathbf{M}_2 be the sampling and rescaling matrices used to acquire $\tilde{\mathbf{A}} \in \mathbb{R}^{N \times D}$, so that by the optimality of \mathbf{U}' ,

$$\|\mathbf{U}'\mathbf{V}' - \mathbf{A}\|_p^p \leq \|\mathbf{M}_1 \tilde{\mathbf{U}} \tilde{\mathbf{V}} \mathbf{M}_2 - \mathbf{A}\|_p^p.$$

Observe that \mathbf{V} is a set of k points in $\{0, 1\}^d$. Thus, each row \mathbf{U}_i of \mathbf{U} induces one of at most 2^k possible points $\mathbf{U}_i \mathbf{V} \in \{0, 1\}^d$. Hence, $\|\mathbf{UV} - \mathbf{A}\|_p^p$ is the objective value of a constrained 2^k -clustering problem under the L_p metric. Similarly, since $\mathbf{V}^{(j)}$ is a set of k points in $\{0, 1\}^d$ for each $j \in [\ell]$, then each row \mathbf{U}_i of \mathbf{U} induces one of at most 2^k possible points $\mathbf{U}_i \mathbf{V}^{(j)} \in \{0, 1\}^d$ for a fixed $j \in [\ell]$. Therefore, $\|\mathbf{UV}' - \mathbf{A}\|_p^p$ is the objective value of a constrained $2^k \ell$ -clustering problem under the L_p metric.

By the choice of t in [Theorem 4.1](#), $\tilde{\mathbf{A}}$ is a strong coresnet, and so

$$\|\mathbf{M}_1 \tilde{\mathbf{U}} \tilde{\mathbf{V}} \mathbf{M}_2 - \mathbf{A}\|_F^2 \leq (1 + \varepsilon) \|\tilde{\mathbf{U}} \tilde{\mathbf{V}} - \tilde{\mathbf{A}}\|_F^2.$$

We decompose the rows of $\tilde{\mathbf{A}}$ into groups $\mathbf{G}^{(0)}, \dots, \mathbf{G}^{(\ell)}$ for $\ell = \mathcal{O}(\frac{\log n}{\varepsilon})$. For each group $\mathbf{G}^{(i)}$, we decompose the columns of $\mathbf{G}^{(i)}$ into groups $\mathbf{G}^{(i,0)}, \dots, \mathbf{G}^{(i,\ell)}$ for $\ell = \mathcal{O}(\frac{\log n}{\varepsilon})$. Let G_i be the indices in $[n]$ corresponding to the rows in $\mathbf{G}^{(i)}$ and let $G_{i,j}$ be the indices in $[n]$ corresponding to the columns in $\mathbf{G}^{(i,j)}$. Then

$$\|\tilde{\mathbf{U}} \tilde{\mathbf{V}} - \tilde{\mathbf{A}}\|_p^p = \sum_{i \in [\ell]} \sum_{a \in G_i} \sum_{j \in [\ell]} \sum_{b \in G_{i,j}} \left| (\mathbf{U}'\mathbf{V}')_{a,b} - \tilde{\mathbf{A}}_{a,b} \right|^p.$$

Since each row in G_i is repeated a number of times in $[(1 + \varepsilon)^i, (1 + \varepsilon)^{i+1})$ and each column in $G_{i,j}$ is repeated a number of times in $[(1 + \varepsilon)^i, (1 + \varepsilon)^{i+1})$, then

$$\sum_{a \in G_i} \sum_{b \in G_{i,j}} \left| (\mathbf{U}'\mathbf{V}')_{a,b} - \tilde{\mathbf{A}}_{a,b} \right|^p \leq (1 + \varepsilon)^3 \min_{\mathbf{U} \in \{0,1\}^{n \times k}, \mathbf{V} \in \{0,1\}^{k \times d}} \sum_{a \in G_i} \sum_{b \in G_{i,j}} \left| (\mathbf{UV})_{a,b} - \tilde{\mathbf{A}}_{a,b} \right|^p,$$

where the first factor of $(1 + \varepsilon)$ is from the $(1 + \varepsilon)$ -approximation guarantee of $\mathbf{U}^{(i)}$ and $\mathbf{V}^{(i)}$ by [Lemma 3.1](#) and the second and third factors of $(1 + \varepsilon)$ is from the number of each row and each column in $\mathbf{G}^{(i,j)}$ varying by at most $(1 + \varepsilon)$ factor. Therefore,

$$\begin{aligned} \|\mathbf{U}'\mathbf{V}' - \mathbf{A}\|_p^p &\leq (1 + \varepsilon) \sum_{i \in [\ell]} \sum_{a \in G_i} \sum_{j \in [\ell]} \sum_{b \in G_{i,j}} \left| (\mathbf{U}'\mathbf{V}')_{a,b} - \tilde{\mathbf{A}}_{a,b} \right|^p \\ &\leq (1 + \varepsilon)^4 \min_{\mathbf{U} \in \{0,1\}^{n \times k}, \mathbf{V} \in \{0,1\}^{k \times d}} \|\mathbf{UV} - \tilde{\mathbf{A}}\|_p^p \end{aligned}$$

Let $\mathbf{U}^* \in \{0,1\}^{n \times k}$ and $\mathbf{V}^* \in \{0,1\}^{k \times d}$ be minimizers to the binary L_p low-rank approximation problem, so that

$$\|\mathbf{U}^*\mathbf{V}^* - \mathbf{A}\|_p^p = \min_{\mathbf{U} \in \{0,1\}^{n \times k}, \mathbf{V} \in \{0,1\}^{k \times d}} \|\mathbf{UV} - \mathbf{A}\|_p^p.$$

Let \mathbf{M}_3 and \mathbf{M}_4 be the indicator matrices that select rows and columns of $\mathbf{U}^*\mathbf{V}^*$ to match to each row of $\tilde{\mathbf{A}}$, so that by [Lemma 2.4](#),

$$\min_{\mathbf{U} \in \{0,1\}^{n \times k}, \mathbf{V} \in \{0,1\}^{k \times d}} \|\mathbf{UV} - \tilde{\mathbf{A}}\|_p^p \leq (1 + \varepsilon) \|\mathbf{M}_3\mathbf{U}^*\mathbf{V}^*\mathbf{M}_4 - \tilde{\mathbf{A}}\|_p^p.$$

Then by the choice of t in [Theorem 4.1](#) so that $\tilde{\mathbf{A}}$ is a strong coreset of \mathbf{A} ,

$$\|\mathbf{M}_3\mathbf{U}^*\mathbf{V}^*\mathbf{M}_4 - \tilde{\mathbf{A}}\|_p^p \leq (1 + \varepsilon) \|\mathbf{U}^*\mathbf{V}^* - \mathbf{A}\|_p^p.$$

Therefore,

$$\|\mathbf{U}'\mathbf{V}' - \mathbf{A}\|_p^p \leq (1 + \varepsilon)^6 \|\mathbf{U}^*\mathbf{V}^* - \mathbf{A}\|_p^p$$

and the desired claim then follows from rescaling ε . \square

We now analyze the runtime of [Algorithm 9](#).

Lemma 4.4. *For any constant $p \geq 1$, [Algorithm 9](#) uses $2^{\text{poly}(k/\varepsilon)} \text{poly}(n, d)$ runtime.*

Proof. By [Theorem 4.1](#), we have that [Algorithm 9](#) uses $2^{\mathcal{O}(k)} \cdot \text{poly}(n, d)$ time to compute $\tilde{\mathbf{A}} \in \{0,1\}^{N \times D}$ with $N, D = \text{poly}(n)$. We now consider the time to compute $\widetilde{\mathbf{U}^{(i,j)}}, \widetilde{\mathbf{V}^{(i,j)}}$ for each $i, j \in [\ell]$ for $\ell = \mathcal{O}\left(\frac{\log n}{\varepsilon}\right)$. For each i, j , we make guesses for \mathbf{SU}^* and \mathbf{SA} in Since \mathbf{SU}^* and \mathbf{SA} have $m = \mathcal{O}\left(\frac{k^{p+1} \log r}{\varepsilon^2}\right)$ rows, then there are $\binom{t}{m}$ possible choices for \mathbf{SU}^* and $\binom{t}{m}$ choices for \mathbf{SA} , where $t = \frac{2^k \log n}{\varepsilon^p}$. Hence, there are $2^{\text{poly}(k/\varepsilon)} \text{poly}(n, d)$ possible guesses for \mathbf{SU}^* and \mathbf{SA} .

For each guess of \mathbf{SU}^* and \mathbf{SA} , [Algorithm 8](#) iterates through the columns of $\widetilde{\mathbf{V}^{(i,j)}}$, which uses $2^{\mathcal{O}(k)} \cdot \text{poly}(n, d)$ time. Similarly, the computation of $\widetilde{\mathbf{U}^{(i,j)}}$, \mathbf{U}' , and \mathbf{V}' all take $2^{\mathcal{O}(k)} \cdot \text{poly}(n, d)$ time. Therefore, the total runtime of [Algorithm 9](#) is $2^{\text{poly}(k/\varepsilon)} \text{poly}(n, d)$. \square

By Lemma 4.3 and Lemma 4.4, we thus have:

Theorem 4.5. *For any constant $p \geq 1$, there exists an algorithm that uses $2^{\text{poly}(k/\varepsilon)} \text{poly}(n, d)$ runtime and with probability at least $\frac{2}{3}$, outputs $\mathbf{U}' \in \{0, 1\}^{n \times k'}$ and $\mathbf{V}' \in \{0, 1\}^{k' \times d}$ such that*

$$\|\mathbf{U}'\mathbf{V}' - \mathbf{A}\|_p^p \leq (1 + \varepsilon) \min_{\mathbf{U} \in \{0, 1\}^{n \times k}, \mathbf{V} \in \{0, 1\}^{k \times d}} \|\mathbf{UV} - \mathbf{A}\|_p^p,$$

where $k' = \mathcal{O}\left(\frac{k \log^2 k}{\varepsilon^2}\right)$.

We note here that the $\text{poly}(k/\varepsilon)$ term in the exponent hides a k^p factor, as we assume p to be a (small) constant.

5 Applications to Big Data Models

This section describes how we can generalize our techniques to big data models such as streaming or distributed models.

Algorithmic modularization. To adapt our algorithm to the streaming model or the distributed model, we first present a high-level modularization of our algorithm across all applications, i.e., Frobenius binary low-rank approximation, binary low-rank approximation over \mathbb{F}_2 , and binary low-rank approximation with L_p loss. We are given the input matrix $\mathbf{A} \in \{0, 1\}^{n \times d}$ in each of these settings. We first construct a weighted coreset $\tilde{\mathbf{A}}$ for \mathbf{A} . We then perform a number of operations on $\tilde{\mathbf{A}}$ to obtain low-rank factors $\tilde{\mathbf{U}}$ and $\tilde{\mathbf{V}}$ for $\tilde{\mathbf{A}}$. Setting $\mathbf{V}' = \tilde{\mathbf{V}}$, our algorithms finally use \mathbf{A} and \mathbf{V}' to construct the optimal factor \mathbf{U}' to match \mathbf{V}' .

5.1 Streaming Model

We can adapt our approach to the streaming model, where either the rows or columns of the input matrix arrive sequentially. For brevity, we shall only discuss the setting where the rows of the input matrix arrive sequentially; the setting where the columns of the input matrix arrive sequentially is symmetric.

Formal streaming model definition. We consider the two-pass row-arrival variant of the streaming model. In this setting, the rank parameter k and the accuracy parameter $\varepsilon > 0$ are given to the algorithm before the data stream. The input matrix $\mathbf{A} \in \{0, 1\}^{n \times d}$ is then defined through the sequence of row-arrivals, $\mathbf{A}_1, \dots, \mathbf{A}_n \in \{0, 1\}^d$, so that the i -th row that arrives in the data stream is \mathbf{A}_i . The algorithm passes over the data twice so that in the first pass, it can store some sketch S that uses space sublinear in the input size, i.e., using $o(nd)$ space. After the first pass, the algorithm can perform some post-processing on S and then must output factors $\mathbf{U} \in \{0, 1\}^{n \times k}$ and $\mathbf{V} \in \{0, 1\}^{k \times d}$ after being given another pass over the data, i.e., the rows $\mathbf{A}_1, \dots, \mathbf{A}_n \in \{0, 1\}^d$.

Two-pass streaming algorithm. To adapt our algorithm to the two-pass streaming model, recall the high-level modularization of our algorithm described at the beginning of Section 5. The first step is constructing a coreset $\tilde{\mathbf{A}}$ of \mathbf{A} . Whereas our previous coreset constructions were offline, we now require a streaming algorithm to produce the coreset $\tilde{\mathbf{A}}$. To that end, we use the following

well-known merge-and-reduce paradigm for converting an offline coresets construction to a coresets construction in the streaming model.

Theorem 5.1. *Suppose there exists an algorithm that, with probability $1 - \frac{1}{\text{poly}(n)}$, produces an offline coresets construction that uses $f(n, \varepsilon)$ space, suppressing dependencies on other input parameters, such as k and p . Then there exists a one-pass streaming algorithm that, with probability $1 - \frac{1}{\text{poly}(n)}$, produces a coresets that uses $f(n, \varepsilon') \cdot \mathcal{O}(\log n)$ space, where $\varepsilon' = \frac{\varepsilon}{\log n}$.*

In the first pass of the stream, we can use [Theorem 5.1](#) to construct a strong coresets C of \mathbf{A} with accuracy $\mathcal{O}(\varepsilon)$. However, C will have $2^{\text{poly}(k)} \cdot \text{poly}(\frac{1}{\varepsilon}, \log n)$ rows, and thus, we cannot immediately duplicate the rows of C to form $\tilde{\mathbf{A}}$ because we cannot have $\log n$ dependencies in the number of rows of $\tilde{\mathbf{A}}$.

After the first pass of the stream, we further apply the respective offline coresets construction, i.e., [Theorem 2.6](#) or [Theorem 4.1](#) to C to obtain a coresets C' with accuracy ε and a number of rows independent of $\log n$. We then use C' to form $\tilde{\mathbf{A}}$ and perform a number of operations on $\tilde{\mathbf{A}}$ to obtain low-rank factors $\tilde{\mathbf{U}}$ and $\tilde{\mathbf{V}}$ for $\tilde{\mathbf{A}}$. Setting $\mathbf{V}' = \tilde{\mathbf{V}}$, we can finally use the second pass of the data stream over \mathbf{A} , along with \mathbf{V}' , to construct the optimal factor \mathbf{U}' to match \mathbf{V}' . Thus the two-pass streaming algorithm uses $2^{\text{poly}(k)} \cdot d \cdot \text{poly}(\frac{1}{\varepsilon}, \log n)$ total space in the row-arrival model. For the column-arrival model, the two-pass streaming algorithm uses $2^{\text{poly}(k)} \cdot n \cdot \text{poly}(\frac{1}{\varepsilon}, \log d)$ total space.

5.2 Two-round distributed algorithm.

Our approach can also be adapted to the distributed model, where the rows or columns of the input matrix are partitioned across multiple users. For brevity, we again discuss the setting where the rows of the input matrix are partitioned; the setting where the columns of the input matrix are partitioned is symmetric.

Formal distributed model definition. We consider the two-round distributed model, where the rank parameter k and the accuracy parameter $\varepsilon > 0$ are known in advance to all users. The input matrix $\mathbf{A} \in \{0, 1\}^{n \times d}$ is then defined arbitrarily through the union of rows, $\mathbf{A}_1, \dots, \mathbf{A}_n \in \{0, 1\}^d$, where each row \mathbf{A}_i may be given to any of γ users. An additional central coordinator sends and receives messages from the users. The protocol is then permitted to use two rounds of communication so that in the first round, the protocol can send $o(nd)$ bits of communication. The coordinator can process the communication to form some sketch S , perform some post-processing on S , and then request additional information from each user, possibly using $o(nd)$ communication to specify the information demanded from each user. After the users again use $o(nd)$ bits of communication in the second round of the protocol, the central coordinator must output factors $\mathbf{U} \in \{0, 1\}^{n \times k}$ and $\mathbf{V} \in \{0, 1\}^{k \times d}$.

Two-round distributed algorithm. To adapt our algorithm to the two-round distributed model, again recall the high-level modularization of our algorithm described at the beginning of [Section 5](#). The first step is constructing a coresets $\tilde{\mathbf{A}}$ of \mathbf{A} . Whereas our previous coresets constructions were offline, we now require a distributed algorithm to produce the coresets $\tilde{\mathbf{A}}$. To that end, we request that each of the t users send a coresets with accuracy $\mathcal{O}(\varepsilon)$ of their respective rows. Note that each user can construct the coresets locally without requiring any communication since

the coreset is only a summary of the rows held by the user. Thus the total communication in the first round is just the offline coreset size times the number of players, i.e., $\gamma \cdot 2^{\text{poly}(k)} \cdot \text{poly}(\frac{1}{\varepsilon}, \log n)$ rows.

Given the union C of the coresets sent by all users, the central coordinator then constructs a coreset C' of \mathbf{A} with accuracy ε , again using an offline coreset construction. The coordinator then uses C' to form $\tilde{\mathbf{A}}$ and performs the required operations on $\tilde{\mathbf{A}}$ to obtain low-rank factors $\tilde{\mathbf{U}}$ and $\tilde{\mathbf{V}}$ for $\tilde{\mathbf{A}}$.

The coordinator can then send \mathbf{V}' to all players, using \mathbf{V}' and their local subset rows of \mathbf{A} to construct \mathbf{U}' collectively. The users then send the rows of \mathbf{U}' corresponding to the rows of \mathbf{A} local to the user back to the central coordinator, who can then construct \mathbf{U}' . Thus the second round of the protocol uses $\tilde{\mathcal{O}}(nk + kd) \cdot \text{poly}(\frac{1}{\varepsilon})$ bits of communication. Hence, the total communication of the protocol is $d\gamma \cdot 2^{\text{poly}(k)} \cdot \text{poly}(\frac{1}{\varepsilon}, \log n) + \tilde{\mathcal{O}}(nk + kd) \cdot \text{poly}(\frac{1}{\varepsilon})$ in the two-round row-partitioned distributed model. For the two-round column-partitioned distributed model, the total communication of the protocol is $n\gamma \cdot 2^{\text{poly}(k)} \cdot \text{poly}(\frac{1}{\varepsilon}, \log d) + \tilde{\mathcal{O}}(nk + kd) \cdot \text{poly}(\frac{1}{\varepsilon})$.

6 Experiments

In this section, we aim to evaluate the feasibility of the algorithmic ideas of our paper against existing algorithms for binary matrix factorization from previous literature. The running time of our full algorithms for BMF is prohibitively expensive, even for small k , so our algorithm will be based on the idea of [KPRW19], who only run their algorithms in part, obtaining weaker theoretical guarantees. Indeed, by simply performing k -means clustering, they obtained a simple algorithm that outperformed more sophisticated heuristics in practice.

We perform two main types of experiments, first comparing the algorithm presented in the next section against existing baselines and then showing the feasibility of using coresets in the BMF setting.

Baseline and algorithm. We compare several algorithms for binary matrix factorization that have implementations available online, namely the algorithm by Zhang et al. [ZLDZ07], which has been implemented in the NIMFA library [ZZ12], the message passing algorithm of Ravanbakhsh et al. [RPG16], as well as our implementation of the algorithm used in the experiments of [KPRW19]. We refer to these algorithms as Zh, MP, and kBMF, respectively. We choose the default parameters provided by the implementations. We chose the maximum number of rounds for the iterative methods so that the runtime does not exceed 20 seconds, as all methods besides [KPRW19] are iterative. However, in our experiments, the algorithms usually converged to a solution below the maximum number of rounds. We let every algorithm use the matrix operations over the preferred semiring, i.e. boolean, integer, or and-or matrix multiplication, in order to achieve the best approximation. We additionally found a binary matrix factorization algorithm for sparse matrices based on subgradient descent and random sampling¹ that is not covered in the literature. This algorithm was excluded from our experiments as it did not produce binary factors in our experiments. Specifically, we found that it produces real-valued \mathbf{U} and \mathbf{V} , and requires binarizing the product \mathbf{UV} after multiplication, therefore not guaranteeing that the binary matrix is of rank k .

¹<https://github.com/david-cortes/binmf>

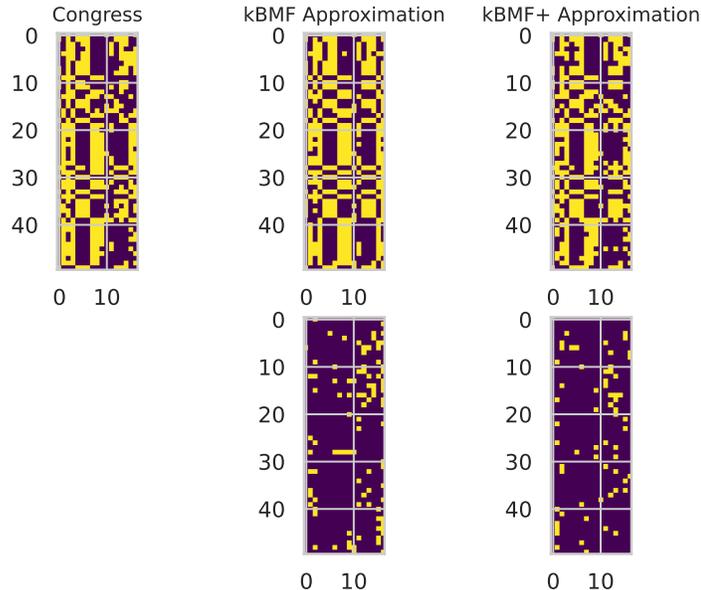


Fig. 1: A demonstration of the improved approximation of our algorithm over the algorithm used in the experiments of [KPRW19]. In the first column, we show the first 50 rows of the congress data set, where purple indicates 0 and yellow indicates 1. The next columns show the approximation of [KPRW19], and our algorithm’s approximation, both with $k = 10$. The second row indicates the entries in which the respective approximations differ from the original dataset in yellow. Our experiments found that the number of wrongly reconstructed entries almost halved from the kBMF to the kBMF+ algorithm on this dataset for $k = 10$.

Motivated by the idea of partially executing a more complicated algorithm with strong theoretical guarantees, we build upon the idea of finding a k -means clustering solution as a first approximation and mapping the Steiner points to their closest neighbors in \mathbf{A} , giving us a matrix \mathbf{V} of k binary points, and a matrix \mathbf{U} of assignments of the points of \mathbf{A} to their nearest neighbors. This solution restricts \mathbf{U} to have a single non-zero entry per row. Instead of outputting this \mathbf{U} as [KPRW19] did, we solve the minimization problem $\min_{\mathbf{U} \in \{0,1\}^{n \times k}} \|\mathbf{UV} - \mathbf{A}\|_F^2$ exactly at a cost of 2^k per row, which is affordable for small k . For a qualitative example of how this step improves the solution quality, see Figure 1. We call this algorithm kBMF+.

Using k -means as the first step in a binary matrix factorization algorithm is well-motivated by the theoretical and experimental results of [KPRW19], but does not guarantee a $(1 + \epsilon)$ -approximation. However, as we do not run the full algorithm, we are not guaranteed a $(1 + \epsilon)$ -approximation either way, as unfortunately, guessing the optimal matrix \mathbf{V} is very time-consuming. We would first have to solve the sketched problem $\|\tilde{\mathbf{S}}\mathbf{A} - \mathbf{S}\mathbf{U}\mathbf{V}\|_F^2$ for all guesses of $\mathbf{S}\mathbf{A}$ and $\mathbf{S}\mathbf{U}$.

We implement our algorithm and the one of [KPRW19] in Python 3.10 and numpy. For solving k -means, we use the implementation of Lloyd’s algorithm with k -means++ seeding provided by the scikit-learn library [PVG+11]. All experiments were performed on a Linux notebook with a 3.9 GHz 12th generation Intel Core i7 six-core processor with 32 gigabytes of RAM.

Datasets. We use both real and synthetic data for our experiments. We choose two datasets from the UCI Machine Learning Repository [DG17], namely the voting record of the 98th Congress, consisting of 435 rows of 16 binary features representing each congressperson’s vote on one of 16 bills, and the Thyroid dataset², of 9371 patient data comprising 31 features. We restricted ourselves to only binary features, leaving us with 21 columns. Finally, we use the ORL dataset of faces, which we binarize using a threshold of 0.33, as in [KPRW19].

For our synthetic data, we generate random matrices, where each entry is set to be 1 independently with probability p , at two different sparsity levels of $p \in \{0.1, 0.5\}$. Additionally, we generate low-rank matrices by generating $\mathbf{U} \in \{0, 1\}^{n \times k}$ and $\mathbf{V} \in \{0, 1\}^{k \times d}$ and multiplying them together in \mathbb{F}_2 . We generate \mathbf{U} and \mathbf{V} at different sparsity levels of 0.5 and 0.1, for $k \in \{5, 10, 15\}$. Finally, we also use these matrices with added noise, where after multiplying, each bit is flipped with probability $p_e \in \{0.01, 0.001\}$.

We generate 25 matrices of size 250×50 for each configuration. These classes are named, in order of introduction: full, lr, and noisy.

Limitations. We opted to use only binary datasets, thus limiting the available datasets for our experiments. Because of this, our largest dataset’s size is less than 10000. Our algorithms are practical for these sizes and the parameters k we have chosen. Investigating the feasibility of algorithms for binary matrix factorization for large datasets may be an interesting direction for future research.

6.1 Comparing Algorithms for BMF

Synthetic data. For each algorithm, Table 2 shows the mean Frobenius norm error (i.e. $\text{err}_{\mathbf{A}}(\mathbf{U}, \mathbf{V}) = \|\mathbf{UV} - \mathbf{A}\|_F$) across 10 runs of each algorithm and the mean runtime in milliseconds for the synthetic datasets described above. For our choices of parameters, we find that all algorithms terminate in under a second, with Zhang’s algorithm and BMF being the fastest and the message-passing algorithm generally being the slowest. This is, of course, also influenced by the fact that the algorithms’ implementations use different technologies, which limits the conclusions we can draw from the data. We find that the kBMF+ algorithm slows down by a factor of 1.5 for small $k \in \{2, 3, 5\}$, and 15 when $k = 15$, compared to the kBMF algorithm.

This is offset by the improved error, where our algorithm kBMF+ generally achieves the best approximation for dense matrices, being able to sometimes find a perfect factorization, for example, in the case of a rank 5 matrix, when using $k \in \{10, 15\}$. Even when the perfect factorization is not found, we see that the Frobenius norm error is 2-10 times lower. On sparse matrices, we find that Zhang’s and the message-passing algorithms outperform kBMF+, yielding solutions that are about 2 times better in the worst case (matrix of rank 5, with sparsity 0.1 and $k = 5$). The kBMF algorithm generally performs the worst across datasets, which is surprising considering the results of [KPRW19]. Another point of note is that Zhang’s algorithm is tuned for sparse matrices, sometimes converging to factors that yield real-valued matrices. If so, we attempted to round the matrix as best we could.

Real data. As before, Table 3 shows the algorithms’ average Frobenius norm error and average running time. We observe, that all algorithms are fairly close in Frobenius norm error, with the

²<https://www.kaggle.com/datasets/emmanuelwerr/thyroid-disease-data>

Dataset	Alg k	Error [Frobenius norm]				Time [ms]			
		kBMF	kBMF+	MP	Zh	kBMF	kBMF+	MP	Zh
Random $p = 0.5$	2	75.8	72.3	71.3	71.3	11.2	8.6	280.7	11.6
	3	74.3	69.9	69.4	68.7	14.9	12.5	309.8	11.7
	5	72.2	65.8	66.6	64.9	10.9	11.5	347.7	13.3
	10	68.7	57.4	61.5	58.5	15.4	53.4	486.6	17.2
	15	66.4	50.4	57.9	53.7	16.2	272.1	667.3	21.7
Random $p = 0.1$	2	36.0	35.0	34.9	35.2	10.8	11.3	277.3	9.9
	3	35.9	34.9	34.9	35.0	7.5	13.9	302.1	10.6
	5	35.6	34.6	35.5	34.2	12.7	18.5	336.9	12.6
	10	35.0	33.9	35.8	31.7	17.0	64.5	459.6	15.9
	15	34.3	33.0	38.5	29.0	20.9	269.5	628.4	19.6
Low-Rank $r = 5$ $p = 0.5$	2	72.5	67.1	66.0	67.8	4.1	7.9	274.9	11.9
	3	69.2	60.0	62.3	64.0	12.8	12.0	301.5	13.5
	5	64.0	26.9	55.2	56.7	10.4	11.9	339.8	15.4
	10	52.9	0.7	41.0	42.5	14.7	72.7	472.5	19.5
	15	43.3	0.0	32.8	31.1	18.0	296.0	658.0	23.8
Low-Rank $r = 5$ $p = 0.1$	2	20.5	20.4	16.5	15.8	9.4	6.3	185.6	4.8
	3	17.0	16.6	13.1	12.0	5.0	5.8	209.1	12.3
	5	11.1	8.4	4.6	5.1	7.0	8.0	275.9	14.8
	10	5.1	0.0	0.7	2.3	19.3	75.0	460.5	18.1
	15	1.5	0.0	0.4	1.4	20.2	297.0	630.9	22.1
Low-Rank $r = 10$ $p = 0.5$	2	75.8	72.2	71.1	71.7	13.4	15.5	281.2	11.5
	3	74.3	69.6	69.1	69.0	15.8	20.0	308.0	11.7
	5	72.0	64.7	66.1	64.8	20.9	19.7	345.5	13.6
	10	68.2	28.4	60.2	57.9	16.2	51.4	477.8	17.3
	15	65.6	0.8	56.0	52.9	19.3	245.2	659.6	21.3
Low-Rank $r = 10$ $p = 0.5$	2	30.8	30.5	27.6	28.5	10.0	14.3	213.4	5.7
	3	28.5	28.1	25.2	25.5	11.1	13.3	248.5	11.5
	5	24.7	23.2	20.4	19.9	13.1	18.7	292.0	13.4
	10	18.3	10.2	7.6	8.8	16.4	76.2	434.6	16.9
	15	15.2	2.5	4.7	5.4	14.8	261.3	638.8	22.1
Low-Rank $r = 15$ $p = 0.5$	2	75.7	72.3	71.2	71.3	14.5	18.6	277.6	11.3
	3	74.2	69.9	69.3	68.7	12.7	11.1	306.5	11.7
	5	72.1	65.7	66.6	64.8	15.0	19.0	339.7	13.0
	10	68.6	56.5	61.5	58.4	18.7	51.4	478.3	17.2
	15	66.4	29.2	57.7	53.6	13.0	239.9	652.8	21.1
Low-Rank $r = 15$ $p = 0.1$	2	38.7	38.2	35.6	36.5	12.1	10.4	242.2	9.7
	3	37.1	36.2	33.7	34.2	10.0	13.0	274.1	12.8
	5	33.7	32.2	29.8	29.5	13.2	17.9	313.2	14.6
	10	28.1	22.3	20.3	19.8	20.2	56.3	457.3	17.9
	15	25.3	14.2	11.6	13.4	21.2	247.9	643.8	21.2
Noisy $r = 10$ $p = 0.5$ $p_{noise} = 0.001$	2	75.8	72.3	71.2	71.6	13.9	12.8	290.4	11.3
	3	74.3	69.6	69.3	69.0	13.8	15.6	309.3	11.6
	5	72.1	64.7	66.2	65.0	17.6	23.8	345.8	13.6
	10	68.2	33.8	60.3	58.1	16.8	54.0	481.1	17.6
	15	65.6	4.8	56.2	53.2	18.4	247.1	661.8	21.6
Noisy $r = 10$ $p = 0.1$ $p_{noise} = 0.001$	2	32.5	32.1	29.3	30.0	6.3	9.6	223.6	7.6
	3	30.0	29.5	26.9	27.1	6.4	10.1	255.4	11.6
	5	26.2	24.6	22.0	21.3	6.6	9.7	291.9	13.5
	10	19.8	12.0	9.3	10.4	16.4	67.4	441.2	18.2
	15	16.7	4.9	6.8	7.2	13.9	255.0	641.8	22.4
Noisy $r = 10$ $p = 0.5$ $p_{noise} = 0.01$	2	75.8	72.1	71.0	71.7	9.7	11.4	276.1	11.4
	3	74.3	69.5	69.0	69.1	12.1	13.3	302.4	12.0
	5	72.0	64.7	66.0	64.8	12.4	12.5	338.9	13.4
	10	68.3	38.2	60.2	57.9	15.0	50.7	475.0	17.2
	15	65.7	16.7	56.1	52.8	18.0	254.0	672.9	21.3
Noisy $r = 10$ $p = 0.1$ $p_{noise} = 0.01$	2	33.3	33.0	30.3	30.9	9.9	11.5	225.3	9.2
	3	31.3	30.8	28.2	28.0	10.8	10.5	257.5	12.5
	5	27.8	26.2	23.6	23.4	9.4	18.3	292.1	14.3
	10	22.3	16.3	14.0	15.1	21.0	58.5	448.5	17.4
	15	19.9	12.5	12.5	12.0	20.5	260.3	645.4	21.7

Table 2: The average running time and error for different Binary Matrix Factorization algorithms on synthetic datasets. The minimum Frobenius norm error is marked in bold.

Dataset	Alg k	Error [Frobenius norm]				Time [ms]			
		kBMF	kBMF+	MP	Zh	kBMF	kBMF+	MP	Zh
Congress	2	40.0	38.8	38.8	36.4	2.0	3.3	280.7	6.9
	3	38.4	36.6	35.9	32.7	2.3	4.1	311.2	13.6
	5	35.7	32.7	31.1	27.7	4.6	5.2	332.9	16.2
	10	32.7	23.9	22.5	18.4	3.2	16.9	407.1	22.6
	15	30.9	14.8	15.5	9.6	7.4	246.7	480.5	27.5
ORL	2	39.4	37.8	35.9	33.5	2.0	2.9	203.7	11.6
	3	35.7	34.6	32.2	29.7	2.9	4.7	241.6	13.1
	5	31.7	30.7	27.7	25.6	3.8	5.8	289.4	15.4
	10	26.4	25.7	21.6	21.4	4.3	22.3	415.7	19.1
	15	23.4	22.8	17.8	19.7	6.1	318.0	575.5	22.2
Thyroid	2	106.6	98.6	90.5	91.6	12.6	14.2	7063.6	44.3
	3	94.5	90.5	75.5	73.9	14.4	18.7	7822.0	92.9
	5	82.7	80.4	78.5	61.8	31.8	25.2	8860.2	132.1
	10	66.0	55.4	54.0	52.9	28.9	59.6	12686.3	241.4
	15	57.6	38.9	39.2	46.7	26.7	313.4	16237.7	432.7

Table 3: The average running time and error for different Binary Matrix Factorization algorithms on real datasets, minimum frobenius norm error highlighted in bold.

best and worst factorizations’ error differing by about up to a factor of 3 across parameters and datasets. Zhang’s algorithm performs best on the Congress dataset, while the message-passing algorithm performs best on the ORL and Thyroid datasets. The kBMF algorithm generally does worst, but the additional processing we do in kBMF+ can improve the solution considerably, putting it on par with the other heuristics. On the Congress dataset, kBMF+ is about 1.1-2 times worse than Zhang’s, while on the ORL dataset, it is about 10-30% worse than the message-passing algorithm. Finally, the Thyroid dataset’s error is about 10-20% worse than competing heuristics.

We note that on the Thyroid datasets, which has almost 10000 rows, Zhang’s algorithm slows considerably, about 10 times slower than kBMF and even slower than kBMF+ for $k = 15$. This suggests that for large matrices and small to moderate k , the kBMF+ algorithm may actually run faster than other heuristics while providing comparable results. The message-passing algorithm slows tremendously, being almost three orders of magnitude slower than kBMF, but we believe this could be improved with another implementation.

Discussion. In our experiments, we found that on dense synthetic data, the algorithm kBMF+ outperforms other algorithms for the BMF problem. Additionally, we found that is competitive for sparse synthetic data and real datasets. One inherent benefit of the kBMF and kBMF+ algorithms is that they are very easily adapted to different norms and matrix products, as the clustering step, nearest neighbor search, and enumeration steps are all easily adapted to the setting we want. A benefit is that the factors are guaranteed to be either 0 or 1, which is not true for Zhang’s heuristic, which does not always converge. None of the existing heuristics consider minimization of L_p norms, so we omitted experimental data for this setting, but we note here that the results are qualitatively similar, with our algorithm performing best on dense matrices, and the heuristics performing well on sparse data.

6.2 Using Coresets with our Algorithm

Motivated by our theoretical use of strong coresets for k -means clustering, we perform experiments to evaluate the increase in error using them. To this end, we run the BMF+ algorithm on either the entire dataset, a coreset constructed via importance sampling [BLK17, BFL⁺21], or a lightweight

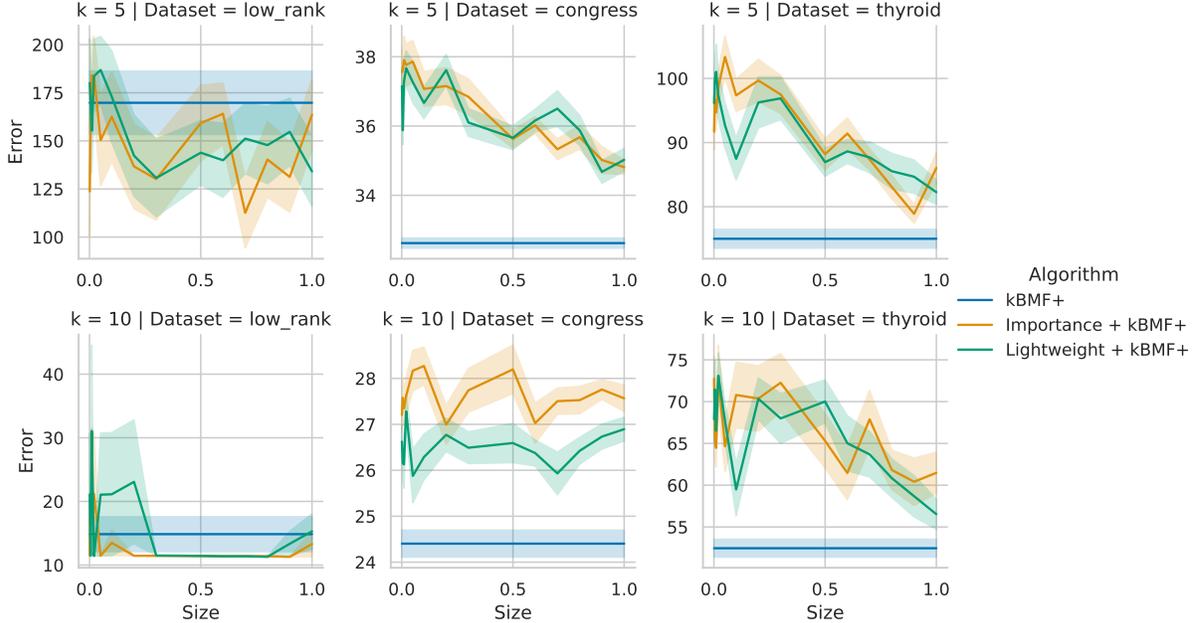


Fig. 2: A plot of the effect of different relative coresets sizes on the results of our algorithm.

coreset [BLK18]. Both of these algorithms were implemented in Python. The datasets in this experiment are a synthetic low-rank dataset with additional noise (size 5000×50 , rank 5 and 0.0005 probability of flipping a bit), the congress, and thyroid datasets.

We construct coresets of size rn for each $r \in \{0.001, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2, \dots, 0.9\}$. We sample 10 coresets at every size and use them when finding \mathbf{V} in our BMF+ algorithm. Theory suggests that the quality of the coreset depends only on k and the dimension of the points d , which is why in Figure 2, we observe a worse approximation for a given size of coreset for larger k . We find that the BMF+ algorithm performs just as well on lightweight coresets as the one utilizing the sensitivity sampling framework. This is expected in the binary setting, as the additive error in the weaker guarantee provided by lightweight coresets depends on the dataset’s diameter. Thus, the faster, lightweight coreset construction appears superior in this setting.

We observe that using coreset increases the Frobenius norm error we observe by about 35%, but curiously, on the low-rank dataset, the average error decreased after using coresets. This may be due to coreset constructions not sampling the noisy outliers that are not in the low-dimensional subspace spanned by the non-noisy low-rank matrix, letting the algorithm better reconstruct the original factors instead.

Our datasets are comparatively small, none exceeding 1000 points, which is why, in combination with the fact that the coreset constructions are not optimized, we observe no speedup compared to the algorithm without coresets. However, even though constructing the coreset takes additional time, the running time between variants remained comparable. We expect to observe significant speedups for large datasets using an optimized implementation of the coreset algorithms. Using *off the shelf* coresets provides a large advantage to this algorithm’s feasibility compared to the iterative methods when handling large datasets.

7 Conclusion

In this paper, we introduced the first $(1 + \varepsilon)$ -approximation algorithms for binary matrix factorization with a singly exponential dependence on the low-rank factor k , which is often a small parameter. We consider optimization with respect to the Frobenius loss, finite fields, and L_p loss. Our algorithms extend naturally to big data models and perform well in practice. Indeed, we conduct empirical evaluations demonstrating the practical effectiveness of our algorithms. For future research, we leave open the question for $(1 + \varepsilon)$ -approximation algorithms for L_p loss without bicriteria requirements.

References

- [BBB⁺19] Frank Ban, Vijay Bhattiprolu, Karl Bringmann, Pavel Kolev, Euiwoong Lee, and David P. Woodruff. A PTAS for ℓ_p -low rank approximation. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 747–766, 2019. [2](#), [3](#), [4](#), [7](#), [8](#), [16](#)
- [BDB13] J Paul Brooks, José H Dulá, and Edward L Boone. A pure l_1 -norm principal component analysis. *Computational statistics & data analysis*, 61:83–98, 2013. [3](#), [8](#)
- [BFL⁺21] Vladimir Braverman, Dan Feldman, Harry Lang, Adiel Statman, and Samson Zhou. Efficient coresets constructions via sensitivity sampling. In *Asian Conference on Machine Learning, ACML*, pages 948–963, 2021. [29](#)
- [BKW17] Karl Bringmann, Pavel Kolev, and David P. Woodruff. Approximation algorithms for l_0 -low rank approximation. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems*, pages 6648–6659, 2017. [8](#)
- [BLK17] Olivier Bachem, Mario Lucic, and Andreas Krause. Practical coresets constructions for machine learning. *arXiv preprint arXiv:1703.06476*, 2017. [29](#)
- [BLK18] Olivier Bachem, Mario Lucic, and Andreas Krause. Scalable k-means clustering via lightweight coresets. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1119–1127, 2018. [30](#)
- [BV10] Radim Belohlávek and Vilém Vychodil. Discovery of optimal factors in binary data via a novel method of matrix decomposition. *J. Comput. Syst. Sci.*, 76(1):3–20, 2010. [4](#)
- [BWZ19] Frank Ban, David P. Woodruff, and Qiuyi (Richard) Zhang. Regularized weighted low rank approximation. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems, NeurIPS*, pages 4061–4071, 2019. [5](#), [7](#)
- [CHHK14] Parinya Chalermsook, Sandy Heydrich, Eugenia Holm, and Andreas Karrenbauer. Nearly tight approximability results for minimum biclique cover and partition. In *Algorithms - ESA 2014 - 22th Annual European Symposium, Proceedings*, volume 8737, pages 235–246, 2014. [7](#)

- [CIK16] L. Sunil Chandran, Davis Issac, and Andreas Karrenbauer. On the parameterized complexity of biclique cover and partition. In *11th International Symposium on Parameterized and Exact Computation, IPEC*, pages 11:1–11:13, 2016. [4](#), [7](#), [9](#)
- [CK19] Vincent Cohen-Addad and Karthik C. S. Inapproximability of clustering in lp metrics. In *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 519–539, 2019. [5](#)
- [CP19] Xue Chen and Eric Price. Active regression via linear-sample sparsification. In *Conference on Learning Theory, COLT*, pages 663–695, 2019. [7](#)
- [CSS21] Vincent Cohen-Addad, David Saulpic, and Chris Schwiegelshohn. A new coresets framework for clustering. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 169–182. ACM, 2021. [19](#)
- [CSTZ22] Sitan Chen, Zhao Song, Runzhou Tao, and Ruizhe Zhang. Symmetric sparse boolean matrix factorization and applications. In *13th Innovations in Theoretical Computer Science Conference, ITCS*, pages 46:1–46:25, 2022. [4](#), [8](#)
- [CW13] Kenneth L. Clarkson and David P. Woodruff. Low rank approximation and regression in input sparsity time. In *Symposium on Theory of Computing Conference, STOC*, pages 81–90, 2013. [11](#), [12](#)
- [DG17] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. [2](#), [27](#)
- [DHJ⁺18] Chen Dan, Kristoffer Arnsfelt Hansen, He Jiang, Liwei Wang, and Yuchen Zhou. Low rank approximation of binary matrices: Column subset selection and generalizations. In *43rd International Symposium on Mathematical Foundations of Computer Science, MFCS*, pages 41:1–41:16, 2018. [3](#), [8](#)
- [DMM06a] Petros Drineas, Michael W. Mahoney, and S. Muthukrishnan. Subspace sampling and relative-error matrix approximation: Column-based methods. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 9th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX and 10th International Workshop on Randomization and Computation, RANDOM, Proceedings*, pages 316–326, 2006. [10](#)
- [DMM06b] Petros Drineas, Michael W. Mahoney, and S. Muthukrishnan. Subspace sampling and relative-error matrix approximation: Column-row-based methods. In *Algorithms - ESA 2006, 14th Annual European Symposium, Proceedings*, pages 304–314, 2006. [10](#)
- [FGL⁺20] Fedor V. Fomin, Petr A. Golovach, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Approximation schemes for low-rank binary matrix approximation problems. *ACM Trans. Algorithms*, 16(1):12:1–12:39, 2020. [2](#), [3](#), [4](#), [5](#), [7](#), [8](#)
- [FJS10] Yinghua Fu, Nianping Jiang, and Hong Sun. Binary matrix factorization and consensus algorithms. In *2010 International Conference on Electrical and Control Engineering*, pages 4563–4567. IEEE, 2010. [8](#)

- [FMPS09] Herbert Fleischner, Egbert Mujuni, Daniël Paulusma, and Stefan Szeider. Covering graphs with few complete bipartite subgraphs. *Theor. Comput. Sci.*, 410(21-23):2045–2053, 2009. 7
- [FSS20] Dan Feldman, Melanie Schmidt, and Christian Sohler. Turning big data into tiny data: Constant-size coresets for k-means, pca, and projective clustering. *SIAM J. Comput.*, 49(3):601–657, 2020. 14
- [GGYT12] Harold W. Gutch, Peter Gruber, Arie Yeredor, and Fabian J. Theis. ICA over finite fields - separability and algorithms. *Signal Process.*, 92(8):1796–1808, 2012. 3
- [GV18] Nicolas Gillis and Stephen A. Vavasis. On the complexity of robust PCA and ℓ_1 -norm low-rank matrix approximation. *Math. Oper. Res.*, 43(4):1072–1084, 2018. 8
- [JPHY14] Peng Jiang, Jiming Peng, Michael Heath, and Rui Yang. A clustering approach to constrained binary matrix factorization. In *Data Mining and Knowledge Discovery for Big Data*, pages 281–303. Springer, 2014. 3, 8
- [KG03] Mehmet Koyutürk and Ananth Grama. PROXIMUS: a framework for analyzing very high dimensional discrete-attributed datasets. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 147–156, 2003. 3, 8
- [KK03] Qifa Ke and Takeo Kanade. Robust subspace computation using l1 norm, 2003. 3, 8
- [KK05] Qifa Ke and Takeo Kanade. Robust l_1 norm factorization in the presence of outliers and missing data by alternative convex programming. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 739–746, 2005. 3, 8
- [KMN⁺04] Tapas Kanungo, David M. Mount, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman, and Angela Y. Wu. A local search approximation algorithm for k-means clustering. *Comput. Geom.*, 28(2-3):89–112, 2004. 5
- [KPRW19] Ravi Kumar, Rina Panigrahy, Ali Rahimi, and David P. Woodruff. Faster algorithms for binary matrix factorization. In *Proceedings of the 36th International Conference on Machine Learning, ICML*, pages 3551–3559, 2019. 2, 3, 4, 5, 7, 8, 9, 25, 26, 27
- [KV09] Ravi Kannan and Santosh S. Vempala. Spectral algorithms. *Found. Trends Theor. Comput. Sci.*, 4(3-4):157–288, 2009. 7
- [Kwa08] Nojun Kwak. Principal component analysis based on l1-norm maximization. *IEEE transactions on pattern analysis and machine intelligence*, 30(9):1672–1680, 2008. 3, 8
- [LSW17] Euiwoong Lee, Melanie Schmidt, and John Wright. Improved and simplified inapproximability for k-means. *Inf. Process. Lett.*, 120:40–43, 2017. 5
- [LVAH12] Haibing Lu, Jaideep Vaidya, Vijayalakshmi Atluri, and Yuan Hong. Constraint-aware role mining via extended boolean matrix decomposition. *IEEE Trans. Dependable Secur. Comput.*, 9(5):655–669, 2012. 4

- [Mag10] Malik Magdon-Ismail. Row sampling for matrix algorithms via a non-commutative bernstein bound. *CoRR*, abs/1008.0587, 2010. 10
- [Mah11] Michael W. Mahoney. Randomized algorithms for matrices and data. *Found. Trends Mach. Learn.*, 3(2):123–224, 2011. 7
- [MKP14] Panos P. Markopoulos, George N. Karystinos, and Dimitrios A. Pados. Optimal algorithms for l_1 -subspace signal processing. *IEEE Trans. Signal Process.*, 62(19):5046–5058, 2014. 3, 8
- [MMG⁺08] 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. 4
- [MMM⁺22] Raphael A. Meyer, Cameron Musco, Christopher Musco, David P. Woodruff, and Samson Zhou. Fast regression for structured inputs. In *The Tenth International Conference on Learning Representations, ICLR, 2022*. 7
- [MMM⁺23] Raphael A. Meyer, Cameron Musco, Christopher Musco, David P. Woodruff, and Samson Zhou. Near-linear sample complexity for l_p polynomial regression. In *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 3959–4025, 2023. 7
- [MMWY22] Cameron Musco, Christopher Musco, David P. Woodruff, and Taisuke Yasuda. Active linear regression for ℓ_p norms and beyond. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 744–753, 2022. 7
- [MW21] Arvind V. Mahankali and David P. Woodruff. Optimal ℓ_1 column subset selection and a fast PTAS for low rank approximation. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 560–578, 2021. 3, 7, 8
- [Neu18] Stefan Neumann. Bipartite stochastic block models with tiny clusters. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems, NeurIPS*, pages 3871–3881, 2018. 7
- [Or177] James Orlin. Contentment in graph theory: covering graphs with cliques. *Indagationes Mathematicae (Proceedings)*, 80(5):406–424, 1977. 7
- [PK18] Young Woong Park and Diego Klabjan. Three iteratively reweighted least squares algorithms for l_1 -norm principal component analysis. *Knowledge and Information Systems*, 54(3):541–565, 2018. 3, 8
- [PPP21] Aditya Parulekar, Advait Parulekar, and Eric Price. L1 regression with lewis weights subsampling. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM*, pages 49:1–49:21, 2021. 7
- [PRF15] Amichai Painsky, Saharon Rosset, and Meir Feder. Generalized independent component analysis over finite alphabets. *IEEE Transactions on Information Theory*, 62(2):1038–1053, 2015. 3

- [PRF18] Amichai Painsky, Saharon Rosset, and Meir Feder. Linear independent component analysis over finite fields: Algorithms and bounds. *IEEE Transactions on Signal Processing*, 66(22):5875–5886, 2018. 3
- [PVG⁺11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 26
- [RPG16] Siamak Ravanbakhsh, Barnabás Póczos, and Russell Greiner. Boolean matrix factorization and noisy completion via message passing. In *Proceedings of the 33rd International Conference on Machine Learning, ICML*, pages 945–954, 2016. 7, 25
- [RSW16] Ilya P. Razenshteyn, Zhao Song, and David P. Woodruff. Weighted low rank approximations with provable guarantees. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC*, pages 250–263, 2016. 5, 7
- [SBM03] Jouni K. Seppänen, Ella Bingham, and Heikki Mannila. A simple algorithm for topic identification in 0-1 data. In *Knowledge Discovery in Databases: PKDD 2003, 7th European Conference on Principles and Practice of Knowledge Discovery in Databases, Proceedings*, pages 423–434, 2003. 4
- [SH06] Tomás Singliar and Milos Hauskrecht. Noisy-or component analysis and its application to link analysis. *J. Mach. Learn. Res.*, 7:2189–2213, 2006. 4
- [SJY09] Bao-Hong Shen, Shuiwang Ji, and Jieping Ye. Mining discrete patterns via binary matrix factorization. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 757–766, 2009. 3, 8
- [SWZ17] Zhao Song, David P. Woodruff, and Peilin Zhong. Low rank approximation with entrywise ℓ_1 -norm error. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC*, pages 688–701, 2017. 3, 7, 8
- [VAG07] Jaideep Vaidya, Vijayalakshmi Atluri, and Qi Guo. The role mining problem: finding a minimal descriptive set of roles. In *Proceedings of the 12th ACM symposium on Access control models and technologies*, pages 175–184, 2007. 4
- [Woo14] David P. Woodruff. Sketching as a tool for numerical linear algebra. *Found. Trends Theor. Comput. Sci.*, 10(1-2):1–157, 2014. 7, 10
- [Yer11] Arie Yeredor. Independent component analysis over galois fields of prime order. *IEEE Trans. Inf. Theory*, 57(8):5342–5359, 2011. 3
- [ZLDZ07] 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. IEEE, 2007. 25

- [ZLS⁺12] Yinqiang Zheng, Guangcan Liu, Shigeki Sugimoto, Shuicheng Yan, and Masatoshi Okutomi. Practical low-rank matrix approximation under robust l1-norm. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1410–1417, 2012. [3](#), [8](#)
- [ZZ12] Marinka Zitnik and Blaz Zupan. Nimfa: A python library for nonnegative matrix factorization. *Journal of Machine Learning Research*, 13:849–853, 2012. [25](#)