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

Ameya Velingker [1] **Maximilian Vötsch** [2] **David P. Woodruff** [3] **Samson Zhou** [4]

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 $\min_{\mathbf{U},\mathbf{V}} \|\mathbf{UV} - \mathbf{A}\|$. 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 $(n+d)k$ 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 $\mathbf{Ax} \approx \mathbf{UVx}$ in time $(n+d)k$, while computing $\mathbf{Ax}$ 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, (Kumar et al., 2019) observed that nearly half of the data sets in the UCI repository (Dua & Graff, 2017) 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 $\mathbf{A} \approx \mathbf{UV}$. The BMF problem restricts the general low-rank approximation problem to a discrete space, making finding good factors more challenging.

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.

Binary matrix factorization. We first consider the minimization of the Frobenius norm, defined by $\|\mathbf{A}-\mathbf{UV}\|_F^2 = \sum_{i\in[n]}\sum_{j\in d}|\mathbf{A}_{i,j} - (\mathbf{UV})_{i,j}|^2$, where $[n] := \{1,\ldots,n\}$ 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 BMF 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 $\|\mathbf{A} - \mathbf{U}'\mathbf{V}'\|_F^2 \leq (1+\varepsilon)\min_{\mathbf{U}\in\{0,1\}^{n\times k},\mathbf{V}\in\{0,1\}^{k\times d}}\|\mathbf{A}-\mathbf{UV}\|_F^2$. For $\varepsilon \in (0,1)$,

[1]Google Research, Mountain View, USA [2]Doctoral School Computer Science DoCS and Faculty of Computer Science, University of Vienna, Austria [3]Carnegie Mellon University, Pittsburgh, USA, and Google, Pittsburgh, USA [4]UC Berkeley and Rice University, USA. Correspondence to: Samson Zhou <samsonzhou@gmail.com>, Maximilian Vötsch <max@voets.ch>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

Reference	Approximation	Running Time	Other
(Kumar et al., 2019)	$C \geq 576$	$2^{\tilde{\mathcal{O}}(k^2)}\,\mathrm{poly}(n,d)$	Frobenius loss
(Fomin et al., 2020)	$1+\varepsilon$	$2^{\frac{2^{\mathcal{O}(k)}}{\varepsilon^2}\log^2\frac{1}{\varepsilon}}\,\mathrm{poly}(n,d)$	Frobenius loss
Here	$1+\varepsilon$	$2^{\tilde{\mathcal{O}}(k^2/\varepsilon^4)}\,\mathrm{poly}(n,d)$	Frobenius loss
(Kumar et al., 2019)	$C \geq 122^{2p-2}+2^{p-1}$	$2^{\mathrm{poly}(k)}\,\mathrm{poly}(n,d)$	L_p loss, $p \geq 1$
Here	$1+\varepsilon$	$2^{\mathrm{poly}(k/\varepsilon)}\,\mathrm{poly}(n,d)$	L_p loss, $p \geq 1$, bicriteria
(Fomin et al., 2020)	$1+\varepsilon$	$2^{\frac{2^{\mathcal{O}(k)}}{\varepsilon^2}\log^2\frac{1}{\varepsilon}}\,\mathrm{poly}(n,d)$	Binary field
(Ban et al., 2019a)	$1+\varepsilon$	$2^{\frac{2^{\mathcal{O}(k)}}{\varepsilon^2}\log\frac{1}{\varepsilon}}\,\mathrm{poly}(n,d)$	Binary field
(Kumar et al., 2019)	$C \geq 40001$	$2^{\mathrm{poly}(k)}\,\mathrm{poly}(n,d)$	Binary field, bicriteria
Here	$1+\varepsilon$	$2^{\mathrm{poly}(k/\varepsilon)}\,\mathrm{poly}(n,d)$	Binary field, bicriteria

Table 1. Summary of related work on binary matrix factorization

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

By comparison, (Kumar et al., 2019) gave a C-approximation algorithm for the BMF problem also using running time $2^{\tilde{\mathcal{O}}(k^2)}\,\mathrm{poly}(n,d)$, 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. (Fomin et al., 2020) introduced a $(1+\varepsilon)$-approximation algorithm for the BMF problem with rank-k factors. However, their algorithm uses time doubly exponential in k, specifically $2^{\frac{2^{\mathcal{O}(k)}}{\varepsilon^2}\log^2\frac{1}{\varepsilon}}\,\mathrm{poly}(n,d)$, which (Ban et al., 2019a) later improved to doubly exponential running time $2^{\frac{2^{\mathcal{O}(k)}}{\varepsilon^2}\log\frac{1}{\varepsilon}}\,\mathrm{poly}(n,d)$, 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 the 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 is a form of 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 (Ke & Kanade, 2003; 2005; Kwak, 2008; Zheng et al., 2012; Brooks et al., 2013; Markopoulos et al., 2014; Song et al., 2017; Park & Klabjan, 2018; Ban et al., 2019a; Mahankali & Woodruff, 2021). On the other hand, for $p>2$, low-rank approximation with L_p error places an increasingly higher priority on outliers, i.e., the larger entries of $\mathbf{UV}$.

We present the first $(1+\varepsilon)$-approximation algorithm for BMF 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 $k' = \mathcal{O}\left(\frac{k\log^2 n}{\varepsilon^3}\right)$. For $\varepsilon \in (0,1)$, our algorithm uses $2^{\mathrm{poly}(k/\varepsilon)}\,\mathrm{poly}(n,d)$ running time and for $\varepsilon \geq 1$, our algorithm uses $2^{\mathrm{poly}(k)}\,\mathrm{poly}(n,d)$ running time.

Previous work (Kumar et al., 2019) gave a C-approximation algorithm for this problem, using singly exponential running time $2^{\mathrm{poly}(k)}\,\mathrm{poly}(n,d)$, 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 achieving 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 (Koyutürk & Grama, 2003; Shen et al., 2009; Jiang et al., 2014; Dan et al., 2018) and independent component analysis, especially in the context of signal processing (Yeredor, 2011; Gutch et al., 2012; Painsky et al., 2015; 2018). 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 (Seppänen et al., 2003; Singliar & Hauskrecht, 2006; Vaidya et al., 2007;

Miettinen et al., 2008; Belohlávek & Vychodil, 2010; Lu et al., 2012; Chandran et al., 2016; Chen et al., 2022).

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

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

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 $2^{\mathrm{poly}(k/\varepsilon)}\,\mathrm{poly}(n,d)$ and for $\varepsilon \geq 1$, our algorithm has running time $2^{\mathrm{poly}(k)}\,\mathrm{poly}(n,d)$.

By comparison, (Kumar et al., 2019) gave a bicriteria C-approximation algorithm for the BMF problem on binary fields with running time $2^{\mathrm{poly}(k)}\,\mathrm{poly}(n,d)$, 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, (Fomin et al., 2020) achieved a $(1+\varepsilon)$-approximation without a bicriteria guarantee, but their algorithm uses doubly exponential running time $2^{\frac{2^{\mathcal{O}(k)}}{\varepsilon^2}\log^2\frac{1}{\varepsilon}}\,\mathrm{poly}(n,d)$, which (Ban et al., 2019a) later improved to doubly exponential running time $2^{\frac{2^{\mathcal{O}(k)}}{\varepsilon^2}\log\frac{1}{\varepsilon}}\,\mathrm{poly}(n,d)$, 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 F 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 $\|\mathbf{U}^*\mathbf{V}^* - \mathbf{A}\|_F^2 = \min_{\mathbf{U}\in\{0,1\}^{n\times k},\mathbf{V}\in\{0,1\}^{k\times d}} \|\mathbf{U}\mathbf{V} - \mathbf{A}\|_F^2$. 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 $\mathbf{V} \in \{0,1\}^{k\times d}$,$(1-\varepsilon)\|\mathbf{U}^*\mathbf{V}-\mathbf{A}\|_F^2 \leq \|\mathbf{S}\mathbf{U}^*\mathbf{V}-\mathbf{S}\mathbf{A}\|_F^2 \leq (1+\varepsilon)\|\mathbf{U}^*\mathbf{V}-\mathbf{A}\|_F^2$. If $\mathbf{S}$ is an affine embedding, 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 the above for $\mathbf{V}^*$, it suffices to solve $\mathrm{argmin}_{\mathbf{V}\in\{0,1\}^{k\times d}} \|\mathbf{S}\mathbf{U}^*\mathbf{V} - \mathbf{S}\mathbf{A}\|_F^2$.

Guessing the sketch matrix S. A general approach taken by (Razenshteyn et al., 2016; Kumar et al., 2019; Ban et al., 2019b) 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 (Razenshteyn et al., 2016; Kumar et al., 2019; Ban et al., 2019b) 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, (Kumar et al., 2019) first find a 2^k-means clustering solution $\widetilde{\mathbf{A}}$ for the rows of $\mathbf{A}$. Instead of solving the problem on $\mathbf{A}$, they then solve BMF on the matrix $\widetilde{\mathbf{A}}$, 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 $\widetilde{\mathbf{A}}$ has 2^k unique rows, they can make all possible guesses for both $\mathbf{S}\mathbf{U}^*$ and $\mathbf{S}\widetilde{\mathbf{A}}$ in time $2^{\tilde{\mathcal{O}}(k^2)}$. By using an algorithm of (Kanungo et al., 2004) that achieves roughly a 9-approximation to k-means clustering, (Kumar et al., 2019) ultimately obtain a C-approximation to the BMF problem, for some $C \geq 576$.

Shortcomings of previous work for $(1+\varepsilon)$-approximation. While (Kumar et al., 2019) 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 (Kanungo et al., 2004), (achieving roughly a 9-approximation) which due to hardness-of-approximation results (Cohen-Addad & Karthik C. S., 2019; Lee et al., 2017) 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 $\mathbf{UV}$ 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, (Kumar et al., 2019) achieves a matrix $\mathbf{S}$ that roughly preserves $\mathbf{SU}^*$ and $\mathbf{SA}$. By generalizations of the triangle inequality, one can show that $\|\mathbf{SU}^*\mathbf{V}^* - \mathbf{SA}\|_F^2$ preserves a constant factor approximation to $\|\mathbf{U}^*\mathbf{V}^* - \mathbf{A}\|_F^2$, but not necessarily a $(1+\varepsilon)$-approximation.

Another related work, (Fomin et al., 2020), 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 coreset. We first reduce the number of unique rows in $\mathbf{A}$ by computing a strong coreset $\widetilde{\mathbf{A}}$ 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 $(1-\varepsilon)\|\mathbf{UV}-\mathbf{A}\|_F^2 \leq \|\mathbf{XV} - \widetilde{\mathbf{A}}\|_F^2 \leq (1+\varepsilon)\|\mathbf{UV} - \mathbf{A}\|_F^2$. Therefore, we first solve the low-rank approximation problem on $\widetilde{\mathbf{A}}$. Crucially, we choose $\widetilde{\mathbf{A}}$ to have $2^{\mathrm{poly}(k/\varepsilon)}$ unique rows so then for a matrix $\mathbf{S}$ that samples $\mathrm{poly}(k/\varepsilon)$ rows, there are $2^{\mathrm{poly}(k/\varepsilon)}$ possibilities for $\mathbf{S}\widetilde{\mathbf{A}}$, so we can make all possible guesses for both $\mathbf{SU}^*$ and $\mathbf{S}\widetilde{\mathbf{A}}$. Unfortunately, we still have the problem that $\|\mathbf{SU}^*\mathbf{V}^* - \mathbf{S}\widetilde{\mathbf{A}}\|_F^2$ does not even necessarily give a $(1+\varepsilon)$-approximation to $\|\mathbf{U}^*\mathbf{V}^* - \widetilde{\mathbf{A}}\|_F^2$.

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 $\|\mathbf{SU}^*\mathbf{V}^* - \mathbf{S}\widetilde{\mathbf{A}}\|_F^2$ produces a $(1+\varepsilon)$-approximation to the cost of the optimal factors $\|\mathbf{U}^*\mathbf{V}^* - \widetilde{\mathbf{A}}\|_F^2$. Thus, we can then solve $\mathbf{V}' = \mathrm{argmin}_{\mathbf{V}\in\{0,1\}^{k\times d}} \|\mathbf{SU}^*\mathbf{V} - \mathbf{S}\widetilde{\mathbf{A}}\|_F^2$ and $\mathbf{U}' = \mathrm{argmin}_{\mathbf{U}\in\{0,1\}^{n\times k}} \|\mathbf{UV}' - \mathbf{A}\|_F^2$, where the latter optimization problem can be solved by iteratively optimizing over each row so that the total computation time is $\mathcal{O}\left(2^k n\right)$ rather than 2^{kn}.

BMF on binary fields. We again form the matrix $\widetilde{\mathbf{A}}$ by taking a strong coreset of $\mathbf{A}$, constructed using an algorithm that gives integer weights w_i to each point, and then duplicating the rows to form $\widetilde{\mathbf{A}}$. That is, if the i-th row $\mathbf{A}_i$ of $\mathbf{A}$ is sampled with weight w_i in the coreset, then $\widetilde{\mathbf{A}}$ will contain w_i repetitions of row $\mathbf{A}_i$. We want to use the same approach for binary fields to make guesses for $\mathbf{SU}^*$ and $\mathbf{S}\widetilde{\mathbf{A}}$. 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 $\widetilde{\mathbf{A}}$ by their number of repetitions, so that group $\mathbf{G}_j$ consists of the rows of $\widetilde{\mathbf{A}}$ that are repeated $[(1+\varepsilon)^j, (1+\varepsilon)^{j+1})$ times. That is, if $\mathbf{A}_i$ appears w_i times in $\widetilde{\mathbf{A}}$, 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 the 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

$$\widetilde{\mathbf{U}} = \left[\widetilde{\mathbf{U}^{(0)}} | \ldots | \widetilde{\mathbf{U}^{(\ell)}}\right], \qquad \widetilde{\mathbf{V}} \leftarrow \widetilde{\mathbf{V}^{(0)}} \circ \ldots \circ \widetilde{\mathbf{V}^{(i)}},$$

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

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. We note that a coreset construction for clustering under L_p metrics rather than Euclidean distance is known, which we can use to construct $\widetilde{\mathbf{A}}$. 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 (Chen & Price, 2019; Parulekar et al., 2021; Musco et al., 2022; Meyer et al., 2022; 2023) can provide such a tool. Unfortunately, adapting these techniques still requires taking a union bound over a number of columns, which would result in the sampling matrix having too many rows for our desired running time.

Instead, we invoke the coreset construction on the rows and the columns so that $\widetilde{\mathbf{A}}$ has a small number of distinct rows and columns. We again partition the rows of $\widetilde{\mathbf{A}}$ 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 towards 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 Applications

This section gives a brief overview of related works and applications of BMF. For a more thorough discussion, including related works, please see Section A

Low-rank approximation is a fundamental and well-studied problem in machine learning and data science. See the surveys (Kannan & Vempala, 2009; Mahoney, 2011; Woodruff, 2014). For real matrices and Frobenius norm loss, low-rank approximation can be ideally solved using the SVD. However, no optimal polynomial time algorithm exists for binary matrices unless $P = NP$ (Ban et al., 2019a).

Binary matrix factorization finds applications in graph partitioning (Chandran et al., 2016), where it is related to the bipartite clique partition problem(Orlin, 1977; Fleischner et al., 2009; Chalermsook et al., 2014; Neumann, 2018), low-density parity-check codes (Ravanbakhsh et al., 2016), and optimizing passive OLED displays (Kumar et al., 2019). As previously noted, many real-world datasets are binary or categorical (Kumar et al., 2019). Hence, BMF on binary fields has been studied in the context of dimensionality reduction on high-dimension binary datasets (Koyutürk & Grama, 2003), gene expression (Zhang et al., 2007), and pattern mining binary data (Shen et al., 2009). To this end, many heuristics have been developed for this problem (Koyutürk & Grama, 2003; Shen et al., 2009; Fu et al., 2010; Jiang et al., 2014), due to its NP-hardness (Gillis & Vavasis, 2018; Dan et al., 2018).

By using an L_p loss function instead of the Frobenius loss, the sensitivity of a BMF algorithm to outliers can be controlled, with $p > 2$ increasing the sensitivity, and $p = 1$ corresponding to robust PCA, which is less sensitive to outliers. (Ke & Kanade, 2003; 2005; Kwak, 2008; Zheng et al., 2012; Brooks et al., 2013; Markopoulos et al., 2014; Song et al., 2017; Park & Klabjan, 2018; Ban et al., 2019a; Mahankali & Woodruff, 2021). Additionally, the $p = 1$ regime finds application in graph theory, where it solves the problem of optimally covering a graph G with k bicliques (Kumar et al., 2019).

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 $\|\mathbf{UV} - \mathbf{A}\|_F^2$. 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 $\|\mathbf{U}^*\mathbf{V}^* - \mathbf{A}\|_F^2 = \min_{\mathbf{U}\in\{0,1\}^{n\times k},\mathbf{V}\in\{0,1\}^{k\times d}} \|\mathbf{UV} - \mathbf{A}\|_F^2$. Observe that if we knew matrices $\mathbf{SU}^*$ and $\mathbf{SA}$ so that for all $\mathbf{V} \in \{0,1\}^{k\times d}$, $(1-\varepsilon)\|\mathbf{U}^*\mathbf{V} - \mathbf{A}\|_F^2 \le \|\mathbf{SU}^*\mathbf{V} - \mathbf{SA}\|_F^2 \le (1+\varepsilon)\|\mathbf{U}^*\mathbf{V} - \mathbf{A}\|_F^2$, then we could find a $(1+\varepsilon)$-approximate solution for $\mathbf{V}^*$ by solving the problem $\text{argmin}_{\mathbf{V}\in\{0,1\}^{k\times d}} \|\mathbf{SU}^*\mathbf{V} - \mathbf{SA}\|_F^2$ instead.

We want to make guesses for the matrices $\mathbf{SU}^*$ and $\mathbf{SA}$, but 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 $\mathbf{SU}^*$ could have too many possibilities because, without additional information, there are 2^{nk} possibilities for the matrix $\mathbf{U}^* \in \{0,1\}^{n\times k}$. Instead, we 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, there are at most 2^k distinct possibilities for each row of $\mathbf{U}^*$. On the other hand, $\mathbf{A} \in \{0,1\}^{n\times d}$ may have 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 $\widetilde{\mathbf{A}}$ 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

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

Therefore, we first solve the low-rank approximation problem on $\widetilde{\mathbf{A}}$. Crucially, $\widetilde{\mathbf{A}}$ 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 $\mathbf{S}\widetilde{\mathbf{A}}$, so we can enumerate all of them for both $\mathbf{SU}^*$ and $\mathbf{S}\widetilde{\mathbf{A}}$. We can then solve $\mathbf{V}' = \text{argmin}_{\mathbf{V}\in\{0,1\}^{k\times d}} \|\mathbf{SU}^*\mathbf{V} - \mathbf{S}\widetilde{\mathbf{A}}\|_F^2$ and $\mathbf{U}' = \text{argmin}_{\mathbf{U}\in\{0,1\}^{n\times k}} \|\mathbf{UV}' - \mathbf{A}\|_F^2$, where the latter optimization problem can be solved by iteratively optimizing over each row so that the total computation time is $\mathcal{O}\left(2^k n\right)$ rather than 2^{kn}. We give the full algorithm in Algorithm 2 and the subroutine for optimizing with respect to $\widetilde{\mathbf{A}}$ in Algorithm 1. We give the subroutines for solving for $\mathbf{V}'$ and $\mathbf{U}'$ in Algorithm 6 and Algorithm 7, respectively.

Lemma 2.1. *Suppose $\varepsilon < \frac{1}{10}$. Then with probability at least 0.97, the output of Algorithm 1 satisfies $\|\mathbf{U}'\mathbf{V}' - \widetilde{\mathbf{A}}\|_F^2 \le (1+6\varepsilon)\|\mathbf{U}^*\mathbf{V}^* - \widetilde{\mathbf{A}}\|_F^2$. Moreover, Algorithm 1 uses $2^{\mathcal{O}\left(m^2+m\log t\right)}\text{poly}(N,d)$ running time for $m = \mathcal{O}\left(\frac{k\log k}{\varepsilon^2}\right)$.*

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 $\mathsf{Cost}(X,S,w) := \sum_{x\in X} w(x)\cdot \min_{s\in S}\|x-s\|_2^2$. When the weights w are uniformly unit across all points in X, we simply write $\mathsf{Cost}(X,S) = \mathsf{Cost}(X,S,w)$.

Algorithm 1 Low-rank approximation for matrix $\widetilde{\mathbf{A}}$ with t distinct rows

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

$V \leftarrow \emptyset$
for each guess of $\mathbf{S}\mathbf{U}^*$ and $\mathbf{S}\widetilde{\mathbf{A}}$, 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 $\mathrm{poly}(N)$ **do**
  $V \leftarrow V \cup \mathrm{argmin}_{\mathbf{V}\in\{0,1\}^{k\times d}} \|\mathbf{S}\mathbf{U}^*\mathbf{V} - \mathbf{S}\widetilde{\mathbf{A}}\|_F^2$ ▷Algorithm 6
end for
for each $\mathbf{V} \in V$ **do**
  Let $\mathbf{U}_\mathbf{V} = \mathrm{argmin}_{\mathbf{U}\in\{0,1\}^{N\times k}} \|\mathbf{U}\mathbf{V} - \widetilde{\mathbf{A}}\|_F^2$ ▷Algorithm 7
end for
$\mathbf{V}' \leftarrow \mathrm{argmin}_{\mathbf{V}\in\{0,1\}^{k\times d}} \|\mathbf{S}\mathbf{U}_\mathbf{V}\mathbf{V} - \mathbf{S}\widetilde{\mathbf{A}}\|_F^2$
$\mathbf{U}' \leftarrow \mathbf{U}_{\mathbf{V}'}$
Return $(\mathbf{U}', \mathbf{V}')$

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

Theorem 2.2 (Theorem 36 in (Feldman et al., 2020)). *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 $\mathcal{O}\left(nd^2 + n^2 d + \frac{nkd}{\varepsilon^2} + \frac{nk^2}{\varepsilon^2}\right)$ 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 at most $\mathrm{poly}(n)$.*

By analyzing correctness and the running time of Algorithm 2, we have:

Theorem 2.3. *There exists an algorithm that uses $2^{\tilde{\mathcal{O}}(k^2/\varepsilon^4)}\,\mathrm{poly}(n,d)$ running time 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*

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

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. That is, to make guesses

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

$t \leftarrow \mathcal{O}\left(\frac{2^{3k}k^2}{\varepsilon^4}\right)$ ▷Theorem 2.2 for 2^k-means clustering
Compute a strong coreset C for 2^k-means clustering of $\mathbf{A}$, with size t and total weight $N = \mathrm{poly}(n)$
Let $\widetilde{\mathbf{A}} \in \{0,1\}^{N\times d}$ be the matrix representation of C, where weighted points are duplicated appropriately
Let $(\widetilde{\mathbf{U}}, \widetilde{\mathbf{V}})$ be the output of Algorithm 1 on input $\widetilde{\mathbf{A}}$
$\mathbf{U}' \leftarrow \mathrm{argmin}_{\mathbf{U}\in\{0,1\}^{n\times k}} \|\mathbf{U}\widetilde{\mathbf{V}} - \mathbf{A}\|_F^2$, $\mathbf{V}' \leftarrow \widetilde{\mathbf{V}}$ ▷Algorithm 7
Return $(\mathbf{U}', \mathbf{V}')$

for the matrices $\mathbf{S}\mathbf{U}^*$ and $\mathbf{S}\mathbf{A}$ while ensuring there are not too many possibilities for these matrices. For matrix operations over general integers, we chose $\mathbf{S}$ as a leverage score sampling matrix that samples rows from $\mathbf{U}^*$ and $\mathbf{A}$. We then used the approximate matrix multiplication property in Lemma C.1 and the subspace embedding property in Theorem C.2 to show that $\mathbf{S}$ provides an affine embedding in Theorem C.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 $\widetilde{\mathbf{A}}$ by taking a strong coreset of $\mathbf{A}$ and then duplicating the rows according to their weight w_i to form $\widetilde{\mathbf{A}}$. That is, if the i-th row $\mathbf{A}_i$ of $\mathbf{A}$ is sampled with weight w_i in the coreset, then $\widetilde{\mathbf{A}}$ 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 $\widetilde{\mathbf{A}}$ by w_i,so that $\mathbf{G}_j$ consists of the rows of $\widetilde{\mathbf{A}}$ that are repeated $[(1+\varepsilon)^j, (1+\varepsilon)^{j+1})$ times. Thus if $\mathbf{A}_i$ appears w_i times in $\widetilde{\mathbf{A}}$, 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 the 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 $\mathbf{A}$. Otherwise if $\mathbf{A}_i$ is not in $\mathbf{G}_j$, then we set the i-th row of $\widetilde{\mathbf{U}^{(j)}}$ to be the all zeros row. We then achieve $\mathbf{V}^{(j)}$ by padding accordingly.

Finally, we collect

$$\widetilde{\mathbf{U}} = \left[\widetilde{\mathbf{U}^{(0)}}|\ldots|\widetilde{\mathbf{U}^{(\ell)}}\right], \qquad \widetilde{\mathbf{V}} \leftarrow \widetilde{\mathbf{V}^{(0)}} \circ \ldots \circ \widetilde{\mathbf{V}^{(i)}}$$

to achieve bicriteria low-rank approximations $\widetilde{\mathbf{U}}$ and $\widetilde{\mathbf{V}}$ to $\widetilde{\mathbf{A}}$. 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 $\widetilde{\mathbf{U}}$. We describe this subroutine in Algorithm 8 and we give the full low-rank approximation bicriteria algorithm in Algorithm 3.

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 (Ban et al., 2019a)). *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 $d \cdot n^{\mathrm{poly}(k/\varepsilon)}$ time.*

Algorithm 3 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 $\|\mathbf{U}'\mathbf{V}' - \mathbf{A}\|_F^2 \le (1+\varepsilon)\min_{\mathbf{U}\in\{0,1\}^{n\times k},\mathbf{V}\in\{0,1\}^{k\times d}} \|\mathbf{U}\mathbf{V}-\mathbf{A}\|_F^2$, where all matrix operations are performed in $\mathbb{F}_2$

$\ell \leftarrow \mathcal{O}\left(\frac{\log n}{\varepsilon}\right)$, $t \leftarrow \mathcal{O}\left(\frac{(2^k\ell)^3k^2}{\varepsilon^4}\right)$, $k' \leftarrow \ell k$ ▷Theorem 2.2 for 2^k-means clustering

Compute a strong coreset C for 2^k-means clustering of $\mathbf{A}$, with size t and total weight $N = \mathrm{poly}(n)$

Let $\widetilde{\mathbf{A}} \in \{0,1\}^{N\times d}$ be the matrix representation of C, where weighted points are duplicated appropriately

For $i \in [\ell]$, let $\mathbf{G}^{(i)}$ be the group of rows (removing multiplicity) of $\widetilde{\mathbf{A}}$ with frequency $[(1+\varepsilon)^i, (1+\varepsilon)^{i+1})$

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

$\widetilde{\mathbf{V}} \leftarrow \widetilde{\mathbf{V}^{(0)}} \circ \ldots \circ \widetilde{\mathbf{V}^{(\ell)}}$

Use Algorithm 8 with $\widetilde{\mathbf{V}^{(0)}}, \ldots, \widetilde{\mathbf{V}^{(\ell)}}$ and $\mathbf{A}$ to find $\mathbf{U}', \mathbf{V}' \leftarrow \widetilde{\mathbf{V}}$

Return $(\mathbf{U}', \mathbf{V}')$

By analyzing the correctness and running time of Algorithm 3, we have:

Theorem 3.2. *There exists an algorithm that uses $2^{\mathrm{poly}(k/\varepsilon)}\,\mathrm{poly}(n,d)$ running time 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*

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

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 $\|\mathbf{U}\mathbf{V}-\mathbf{A}\|_p^p$. We would like to use the same approach as in Section 2, where we first compute a weighted matrix $\widetilde{\mathbf{A}}$ from a strong coreset for $\mathbf{A}$, and then we make guesses for the matrices $\mathbf{S}\mathbf{U}^*$ and $\mathbf{S}\mathbf{A}$ and solve for $\min_{\mathbf{V}\in\{0,1\}^{k\times d}} \|\mathbf{S}\mathbf{U}^*\mathbf{V}-\mathbf{S}\mathbf{A}\|_F^2$ while ensuring there are not too many possibilities for the matrices $\mathbf{S}\mathbf{U}^*$ and $\mathbf{S}\mathbf{A}$. 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 (Cohen-Addad et al., 2021)). *Let $X \subset \mathbb{R}^d$ be a subset of n points, $\varepsilon \in (0,1)$ be an accuracy parameter, $p \ge 1$ be a constant, and let*

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

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

We crucially require the affine embedding property that $(1-\varepsilon)\|\mathbf{U}^*\mathbf{V}-\mathbf{A}\|_F^2 \le \|\mathbf{S}\mathbf{U}^*\mathbf{V}-\mathbf{S}\mathbf{A}\|_F^2 \le (1+\varepsilon)\|\mathbf{U}^*\mathbf{V}-\mathbf{A}\|_F^2$, for all $\mathbf{V} \in \{0,1\}^{k\times d}$. Unfortunately, whether an efficient sampling-based affine embedding exists for L_p loss is not known.

Algorithm 4 Low-rank approximation for matrix $\widetilde{\mathbf{A}}$ with t distinct rows and t' distinct columns

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

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

$V \leftarrow \emptyset$

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 $\mathrm{poly}(N)$ **do**

  $V \leftarrow V \cup \mathrm{argmin}_{\mathbf{V}\in\{0,1\}^{k\times D}} \|\mathbf{S}\mathbf{U}^*\mathbf{V}-\mathbf{S}\mathbf{A}\|_p^p$ ▷Algorithm 6

end for

for each $\mathbf{V} \in V$ **do**

  Let $\mathbf{U}_{\mathbf{V}} = \mathrm{argmin}_{\mathbf{U}\in\{0,1\}^{N\times k}} \|\mathbf{U}\mathbf{V}-\mathbf{A}\|_p^p$ ▷Algorithm 7

end for

$\mathbf{V}' \leftarrow \mathrm{argmin}_{\mathbf{V}\in\{0,1\}^{k\times d}} \|\mathbf{S}\mathbf{U}_{\mathbf{V}}\mathbf{V}-\mathbf{S}\mathbf{A}\|_p^p$

$\mathbf{U}' \leftarrow \mathbf{U}_{\mathbf{V}'}$

Return $(\mathbf{U}', \mathbf{V}')$

We first justify the correctness of Algorithm 5.

Algorithm 5 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 $\|\mathbf{U}'\mathbf{V}' - \mathbf{A}\|_p^p \leq (1+\varepsilon)\min_{\mathbf{U}\in\{0,1\}^{n\times k},\mathbf{V}\in\{0,1\}^{k\times d}} \|\mathbf{U}\mathbf{V} - \mathbf{A}\|_p^p$

$t \leftarrow \mathcal{O}\left(\min(\varepsilon^{-2}+\varepsilon^{-p}, k\varepsilon^{-2})\cdot k\log n\right)$ ▷Theorem 4.1

$\ell \leftarrow \mathcal{O}\left(\frac{\log n}{\varepsilon}\right)$, $k' \leftarrow \ell k$

Compute a strong coreset C for 2^k-means clustering of $\mathbf{A}$, with t rows, with weights $N = \text{poly}(n)$

Compute a strong coreset C' for 2^k-means clustering of C, with t rows and columns, with weights $N, D = \text{poly}(n)$

Let $\widetilde{\mathbf{A}} \in \{0,1\}^{N\times D}$ be the matrix representation of C, where weighted points are duplicated appropriately

For $i \in [\ell]$, let $\mathbf{G}^{(i)}$ be the group of rows (removing multiplicity) of $\widetilde{\mathbf{A}}$ with frequency $[(1+\varepsilon)^i, (1+\varepsilon)^{i+1})$

For $i, j \in [\ell]$, let $\mathbf{G}^{(i,j)}$ be the group of columns (removing multiplicity) of $\mathbf{G}^{(i,j)}$ with frequency $[(1+\varepsilon)^j, (1+\varepsilon)^{j+1})$

Compute the low-rank minimizers $(\widetilde{\mathbf{U}^{(i,j)}}, \widetilde{\mathbf{V}^{(i,j)}})$ on input $\mathbf{G}^{(i,j)}$ using Algorithm 4, padded to $\mathbb{R}^{n\times k}$ and $\mathbb{R}^{k\times D}$, respectively

$\widetilde{\mathbf{U}} \leftarrow \left[\widetilde{\mathbf{U}^{(0,0)}}|\widetilde{\mathbf{U}^{(1,0)}}|\ldots|\widetilde{\mathbf{U}^{(\ell,\ell)}}\right]$, $\widetilde{\mathbf{V}} \leftarrow \widetilde{\mathbf{V}^{(0,0)}} \circ \widetilde{\mathbf{V}^{(1,0)}} \ldots \circ \widetilde{\mathbf{V}^{(\ell,\ell)}}$

Use Algorithm 9 with $\widetilde{\mathbf{U}^{(0,0)}}, \widetilde{\mathbf{U}^{(1,0)}} \ldots, \widetilde{\mathbf{U}^{(\ell,\ell)}}$ and C to find $\mathbf{V}'$

Use $\mathbf{V}'$ and $\mathbf{A}$ to find $\mathbf{U}'$, i.e., Algorithm 7 with dimension k' and L_p loss

Return $(\mathbf{U}', \mathbf{V}')$

Lemma 4.2. *With probability at least* 0.95*, Algorithm 5 returns* $\mathbf{U}', \mathbf{V}'$ *such that*

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

We then analyze the running time of Algorithm 5.

Lemma 4.3. *For any constant* $p \geq 1$*, Algorithm 5 uses* $2^{\text{poly}(k/\varepsilon)}\,\text{poly}(n,d)$ *running time.*

By Lemma 4.2 and Lemma 4.3, we thus have:

Theorem 4.4. *For any constant* $p \geq 1$*, there exists an algorithm that uses* $2^{\text{poly}(k/\varepsilon)}\,\text{poly}(n,d)$ *running time 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* $k' = \mathcal{O}\left(\frac{k\log^2 k}{\varepsilon^2}\right)$ *and*

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

5. Experiments

In this section, we evaluate the practical use of our algorithmic ideas against existing algorithms. Running our full algorithm for BMF is too expensive, even for small k, so our algorithm is based on the idea of partially executing an algorithm with strong theoretical guarantees of (Kumar et al., 2019). 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. For a thorough discussion of our experimental results see Section G.

Baseline and algorithm. We compare several algorithms for binary matrix factorization that have implementations available online, namely (Zhang et al., 2007), which has been implemented in the `NIMFA` library (Zitnik & Zupan, 2012), the message passing algorithm (Ravanbakhsh et al., 2016), as well as our implementation of the algorithm used in the experiments of (Kumar et al., 2019). We refer to these algorithms as Zh, MP, and kBMF. We choose the default parameters provided by the implementations and perform matrix operations in the setting specified by the algorithm. We limit the maximum number of iterations so a running time of 20 seconds is not exceeded.

Motivated by (Kumar et al., 2019), 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 (Kumar et al., 2019) do, we solve the minimization problem $\min_{\mathbf{U}\in\{0,1\}^{n\times k}} \|\mathbf{U}\mathbf{V}-\mathbf{A}\|_F^2$ exactly at a cost of 2^k per row, which is affordable for small k. For a qualitative example of how this step can improve the solution quality, see Figure 1. We call this algorithm kBMF+.

Using k-means as the initial step in this manner is well-motivated by the theoretical and experimental results of (Kumar et al., 2019), but does not guarantee a $(1+\varepsilon)$-approximation, which we are not guaranteed as we do not run our full algorithm, to begin with.

We implemented our algorithm and the one of (Kumar et al., 2019) in Python. For solving k-means, we used the implementation of Lloyd's algorithm provided by the `scikit-learn` library (Pedregosa et al., 2011). All experiments were performed on a Linux notebook with a 3.9 GHz 12th generation Intel Core i7 six-core processor with 32 GiB of RAM.

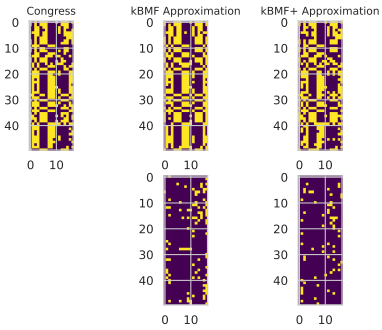

Figure 1. A demonstration of the improved approximation of our algorithm over the algorithm used in the experiments of (Kumar et al., 2019). In the first column, we show the first 50 rows of the congress data set, where purple indicates 0 and yellow indicates 1. The next columns show the approximation of (Kumar et al., 2019), and our algorithm's approximation, both with $k = 10$. The second row shows entries in which the 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 $k = 10$.

Datasets. We use both real and synthetic data for our experiments. We choose two datasets from the UCI Machine Learning Repository (Dua & Graff, 2017), 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 dataset [1], 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 (Kumar et al., 2019).

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, Low-Rank, 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.

[1] https://www.kaggle.com/datasets/emmanuelfwerr/thyroid-disease-data

		Error [Frobenius norm]				Time [ms]			
Dataset	Alg k	kBMF	kBMF+	MP	Zh	kBMF	kBMF+	MP	Zh
Low-Rank	2	75.8	72.2	**71.1**	71.7	13.4	15.5	281.2	11.5
$r = 10$	3	74.3	69.6	69.1	**69.0**	15.8	20.0	308.0	11.7
$p = 0.5$	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	2	30.8	30.5	**27.6**	28.5	10.0	14.3	213.4	5.7
$r = 10$	3	28.5	28.1	**25.2**	25.5	11.1	13.3	248.5	11.5
$p = 0.1$	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

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

Discussion. Our experiments found that the algorithm kBMF+ outperforms other algorithms for the BMF problem on dense synthetic data. Additionally, we found that it 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. Still, 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.

Conclusion. We introduced the first $(1 + \varepsilon)$-approximation algorithms for binary matrix factorization with a singly exponential dependence on k. We optimized the Frobenius loss, finite fields, and L_p loss. Our algorithms work naturally in big data models. Our experiments demonstrate the practicality of our algorithms, particularly for dense low-rank datasets. We leave open the question of $(1 + \varepsilon)$-approximation algorithms for L_p loss without bicriteria requirements.

Acknowledgments

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)". *D. Woodruff:* Work done while the author was at Google Research.

References

Bachem, O., Lucic, M., and Krause, A. Practical coreset constructions for machine learning. *arXiv preprint arXiv:1703.06476*, 2017. 25

Bachem, O., Lucic, M., and Krause, A. Scalable k-means clustering via lightweight coresets. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1119–1127, 2018. 25

Ban, F., Bhattiprolu, V., Bringmann, K., Kolev, P., Lee, E., and Woodruff, D. P. A PTAS for ℓ_p-low rank approximation. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pp. 747–766, 2019a. 2, 3, 5, 7, 13, 14

Ban, F., Woodruff, D. P., and Zhang, Q. R. Regularized weighted low rank approximation. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems, NeurIPS*, pp. 4061–4071, 2019b. 3, 13

Belohlávek, R. and Vychodil, V. Discovery of optimal factors in binary data via a novel method of matrix decomposition. *J. Comput. Syst. Sci.*, 76(1):3–20, 2010. 3

Braverman, V., Feldman, D., Lang, H., Statman, A., and Zhou, S. Efficient coreset constructions via sensitivity sampling. In *Asian Conference on Machine Learning, ACML*, pp. 948–963, 2021. 25

Bringmann, K., Kolev, P., and Woodruff, D. P. Approximation algorithms for l_0-low rank approximation. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems*, pp. 6648–6659, 2017. 13

Brooks, J. P., Dulá, J. H., and Boone, E. L. A pure l1-norm principal component analysis. *Computational statistics & data analysis*, 61:83–98, 2013. 2, 5, 14

Chalermsook, P., Heydrich, S., Holm, E., and Karrenbauer, A. Nearly tight approximability results for minimum biclique cover and partition. In *Algorithms - ESA 2014 - 22th Annual European Symposium, Proceedings*, volume 8737, pp. 235–246, 2014. 5, 13

Chandran, L. S., Issac, D., and Karrenbauer, A. On the parameterized complexity of biclique cover and partition. In *11th International Symposium on Parameterized and Exact Computation, IPEC*, pp. 11:1–11:13, 2016. 3, 5, 13, 14

Chen, S., Song, Z., Tao, R., and Zhang, R. Symmetric sparse boolean matrix factorization and applications. In *13th Innovations in Theoretical Computer Science Conference, ITCS*, pp. 46:1–46:25, 2022. 3, 13

Chen, X. and Price, E. Active regression via linear-sample sparsification. In *Conference on Learning Theory, COLT*, pp. 663–695, 2019. 4

Clarkson, K. L. and Woodruff, D. P. Low rank approximation and regression in input sparsity time. In *Symposium on Theory of Computing Conference, STOC*, pp. 81–90, 2013. 15, 16

Cohen-Addad, V. and Karthik C. S. Inapproximability of clustering in lp metrics. In *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pp. 519–539, 2019. 3

Cohen-Addad, V., Saulpic, D., and Schwiegelshohn, C. A new coreset framework for clustering. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pp. 169–182. ACM, 2021. 7

Dan, C., Hansen, K. A., Jiang, H., Wang, L., and Zhou, Y. Low rank approximation of binary matrices: Column subset selection and generalizations. In *43rd International Symposium on Mathematical Foundations of Computer Science, MFCS*, pp. 41:1–41:16, 2018. 2, 5, 13

Drineas, P., Mahoney, M. W., and Muthukrishnan, S. 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*, pp. 316–326, 2006a. 15

Drineas, P., Mahoney, M. W., and Muthukrishnan, S. Subspace sampling and relative-error matrix approximation: Column-row-based methods. In *Algorithms - ESA 2006, 14th Annual European Symposium, Proceedings*, pp. 304–314, 2006b. 15

Dua, D. and Graff, C. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`. 1, 9

Feldman, D., Schmidt, M., and Sohler, C. Turning big data into tiny data: Constant-size coresets for k-means, pca, and projective clustering. *SIAM J. Comput.*, 49(3):601–657, 2020. 6

Fleischner, H., Mujuni, E., Paulusma, D., and Szeider, S. Covering graphs with few complete bipartite subgraphs. *Theor. Comput. Sci.*, 410(21-23):2045–2053, 2009. 5, 13

Fomin, F. V., Golovach, P. A., Lokshtanov, D., Panolan, F., and Saurabh, S. Approximation schemes for low-rank binary matrix approximation problems. *ACM Trans. Algorithms*, 16(1):12:1–12:39, 2020. 2, 3, 4, 13

Fu, Y., Jiang, N., and Sun, H. Binary matrix factorization and consensus algorithms. In *2010 International Conference on Electrical and Control Engineering*, pp. 4563–4567. IEEE, 2010. 5, 13

Gillis, N. and Vavasis, S. A. On the complexity of robust PCA and ℓ_1-norm low-rank matrix approximation. *Math. Oper. Res.*, 43(4):1072–1084, 2018. 5, 13

Gutch, H. W., Gruber, P., Yeredor, A., and Theis, F. J. ICA over finite fields - separability and algorithms. *Signal Process.*, 92(8):1796–1808, 2012. 2

Jiang, P., Peng, J., Heath, M., and Yang, R. A clustering approach to constrained binary matrix factorization. In *Data Mining and Knowledge Discovery for Big Data*, pp. 281–303. Springer, 2014. 2, 5, 13

Kannan, R. and Vempala, S. S. Spectral algorithms. *Found. Trends Theor. Comput. Sci.*, 4(3-4):157–288, 2009. 5, 13

Kanungo, T., Mount, D. M., Netanyahu, N. S., Piatko, C. D., Silverman, R., and Wu, A. Y. A local search approximation algorithm for k-means clustering. *Comput. Geom.*, 28(2-3):89–112, 2004. 3

Ke, Q. and Kanade, T. Robust subspace computation using l1 norm, 2003. 2, 5, 14

Ke, Q. and Kanade, T. 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)*, pp. 739–746, 2005. 2, 5, 14

Koyutürk, M. and Grama, A. 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*, pp. 147–156, 2003. 2, 5, 13

Kumar, R., Panigrahy, R., Rahimi, A., and Woodruff, D. P. Faster algorithms for binary matrix factorization. In *Proceedings of the 36th International Conference on Machine Learning, ICML*, pp. 3551–3559, 2019. 1, 2, 3, 4, 5, 8, 9, 13, 14, 23

Kwak, N. Principal component analysis based on l1-norm maximization. *IEEE transactions on pattern analysis and machine intelligence*, 30(9):1672–1680, 2008. 2, 5, 14

Lee, E., Schmidt, M., and Wright, J. Improved and simplified inapproximability for k-means. *Inf. Process. Lett.*, 120:40–43, 2017. 3

Lu, H., Vaidya, J., Atluri, V., and Hong, Y. Constraint-aware role mining via extended boolean matrix decomposition. *IEEE Trans. Dependable Secur. Comput.*, 9(5): 655–669, 2012. 3

Magdon-Ismail, M. Row sampling for matrix algorithms via a non-commutative bernstein bound. *CoRR*, abs/1008.0587, 2010. 15

Mahankali, A. V. and Woodruff, D. P. 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*, pp. 560–578, 2021. 2, 5, 13, 14

Mahoney, M. W. Randomized algorithms for matrices and data. *Found. Trends Mach. Learn.*, 3(2):123–224, 2011. 5, 13

Markopoulos, P. P., Karystinos, G. N., and Pados, D. A. Optimal algorithms for l_1-subspace signal processing. *IEEE Trans. Signal Process.*, 62(19):5046–5058, 2014. 2, 5, 14

Meyer, R. A., Musco, C., Musco, C., Woodruff, D. P., and Zhou, S. Fast regression for structured inputs. In *The Tenth International Conference on Learning Representations, ICLR*, 2022. 4

Meyer, R. A., Musco, C., Musco, C., Woodruff, D. P., and Zhou, S. Near-linear sample complexity for l_p polynomial regression. In *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA*, pp. 3959–4025, 2023. 4

Miettinen, P., Mielikäinen, T., Gionis, A., Das, G., and Mannila, H. The discrete basis problem. *IEEE transactions on knowledge and data engineering*, 20(10):1348–1362, 2008. 3

Musco, C., Musco, C., Woodruff, D. P., and Yasuda, T. Active linear regression for ℓ_p norms and beyond. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pp. 744–753, 2022. 4

Neumann, S. Bipartite stochastic block models with tiny clusters. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems, NeurIPS*, pp. 3871–3881, 2018. 5, 13

Orlin, J. Contentment in graph theory: covering graphs with cliques. *Indagationes Mathematicae (Proceedings)*, 80(5):406–424, 1977. 5, 13

Painsky, A., Rosset, S., and Feder, M. Generalized independent component analysis over finite alphabets. *IEEE Transactions on Information Theory*, 62(2):1038–1053, 2015. 2

Painsky, A., Rosset, S., and Feder, M. Linear independent component analysis over finite fields: Algorithms and bounds. *IEEE Transactions on Signal Processing*, 66 (22):5875–5886, 2018. 2

Park, Y. W. and Klabjan, D. Three iteratively reweighted least squares algorithms for l_1-norm principal component analysis. *Knowledge and Information Systems*, 54 (3):541–565, 2018. 2, 5, 14

Parulekar, A., Parulekar, A., and Price, E. L1 regression with lewis weights subsampling. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM*, pp. 49:1–49:21, 2021. 4

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 8

Ravanbakhsh, S., Póczos, B., and Greiner, R. Boolean matrix factorization and noisy completion via message passing. In *Proceedings of the 33nd International Conference on Machine Learning, ICML*, pp. 945–954, 2016. 5, 8, 13

Razenshteyn, I. P., Song, Z., and Woodruff, D. P. Weighted low rank approximations with provable guarantees. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC*, pp. 250–263, 2016. 3, 13

Seppänen, J. K., Bingham, E., and Mannila, H. 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*, pp. 423–434, 2003. 2

Shen, B., Ji, S., and Ye, J. Mining discrete patterns via binary matrix factorization. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 757–766, 2009. 2, 5, 13

Singliar, T. and Hauskrecht, M. Noisy-or component analysis and its application to link analysis. *J. Mach. Learn. Res.*, 7:2189–2213, 2006. 2

Song, Z., Woodruff, D. P., and Zhong, P. Low rank approximation with entrywise ℓ_1-norm error. In Hatami, H., McKenzie, P., and King, V. (eds.), *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC*, pp. 688–701, 2017. 2, 5, 13, 14

Vaidya, J., Atluri, V., and Guo, Q. The role mining problem: finding a minimal descriptive set of roles. In *Proceedings of the 12th ACM symposium on Access control models and technologies*, pp. 175–184, 2007. 2

Woodruff, D. P. Sketching as a tool for numerical linear algebra. *Found. Trends Theor. Comput. Sci.*, 10(1-2):1–157, 2014. 5, 13, 15

Yeredor, A. Independent component analysis over galois fields of prime order. *IEEE Trans. Inf. Theory*, 57(8): 5342–5359, 2011. 2

Zhang, Z., Li, T., Ding, C., and Zhang, X. Binary matrix factorization with applications. In *Seventh IEEE international conference on data mining (ICDM 2007)*, pp. 391–400. IEEE, 2007. 5, 8

Zheng, Y., Liu, G., Sugimoto, S., Yan, S., and Okutomi, M. Practical low-rank matrix approximation under robust l1-norm. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1410–1417, 2012. 2, 5, 14

Zitnik, M. and Zupan, B. Nimfa: A python library for nonnegative matrix factorization. *Journal of Machine Learning Research*, 13:849–853, 2012. 8

A. 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 (Kannan & Vempala, 2009; Mahoney, 2011; Woodruff, 2014). When the underlying loss function is the Frobenius norm, the low-rank approximation problem can be optimally solved via the 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 (Razenshteyn et al., 2016; Song et al., 2017; Kumar et al., 2019; Ban et al., 2019a;b; Fomin et al., 2020; Mahankali & Woodruff, 2021).

Motivation and background for BMF. The BMF problem has applications to graph partitioning (Chandran et al., 2016), low-density parity-check codes (Ravanbakhsh et al., 2016), and optimizing passive organic LED (OLED) displays (Kumar et al., 2019). 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 (Orlin, 1977; Fleischner et al., 2009; Chalermsook et al., 2014; Neumann, 2018), 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.

(Kumar et al., 2019) 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, (Kumar et al., 2019) 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 (Kumar et al., 2019) 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. (Kumar et al., 2019) first gave a constant factor approximation algorithm for the BMF problem using running time $2^{\tilde{\mathcal{O}}(k^2)}\operatorname{poly}(n,d)$, i.e., singly exponential time. (Fomin et al., 2020) introduced a $(1+\varepsilon)$-approximation to the BMF problem with rank-k factors, but their algorithm uses doubly exponential time, specifically running time $2^{\frac{2^{\mathcal{O}(k)}}{\varepsilon^2}\log^2\frac{1}{\varepsilon}}\operatorname{poly}(n,d)$, which was later improved to doubly exponential running time $2^{\frac{2^{\mathcal{O}(k)}}{\varepsilon^2}\log\frac{1}{\varepsilon}}\operatorname{poly}(n,d)$ by (Ban et al., 2019a), who also showed that $2^{k^{\Omega(1)}}$ running time 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 $\mathbf{V}$, (Chen et al., 2022) provide an alternate parametrization of the running time, though, at the cost of running time 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-dimensional datasets (Koyutürk & Grama, 2003). To this end, many heuristics have been developed for this problem (Koyutürk & Grama, 2003; Shen et al., 2009; Fu et al., 2010; Jiang et al., 2014), due to its NP-hardness (Gillis & Vavasis, 2018; Dan et al., 2018).

For the special case of $k = 1$, (Shen et al., 2009) first gave a 2-approximation algorithm that uses polynomial time through a relaxation of integer linear programming. Subsequently, (Jiang et al., 2014) produced a simpler approach, and (Bringmann et al., 2017) introduced a sublinear time algorithm. For general k, (Kumar et al., 2019) gave a constant factor approximation algorithm using running time $2^{\operatorname{poly}(k)}\operatorname{poly}(n,d)$, i.e., singly exponential time, at the expense of a bicriteria solution, i.e., factors with rank $k' = \mathcal{O}(k\log n)$. (Fomin et al., 2020) introduced a $(1+\varepsilon)$-approximation to the BMF problem with rank-k factors, but their algorithm uses doubly exponential time, specifically running time $2^{\frac{2^{\mathcal{O}(k)}}{\varepsilon^2}\log^2\frac{1}{\varepsilon}}\operatorname{poly}(n,d)$, which was later improved to doubly exponential running time $2^{\frac{2^{\mathcal{O}(k)}}{\varepsilon^2}\log\frac{1}{\varepsilon}}\operatorname{poly}(n,d)$ by (Ban et al., 2019a), who also showed that doubly exponential running time 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 (Ke & Kanade, 2003; 2005; Kwak, 2008; Zheng et al., 2012; Brooks et al., 2013; Markopoulos et al., 2014; Song et al., 2017; Park & Klabjan, 2018; Ban et al., 2019a; Mahankali & Woodruff, 2021). The first approximation algorithm with provable guarantees for L_1 low-rank approximation on the reals was given by (Song et al., 2017). They achieved $\mathrm{poly}(k) \cdot \log d$-approximation in roughly $\mathcal{O}(nd)$ time. For constant k, (Song et al., 2017) further achieved constant-factor approximation in polynomial time.

When we restrict the inputs and factors to be binary, (Kumar et al., 2019) 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 $\|\mathbf{UV} - \mathbf{A}\|_1 = |E(G) \triangle E(H)|$. (Chandran et al., 2016) 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 $2^{\mathcal{O}(k^2)}\,\mathrm{poly}(n, d)$ time. (Kumar et al., 2019) recently gave the first constant-factor approximation algorithm for this problem, achieving a C-approximation using $2^{\mathrm{poly}(k)}\,\mathrm{poly}(n, d)$ time, for some constant $C \geq 122^{2p-2} + 2^{p-1}$.

B. Preliminaries

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

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

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

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

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

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 B.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*

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

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 $\tilde{\mathcal{O}}\left(\text{poly}(k, d, \epsilon^{-1})\right)$, 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 B.2 (Leverage score sampling matrix)**.** *(Drineas et al., 2006a;b; Magdon-Ismail, 2010; Woodruff, 2014) 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 B.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 $\mathbf{DT}$, 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.

C. Missing Proofs from Section 2

We first give the subroutines for solving for $\mathbf{V}'$ and $\mathbf{U}'$ in Algorithm 6 and Algorithm 7, respectively.

Algorithm 6 Algorithm for computing optimal $\mathbf{V}$ given $\mathbf{U}$

Input: $\widetilde{\mathbf{A}} \in \{0,1\}^{N \times d}$, $\mathbf{U} \in \{0,1\}^{N \times k}$
Output: $\mathbf{V}' = \text{argmin}_{\mathbf{V} \in \{0,1\}^{k \times d}} \|\mathbf{U}\mathbf{V} - \widetilde{\mathbf{A}}\|_F$
1: **for** $i = 1$ to $i = d$ **do**
2: Set $\mathbf{V}'^{(i)} = \text{argmin}_{\mathbf{V}^{(i)} \in \{0,1\}^{k \times 1}} \|\mathbf{U}\mathbf{V}^{(i)} - \widetilde{\mathbf{A}}^{(i)}\|_2$ ▷Enumerate over all 2^k possible binary vectors
3: **end for**
4: Return $\mathbf{V}' = \left[\mathbf{V}'^{(1)} | \ldots | \mathbf{V}'^{(d)}\right]$

Algorithm 7 Algorithm for computing optimal $\mathbf{U}$ given $\mathbf{V}$

Input: $\widetilde{\mathbf{A}} \in \{0,1\}^{N \times d}$, $\mathbf{V} \in \{0,1\}^{k \times d}$
Output: $\mathbf{U}' = \text{argmin}_{\mathbf{U} \in \{0,1\}^{N \times k}} \|\mathbf{U}\mathbf{V} - \widetilde{\mathbf{A}}\|_F$
1: **for** $i = 1$ to $i = N$ **do**
2: Set $\mathbf{U}'_i = \text{argmin}_{\mathbf{U}_i \in \{0,1\}^{1 \times k}} \|\mathbf{U}_i\mathbf{V} - \widetilde{\mathbf{A}}_i\|_2$ ▷Enumerate over all 2^k possible binary vectors
3: **end for**
4: Return $\mathbf{U}' = \mathbf{U}'_1 \circ \ldots \circ \mathbf{U}'_N$

We recall that leverage score sampling matrices provide approximate matrix multiplication.

Lemma C.1 (Lemma 32 in (Clarkson & Woodruff, 2013))**.** *Let $\mathbf{U} \in \mathbb{R}^{N \times k}$ have orthonormal columns, $\widetilde{\mathbf{A}} \in \{0,1\}^{N \times d}$, and $\mathbf{S} \in \mathbb{R}^{m \times N}$ be a leverage score sampling matrix for $\mathbf{U}$ with $m = \mathcal{O}\left(\frac{1}{\varepsilon^2}\right)$ rows. Then,*

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

We next recall that leverage score sampling matrices give subspace embeddings.

Theorem C.2 (Theorem 42 in (Clarkson & Woodruff, 2013))**.** *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 $\mathbf{V} \in \mathbb{R}^{k \times d}$,*

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

We now recall that approximate matrix multiplication and leverage score sampling suffice to achieve an affine embedding.

Theorem C.3 (Theorem 39 in (Clarkson & Woodruff, 2013)). *Let $\mathbf{U} \in \mathbb{R}^{N\times k}$ have orthonormal columns. Let $\mathbf{S}$ be a sampling matrix that satisfies Lemma C.1 with error parameter $\frac{\varepsilon}{\sqrt{k}}$ and also let $\mathbf{S}$ be a subspace embedding for $\mathbf{U}$ with error parameter ε. Let $\mathbf{V}^* = \operatorname{argmin}_{\mathbf{V}} \|\mathbf{UV} - \widetilde{\mathbf{A}}\|_F$ and $\mathbf{X} = \mathbf{UV}^* - \widetilde{\mathbf{A}}$. Then for all $\mathbf{V} \in \mathbb{R}^{k\times d}$,*

$$(1-2\varepsilon)\|\mathbf{UV} - \widetilde{\mathbf{A}}\|_F^2 - \|\mathbf{X}\|_F^2 \le \|\mathbf{SUV} - \mathbf{S}\widetilde{\mathbf{A}}\|_F^2 - \|\mathbf{SX}\|_F^2 \le (1+2\varepsilon)\|\mathbf{UV} - \widetilde{\mathbf{A}}\|_F^2 - \|\mathbf{X}\|_F^2.$$

We first show that Algorithm 1 achieves a good approximation to the optimal low-rank factors for the coreset $\widetilde{\mathbf{A}}$.

Lemma C.4. *Suppose $\varepsilon < \frac{1}{10}$. Then with probability at least 0.97, the output of Algorithm 1 satisfies*

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

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

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

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

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

Let $\mathbf{X} = \mathbf{U}^*\mathbf{V}^* - \widetilde{\mathbf{A}}$. Note that since $\mathbf{U}^*$ has orthonormal columns, then by Lemma C.1, the leverage score sampling matrix $\mathbf{S}$ achieves approximate matrix multiplication with probability at least 0.99. By Theorem C.2, the matrix $\mathbf{S}$ also is a subspace embedding for $\mathbf{U}$. Thus, $\mathbf{S}$ meets the criteria for applying Theorem C.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 C.3,

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

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

$$\frac{1}{1-2\varepsilon}[\|\mathbf{SU}^*\mathbf{V}'' - \mathbf{S}\widetilde{\mathbf{A}}\|_F^2 - \|\mathbf{SX}\|_F^2 + \|\mathbf{X}\|_F^2] \le \frac{1}{1-2\varepsilon}[\|\mathbf{SU}^*\mathbf{V}^* - \mathbf{S}\widetilde{\mathbf{A}}\|_F^2 - \|\mathbf{SX}\|_F^2 + \|\mathbf{X}\|_F^2].$$

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

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

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 C.3,

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

Since the approximate matrix multiplication property of Lemma C.1, the subspace embedding property of Theorem C.2, and the affine embedding property of Theorem C.3 all fail with probability at most 0.01, then by a union bound, $\mathbf{S}$ succeeds with probability at least 0.97. □

We now analyze the running time of the subroutine Algorithm 1.

Lemma C.5. *Algorithm 1 uses $2^{\mathcal{O}\left(m^2 + m\log t\right)}\operatorname{poly}(N,d)$ running time for $m = \mathcal{O}\left(\frac{k\log k}{\varepsilon^2}\right)$.*

Proof. 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 $\widetilde{\mathbf{A}}$. Thus there are $2^{\mathcal{O}(m \log t)}$ possible matrices $\mathbf{T}$ that selects m rows of $\widetilde{\mathbf{A}}$, for the purposes of leverage score sampling. Assuming the leverage score sampling matrix does not sample any rows with leverage score less than $\frac{1}{\text{poly}(N)}$, 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 $\mathbf{S}\widetilde{\mathbf{A}}$.

For each guess of $\mathbf{S}$ and $\mathbf{S}\widetilde{\mathbf{A}}$, 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 $\text{argmin}_{\mathbf{V} \in \{0,1\}^{k \times d}} \|\mathbf{S}\mathbf{U}^*\mathbf{V} - \mathbf{S}\mathbf{A}\|_F^2$ using $\mathcal{O}\left(2^k d\right)$ time through the subroutine Algorithm 6, which enumerates through all possible 2^k binary vectors for each column. For a fixed $\mathbf{V}$, we can then compute $\mathbf{U}_\mathbf{V} = \text{argmin}_{\mathbf{U} \in \{0,1\}^{N \times k}} \|\mathbf{U}\mathbf{V} - \mathbf{A}\|_F^2$ using $\mathcal{O}\left(2^k N\right)$ time through the subroutine Algorithm 7, which enumerates through all possible 2^k binary vectors for each row of $\mathbf{U}_\mathbf{V}$. Therefore, the total running time of Algorithm 1 is $2^{\mathcal{O}(m^2 + m \log t)} \text{poly}(N, d)$. □

From Lemma C.4 and Lemma C.5, we have:

Lemma 2.1. *Suppose $\varepsilon < \frac{1}{10}$. Then with probability at least 0.97, the output of Algorithm 1 satisfies $\|\mathbf{U}'\mathbf{V}' - \widetilde{\mathbf{A}}\|_F^2 \leq (1 + 6\varepsilon)\|\mathbf{U}^*\mathbf{V}^* - \widetilde{\mathbf{A}}\|_F^2$. Moreover, Algorithm 1 uses $2^{\mathcal{O}(m^2 + m \log t)} \text{poly}(N, d)$ running time for $m = \mathcal{O}\left(\frac{k \log k}{\varepsilon^2}\right)$.*

We now justify the correctness of Algorithm 2.

Lemma C.6. *With probability at least 0.95, Algorithm 2 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 2.2, we have that $\widetilde{\mathbf{A}}$ is a strong coreset, so that

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

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 $\widetilde{\mathbf{A}}$, so that by Lemma C.4,

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

Then by the choice of t in Theorem 2.2, we have that

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

The desired claim then follows from rescaling ε. □

We now analyze the running time of Algorithm 2.

Lemma C.7. *Algorithm 2 uses $2^{\tilde{\mathcal{O}}(k^2/\varepsilon^4)} \text{poly}(n, d)$ running time.*

Proof. By Theorem 2.2, it follows that Algorithm 2 uses $\mathcal{O}\left(nd^2+n^2d+\frac{nkd}{\varepsilon^2}+\frac{nk^2}{\varepsilon^2}\right)$ time to compute $\widetilde{\mathbf{A}} \in \{0,1\}^{N\times d}$ with $N = \mathrm{poly}(n)$. By Lemma C.5, it follows that Algorithm 1 on input $\widetilde{\mathbf{A}}$ thus uses running time $2^{\mathcal{O}(m^2+m\log t)}\,\mathrm{poly}(N,d)$ 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 7 takes $\mathcal{O}\left(2^k n\right)$ time after enumerating through all possible 2^k binary vectors for each row of $\mathbf{U}'$. Therefore, the total running time of Algorithm 2 is $2^{\tilde{\mathcal{O}}\left(\frac{k^2\log^2 k}{\varepsilon^4}\right)}\,\mathrm{poly}(n,d) = 2^{\tilde{\mathcal{O}}(k^2/\varepsilon^4)}\,\mathrm{poly}(n,d)$. □

D. Missing Proofs from Section 3

We first give the standard enumeration subroutine in Algorithm 8.

Algorithm 8 Algorithm for computing optimal $\mathbf{U}$ given $\mathbf{V}^{(1)},\ldots,\mathbf{V}^{(\ell)}$

Input: $\widetilde{\mathbf{A}} \in \{0,1\}^{N\times d}$, $\mathbf{V}^{(1)},\ldots,\mathbf{V}^{(\ell)} \in \{0,1\}^{k\times d}$
Output: $\mathbf{U}' = \mathrm{argmin}_{\mathbf{U}\in\{0,1\}^{N\times \ell k}} \|\mathbf{U}\mathbf{V}-\widetilde{\mathbf{A}}\|_F$, where $\mathbf{U}$ is restricted to one nonzero block of k coordinates
1: **for** $i = 1$ to $i = N$ **do**
2: Set $(\mathbf{U}'_i, j') = \mathrm{argmin}_{\mathbf{U}_i\in\{0,1\}^{1\times k}, j\in[\ell]} \|\mathbf{U}_i\mathbf{V}^{(j)}-\widetilde{\mathbf{A}}_i\|_2$ ▷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: **end for**
5: Return $\mathbf{U}' = \mathbf{U}'_1 \circ \ldots \circ \mathbf{U}'_N$

We first justify the correctness of Algorithm 3.

Lemma D.1. *With probability at least* 0.95*, Algorithm 3 returns* $\mathbf{U}', \mathbf{V}'$ *such that*

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

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

Proof. Let $\widetilde{\mathbf{U}} \leftarrow \left[\widetilde{\mathbf{U}^{(0)}}|\ldots|\widetilde{\mathbf{U}^{(\ell)}}\right]$ in Algorithm 3. 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 $\mathbf{U}'$,

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

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 2.2, it follows that $\widetilde{\mathbf{A}}$ is a strong coreset, and so

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

We decompose the rows of $\widetilde{\mathbf{A}}$ into $\mathbf{G}^{(0)},\ldots,\mathbf{G}^{(\ell)}$ 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 $\widetilde{\mathbf{A}_j}$ is nonzero in $\mathbf{G}_i$. Then we have

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

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

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

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,

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

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 C.4,

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

Then by the choice of t in Theorem 2.2 so that $\widetilde{\mathbf{A}}$ is a strong coreset of $\mathbf{A}$,

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

Therefore, we have

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

and the desired claim then follows from rescaling ε. □

We now analyze the running time of Algorithm 3.

Lemma D.2. *Algorithm 3 uses* $2^{\mathrm{poly}(k/\varepsilon)}\,\mathrm{poly}(n,d)$ *running time.*

Proof. By Theorem 2.2, we have that Algorithm 3 uses $\mathcal{O}\left(nd^2+n^2d+\frac{nkd}{\varepsilon^2}+\frac{nk^2}{\varepsilon^2}\right)$ time to compute $\widetilde{\mathbf{A}}\in\{0,1\}^{N\times d}$ with $N=\mathrm{poly}(n)$. By Lemma 3.1, it takes $d\cdot(2^k)^{\mathrm{poly}(k/eps)}$ 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 $2^{\mathrm{poly}(k/eps)}\,\mathrm{poly}(n,d)$ running time to compute $\widetilde{\mathbf{U}}$ and $\widetilde{\mathbf{V}}$. Finally, computing $\mathbf{U}'$ via Algorithm 8 takes $\mathcal{O}\left(2^{k'}n\right)$ time after enumerating through all possible $2^k\ell$ binary vectors for each row of $\mathbf{U}'$. Therefore, the total running time of Algorithm 2 is $2^{\mathrm{poly}(k/\varepsilon)}\,\mathrm{poly}(n,d)$. □

E. Missing Proofs from Section 4

We first give the standard subroutine for computing optimal $\mathbf{U}$ given $\mathbf{V}^{(1)},\ldots,\mathbf{V}^{(\ell)}$ in Algorithm 9.

Algorithm 9 Algorithm for computing optimal $\mathbf{U}$ given $\mathbf{V}^{(1)},\ldots,\mathbf{V}^{(\ell)}$

Input: $\widetilde{\mathbf{A}}\in\{0,1\}^{N\times d}$, $\mathbf{V}^{(1)},\ldots,\mathbf{V}^{(\ell)}\in\{0,1\}^{k\times d}$
Output: $\mathbf{U}'=\mathrm{argmin}_{\mathbf{U}\in\{0,1\}^{N\times\ell k}}\|\mathbf{U}\mathbf{V}-\widetilde{\mathbf{A}}\|_p^p$, where $\mathbf{U}$ is restricted to one nonzero block of k coordinates

1: **for** $i=1$ to $i=N$ **do**
2: Set $(\mathbf{U}'_i,j')=\mathrm{argmin}_{\mathbf{U}_i\in\{0,1\}^{1\times k},j\in[\ell]}\|(\mathbf{U}_i\mathbf{V}^{(j)}-\widetilde{\mathbf{A}}_i\|_p^p$ ▷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: **end for**
5: Return $\mathbf{U}'=\mathbf{U}'_1\circ\ldots\circ\mathbf{U}'_N$

Lemma E.1. *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* $\mathbf{X}\in\{0,1\}^{k\times r}$,

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

Proof. Let $\mathbf{M} \in \{0, 1, \ldots, k\}^{n\times 1}$ 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

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

Similarly, we have

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

Hence if we repeat take the mean of $\mathcal{O}\left(\frac{k^p}{\varepsilon^2}\right)$ 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 $\mathcal{O}\left(\log \frac{1}{\delta}\right)$ 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 $i \in [r]$. □

We justify the correctness of Algorithm 5.

Lemma 4.2. *With probability at least* 0.95*, Algorithm 5 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 $\widetilde{\mathbf{A}} \in \mathbb{R}^{N\times D}$, so that by the optimality of $\mathbf{U}'$,

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

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, $\widetilde{\mathbf{A}}$ is a strong coreset, and so

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

We decompose the rows of $\widetilde{\mathbf{A}}$ into groups $\mathbf{G}^{(0)}, \ldots, \mathbf{G}^{(\ell)}$ for $\ell = \mathcal{O}\left(\frac{\log n}{\varepsilon}\right)$. For each group $\mathbf{G}^{(i)}$, we decompose the columns of $\mathbf{G}^{(i)}$ into groups $\mathbf{G}^{(i,0)}, \ldots, \mathbf{G}^{(i,\ell)}$ for $\ell = \mathcal{O}\left(\frac{\log n}{\varepsilon}\right)$. 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

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

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

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

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,

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

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

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

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 $\widetilde{\mathbf{A}}$, so that by Lemma C.4,

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

Then by the choice of t in Theorem 4.1 so that $\widetilde{\mathbf{A}}$ is a strong coreset of $\mathbf{A}$,

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

Therefore,

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

and the desired claim then follows from rescaling ε. □

We now analyze the running time of Algorithm 5.

Lemma 4.3. *For any constant $p \ge 1$, Algorithm 5 uses $2^{\mathrm{poly}(k/\varepsilon)}\,\mathrm{poly}(n,d)$ running time.*

Proof. By Theorem 4.1, we have that Algorithm 5 uses $2^{\mathcal{O}(k)}\cdot\mathrm{poly}(n,d)$ time to compute $\widetilde{\mathbf{A}} \in \{0,1\}^{N\times D}$ with $N, D = \mathrm{poly}(n)$. 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 $\mathbf{S}\mathbf{U}^*$ and $\mathbf{S}\mathbf{A}$ in Since $\mathbf{S}\mathbf{U}^*$ and $\mathbf{S}\mathbf{A}$ have $m = \mathcal{O}\left(\frac{k^{p+1}\log r}{\varepsilon^2}\right)$ rows, then there are $\binom{t}{m}$ possible choices for $\mathbf{S}\mathbf{U}^*$ and $\binom{t}{m}$ choices for $\mathbf{S}\mathbf{A}$, where $t = \frac{2^k\log n}{\varepsilon^p}$. Hence, there are $2^{\mathrm{poly}(k/\varepsilon)}\,\mathrm{poly}(n,d)$ possible guesses for $\mathbf{S}\mathbf{U}^*$ and $\mathbf{S}\mathbf{A}$.

For each guess of $\mathbf{S}\mathbf{U}^*$ and $\mathbf{S}\mathbf{A}$, Algorithm 4 iterates through the columns of $\widetilde{\mathbf{V}^{(i,j)}}$, which uses $2^{\mathcal{O}(k)}\cdot\mathrm{poly}(n,d)$ time. Similarly, the computation of $\widetilde{\mathbf{U}^{(i,j)}}$, $\mathbf{U}'$, and $\mathbf{V}'$ all take $2^{\mathcal{O}(k)}\cdot\mathrm{poly}(n,d)$ time. Therefore, the total running time of Algorithm 5 is $2^{\mathrm{poly}(k/\varepsilon)}\,\mathrm{poly}(n,d)$. □

F. 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 $\widetilde{\mathbf{A}}$ for $\mathbf{A}$. We then perform a number of operations on $\widetilde{\mathbf{A}}$ to obtain low-rank factors $\widetilde{\mathbf{U}}$ and $\widetilde{\mathbf{V}}$ for $\widetilde{\mathbf{A}}$. Setting $\mathbf{V}' = \widetilde{\mathbf{V}}$, our algorithms finally use $\mathbf{A}$ and $\mathbf{V}'$ to construct the optimal factor $\mathbf{U}'$ to match $\mathbf{V}'$.

F.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, $\mathbf{A}_1, \ldots, \mathbf{A}_n \in \{0,1\}^d$, 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 $\mathbf{A}_1, \ldots, \mathbf{A}_n \in \{0,1\}^d$.

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 F. The first step is constructing a coreset $\widetilde{\mathbf{A}}$ of $\mathbf{A}$. Whereas our previous coreset constructions were offline, we now require a streaming algorithm to produce the coreset $\widetilde{\mathbf{A}}$. To that end, we use the following well-known merge-and-reduce paradigm for converting an offline coreset construction to a coreset construction in the streaming model.

Theorem F.1. *Suppose there exists an algorithm that, with probability $1 - \frac{1}{\text{poly}(n)}$, produces an offline coreset 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 coreset 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 F.1 to construct a strong coreset C of $\mathbf{A}$ with accuracy $\mathcal{O}(\varepsilon)$. However, C will have $2^{\text{poly}(k)} \cdot \text{poly}\left(\frac{1}{\varepsilon}, \log n\right)$ rows, and thus, we cannot immediately duplicate the rows of C to form $\widetilde{\mathbf{A}}$ because we cannot have $\log n$ dependencies in the number of rows of $\widetilde{\mathbf{A}}$.

After the first pass of the stream, we further apply the respective offline coreset construction, i.e., Theorem 2.2 or Theorem 4.1 to C to obtain a coreset C' with accuracy ε and a number of rows independent of $\log n$. We then use C' to form $\widetilde{\mathbf{A}}$ and perform a number of operations on $\widetilde{\mathbf{A}}$ to obtain low-rank factors $\widetilde{\mathbf{U}}$ and $\widetilde{\mathbf{V}}$ for $\widetilde{\mathbf{A}}$. Setting $\mathbf{V}' = \widetilde{\mathbf{V}}$, 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 $2^{\text{poly}(k)} \cdot d \cdot \text{poly}\left(\frac{1}{\varepsilon}, \log n\right)$ total space in the row-arrival model. For the column-arrival model, the two-pass streaming algorithm uses $2^{\text{poly}(k)} \cdot n \cdot \text{poly}\left(\frac{1}{\varepsilon}, \log d\right)$ total space.

F.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, $\mathbf{A}_1, \ldots, \mathbf{A}_n \in \{0,1\}^d$, 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 F. The first step is constructing a coreset $\widetilde{\mathbf{A}}$ of $\mathbf{A}$. Whereas our previous coreset constructions were offline, we now require a distributed algorithm to produce the coreset $\widetilde{\mathbf{A}}$. To that end, we request that each of the t users send a coreset with accuracy $\mathcal{O}(\varepsilon)$ of their respective rows. Note that each user can construct the coreset 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., $\gamma \cdot 2^{\text{poly}(k)} \cdot \text{poly}\left(\frac{1}{\varepsilon}, \log n\right)$ 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 $\widetilde{\mathbf{A}}$ and performs the required operations on $\widetilde{\mathbf{A}}$ to obtain low-rank factors $\widetilde{\mathbf{U}}$ and $\widetilde{\mathbf{V}}$ for $\widetilde{\mathbf{A}}$.

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 $\tilde{\mathcal{O}}\left(nk + kd\right) \cdot \text{poly}\left(\frac{1}{\varepsilon}\right)$ bits of communication. Hence, the total communication of the protocol is $d\gamma \cdot 2^{\text{poly}(k)} \cdot \text{poly}\left(\frac{1}{\varepsilon}, \log n\right) + \tilde{\mathcal{O}}\left(nk + kd\right) \cdot \text{poly}\left(\frac{1}{\varepsilon}\right)$ in the two-round row-partitioned distributed model. For the two-round column-partitioned distributed model, the total communication of the protocol is $n\gamma \cdot 2^{\text{poly}(k)} \cdot \text{poly}\left(\frac{1}{\varepsilon}, \log d\right) + \tilde{\mathcal{O}}\left(nk + kd\right) \cdot \text{poly}\left(\frac{1}{\varepsilon}\right)$.

G. Description of Experiments

In this section, we discuss a number of additional settings on which we perform our empirical evaluations comparing various algorithms for binary matrix factorization.

Remark G.1. *We additionally noticed a binary matrix factorization algorithm for sparse matrices based on subgradient descent and random sampling*[2] *that is not covered in the literature. This algorithm was excluded from our experiments as it did not produce binary factors in our experiments. 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*.*

G.1. Comparing Algorithms for BMF

Synthetic data. For each algorithm, Table 3 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 running time 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 spare 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 (Kumar et al., 2019). 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 4 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 best and worst factorizations' error differing by up to a factor of about 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 down 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 down tremendously, being almost three orders of magnitude slower than kBMF, but we believe this could be improved with another implementation.

[2] https://github.com/david-cortes/binmf

		Error [Frobenius norm]				Time [ms]			
	Alg	kBMF	kBMF+	MP	Zh	kBMF	kBMF+	MP	Zh
Dataset	k								
Random	2	75.8	72.3	**71.3**	**71.3**	11.2	8.6	280.7	11.6
$p = 0.5$	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	2	36.0	35.0	**34.9**	35.2	10.8	11.3	277.3	9.9
$p = 0.1$	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	2	72.5	67.1	**66.0**	67.8	4.1	7.9	274.9	11.9
$r = 5$	3	69.2	**60.0**	62.3	64.0	12.8	12.0	301.5	13.5
$p = 0.5$	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	2	20.5	20.4	16.5	**15.8**	9.4	6.3	185.6	4.8
$r = 5$	3	17.0	16.6	13.1	**12.0**	5.0	5.8	209.1	12.3
$p = 0.1$	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	2	75.8	72.2	**71.1**	71.7	13.4	15.5	281.2	11.5
$r = 10$	3	74.3	69.6	69.1	**69.0**	15.8	20.0	308.0	11.7
$p = 0.5$	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	2	30.8	30.5	**27.6**	28.5	10.0	14.3	213.4	5.7
$r = 10$	3	28.5	28.1	**25.2**	25.5	11.1	13.3	248.5	11.5
$p = 0.1$	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	2	75.7	72.3	**71.2**	71.3	14.5	18.6	277.6	11.3
$r = 15$	3	74.2	69.9	69.3	**68.7**	12.7	11.1	306.5	11.7
$p = 0.5$	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	2	38.7	38.2	**35.6**	36.5	12.1	10.4	242.2	9.7
$r = 15$	3	37.1	36.2	**33.7**	34.2	10.0	13.0	274.1	12.8
$p = 0.1$	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	2	75.8	72.3	**71.2**	71.6	13.9	12.8	290.4	11.3
$r = 10$	3	74.3	69.6	69.3	**69.0**	13.8	15.6	309.3	11.6
$p = 0.5$	5	72.1	**64.7**	66.2	65.0	17.6	23.8	345.8	13.6
$p_{noise} = 0.001$	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	2	32.5	32.1	**29.3**	30.0	6.3	9.6	223.6	7.6
$r = 10$	3	30.0	29.5	**26.9**	27.1	6.4	10.1	255.4	11.6
$p = 0.1$	5	26.2	24.6	22.0	**21.3**	6.6	9.7	291.9	13.5
$p_{noise} = 0.001$	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	2	75.8	72.1	**71.0**	71.7	9.7	11.4	276.1	11.4
$r = 10$	3	74.3	69.5	**69.0**	69.1	12.1	13.3	302.4	12.0
$p = 0.5$	5	72.0	**64.7**	66.0	64.8	12.4	12.5	338.9	13.4
$p_{noise} = 0.01$	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	2	33.3	33.0	**30.3**	30.9	9.9	11.5	225.3	9.2
$r = 10$	3	31.3	30.8	28.2	**28.0**	10.8	10.5	257.5	12.5
$p = 0.1$	5	27.8	26.2	23.6	**23.4**	9.4	18.3	292.1	14.3
$p_{noise} = 0.01$	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 3. The average running time and error for different Binary Matrix Factorization algorithms on synthetic datasets. The minimum Frobenius norm error is marked in bold.

		Error [Frobenius norm]				Time [ms]			
Dataset	Alg k	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 4. The average running time and error for different Binary Matrix Factorization algorithms on real datasets, minimum Frobenius norm error highlighted in bold.

G.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 (Bachem et al., 2017; Braverman et al., 2021), or a lightweight coreset (Bachem et al., 2018). 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 $r \in \{0.001, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2, \ldots, 0.9\}$. We sample 10 coresets at every size and use them when finding $\mathbf{V}$ in our kBMF+ algorithm. The theory suggests that the quality of the coreset depends only on k and the dimension d of the points, which is why in Figure 2, we observe a worse approximation for a given size of coreset for larger k. We find that the kBMF+ 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 a 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.

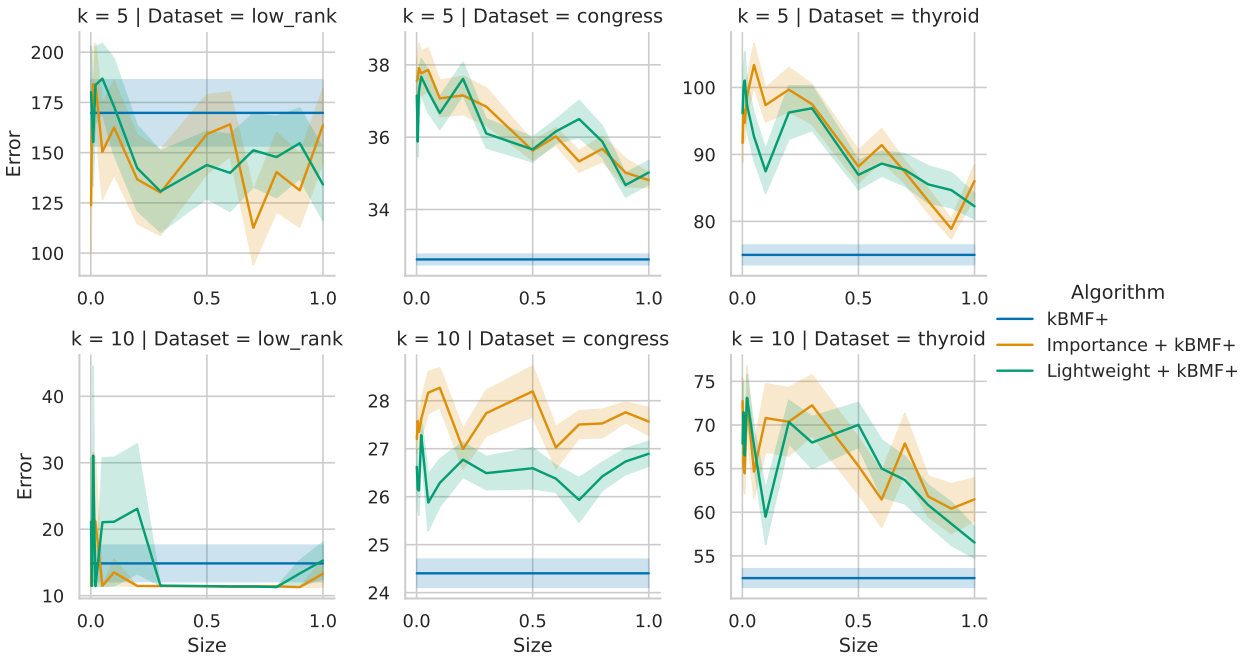

Figure 2. A plot of the effect of different relative coreset sizes on the results of our algorithm.