A Quantum-inspired Classical Algorithm for Separable Non-negative Matrix Factorization

Zhihuai Chen[1,2] , **Yinan Li**[3] , **Xiaoming Sun**[1,2] , **Pei Yuan**[1,2] and **Jialin Zhang**[1,2]
[1]CAS Key Lab of Network Data Science and Technology, Institute of Computing Technology, Chinese Academy of Sciences, 100190, Beijing, China
[2]University of Chinese Academy of Sciences, 100049, Beijing, China
[3]Centrum Wiskunde & Informatica and QuSoft, Science Park 123, 1098XG Amsterdam, Netherlands
{chenzhihuai, sunxiaoming, yuanpei, zhangjialin}@ict.ac.cn, yinan.li@cwi.nl

Abstract

Non-negative Matrix Factorization (NMF) asks to decompose a (entry-wise) non-negative matrix into the product of two smaller-sized nonnegative matrices, which has been shown intractable in general. In order to overcome this issue, separability assumption is introduced which assumes all data points are in a conical hull. This assumption makes NMF tractable and is widely used in text analysis and image processing, but still impractical for huge-scale datasets. In this paper, inspired by recent development on dequantizing techniques, we propose a new classical algorithm for separable NMF problem. Our new algorithm runs in polynomial time in the rank and logarithmic in the size of input matrices, which achieves an exponential speedup in the low-rank setting.

1 Introduction

Non-negative Matrix Factorization (NMF) aims to approximate a non-negative data matrix $A \in \mathbb{R}_{\geq 0}^{m\times n}$ by the product of two non-negative low rank factors, *i.e.*, $A \approx WH^T$, where $W \in \mathbb{R}_{\geq 0}^{m\times k}$ is called basis matrix, $H \in \mathbb{R}_{\geq 0}^{n\times k}$ is called encoding matrix and $k \ll \min\{m, n\}$. In many applications, an NMF often results in more natural and interpretable part-based decomposition of data [Lee and Seung, 1999]. Therefore, NMF has been widely used in a number of practical applications, such as topic modeling in text, signal separation, social network, collaborative filtering, dimension reduction, sparse coding, feature selection and hyperspectral image analysis. Since computing an NMF is NP-hard [Vavasis, 2009], a series of heuristic algorithms have been proposed [Lee and Seung, 2001; Lin, 2007; Hsieh and Dhillon, 2011; Kim and Park, 2008; Ding *et al.*, 2010; Guan *et al.*, 2012]. All of the heuristic algorithms aim to minimize the reconstruction error, the formula which is a non-convex program and lack optimality guarantee:

$$\min_{W\in\mathbb{R}_{\geq 0}^{m\times k} H\in\mathbb{R}_{\geq 0}^{n\times k}} \left\|A - WH^T\right\|_F .$$

A natural assumption on the data called *separability assumption*, was observed in [Donoho and Stodden, 2004] . From a geometry perspective, the separable assumption means that all rows of A reside in a cone generated by a rather smaller number of rows. In particular, these generators are called anchors of A. To solve the Separable Non-Negative Matrix Factorizations (SNMF), it is sufficient to identify the anchors in the input matrices, which can be solved in polynomial time [Arora *et al.*, 2012a; Arora *et al.*, 2012b; Gillis and Vavasis, 2014; Esser *et al.*, 2012; Elhamifar *et al.*, 2012; Zhou *et al.*, 2013; Zhou *et al.*, 2014]. Separability assumption is favored by various practical applications. For example, in the unmixing task in hyperspectral imaging, separability implies the existence of 'pure' pixel [Gillis and Vavasis, 2014]. And in the topic detection task, it also means some words are associated with unique topic [Hofmann, 2017]. In huge datasets, it is useful to pick up some representative data points to stand for other points. Such 'self-expression' assumption helps to improve the data analysis procedure [Mahoney and Drineas, 2009; Elhamifar and Vidal, 2009].

1.1 Related Work

It is natural to assume all the rows of the input A has unit ℓ_1-norm, since ℓ_1-normalization translates the conical hull to convex hull while keeping the anchors unchanged. From this perspective, most algorithms essentially identify the extreme points in the convex hull of the (ℓ_1-normalized) data vectors. In [Arora *et al.*, 2012a], the authors use m linear programs in $O(m)$ variables to identify the anchors out of m data points, and it is therefore not suitable for dealing with large-scale real-world problems. Furthermore, [Recht *et al.*, 2012] presents a single LP in n^2 variables for SNMF to deal with large-scale problems (but is still impractical for huge-scale problems).

There is another class of algorithms based on greedy algorithms. The main idea is to opt a data point on the direction where the current residual decreases fast. The algorithms terminate with a sufficiently small error or a large iteration times. For example, Successful Projection Algorithm (SPA) [Gillis and Vavasis, 2014] derives from Gram-Schmidt orthogonalization with row or column pivoting. XRAY [Kumar *et al.*, 2013] detects a new anchor referring to the residual of exterior data points and updates the residual matrix by solving a non-negative least square regression. Both of these two algorithms based on greedy pursuit have smaller time complexity compared with LP-based methods. However, the time complexity is still too large for large-scaled data.

[Zhou *et al.*, 2013; Zhou *et al.*, 2014] utilize a Divide-and-Conquer Anchoring (DCA) framework to tackle the SNMF. Namely, by projecting the data set into several low-dimension

subspaces, and each projection can determines a small set of anchors. Moreover, it can be proven that all the k anchors can be identified by $O(k \log k)$ projections.

Recently, a quantum algorithm for SNMF called Quantum Divide-and-Conquer Anchoring algorithm (QDCA), has been presented [Du *et al.*, 2018], which uses the quantum technology to speed up the random projection step in [Zhou *et al.*, 2013]. QDCA implements matrix-vector product (*i.e.*, random projection) via quantum principal component analysis and then a quantum state encoding the projected data points could be prepared efficiently. Moreover, there are also several papers utilizing dequantizing techniques to solve some low-rank matrix operations, such as recommendation systems [Tang, 2018] and matrix inversion [Gilyén *et al.*, 2018; Chia *et al.*, 2018]. Dequantizing techniques in those algorithms involve two technologies, the Monte-Carlo singular value decomposition and rejection sampling, which could efficiently simulate some special operations on low-rank matrices.

Inspired by QDCA and the dequantizing techniques , we propose a classical randomized algorithm which speeds up the random projection step in [Zhou *et al.*, 2013] and thereby identifies all anchors efficiently. Our algorithm takes time polynomial in rank k, condition number κ and *logarithm* of the size of matrix. When rank $k = O(\log(mn))$, our algorithm achieves exponentially speedup than any other classical algorithms for SNMF.

2 Preliminaries

2.1 Notations

Let $[n] := \{1, 2, \ldots, n\}$. Let $\text{span}\{x_i \in \mathbb{R}^n | i \in [k]\} := \{\sum_{i=1}^k \alpha_i x_i | \alpha_i \in \mathbb{R}, i \in [k]\}$ denote the space spanned by x_i for $i \in [k]$. For a matrix $A \in \mathbb{R}^{m\times n}$, $A_{(i)}$ and $A^{(j)}$ denote the ith row and the jth column of A for $i \in [m]$, $j \in [n]$, respectively. Let $A_R = [A_{(i_1)}^T, A_{(i_2)}^T, \ldots, A_{(i_r)}^T]^T$ where $A \in \mathbb{R}^{m\times n}$ and $R = \{i_1, i_2, \ldots, i_r\} \subseteq [m]$ (without loss of generality, assume $i_1 \le i_2 \le \cdots \le i_r$). $\|A\|_F$ and $\|A\|_2$ refer to Frobenius norm and spectral norm, respectively. For a vector $v \in \mathbb{R}^n$, $\|v\|$ denotes its ℓ_2-norm. For two probability distributions p, q (as density functions) over a discrete universe D, the *total variation distance* between them is defined as $\left\|p, q\right\|_{TV} := \frac{1}{2}\sum_{i\in D} |p(i) - q(i)|$. $\kappa(A) := \sigma_{\max}/\sigma_{\min}$ denotes the condition number of A, where $\sigma_{\max}$ and $\sigma_{\min}$ are the maximal and minimal *non-zero* singular values of A.

2.2 Sample Model

In query model, algorithms for SNMF problem require time which is at least linear in the number of nonzero elements of the matrix, since in the worst case, they have to read out all entries. However, we expect our algorithm to be efficient even if the datasets are extremely large. Considering the QDCA in [Du *et al.*, 2018], one of its advantage is that data is prepared in quantum state and can be access via 'quantum' way (like sampling). Thus, in quantum algorithm, quantum state is served to represent data implicitly which can be read out by measurement only. In order to avoiding reading the whole matrix, we introduce a new sample model other than the query model based on the idea of quantum state preparation assumption.

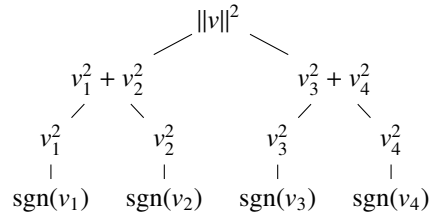

Figure 1: Binary search tree for $v = (v_1, v_2, v_3, v_4)^T \in \mathbb{R}^4$. The leaf node stores v_i^2 and interior node stores the sum of the children. In order to restore the original vector, we also store the sign of v_i in leaf node. To sampling from $\mathcal{D}_v$, we can start from top and randomly recurring on a child, with probability proportional to its weight.

Definition 2.1 (ℓ_2-norm Sampling)**.** *Let $\mathcal{D}_v$ denote the distribution over $[n]$ with density function $\mathcal{D}_v(i) = v_i^2/\|v\|^2$ for $v \in \mathbb{R}^n$. A sample from a distribution $\mathcal{D}_v$ is called a sample from v.*

Lemma 2.2 (Vector Sample Model)**.** *There is a data structure storing vector $v \in \mathbb{R}^n$ in $O(n \log n)$ space, and supporting following operations:*

- *Querying and updating a entry in $O(\log n)$ time;*
- *Sampling from $\mathcal{D}_v$ in $O(\log n)$ time;*
- *Finding $\|v\|$ in $O(1)$ time.*

Such a data structure can be easily implemented via Binary Search Tree (BST) (see Figure 1).

Proposition 2.3 (Matrix Sample Model)**.** *Considering matrix $A \in \mathbb{R}^{m\times n}$, let $\tilde{A}$ and $\tilde{A}'$ be the vector whose entry is $\left\|A_{(i)}\right\|$ and $\left\|A^{(j)}\right\|$, respectively. There is a data structure storing matrix $A \in \mathbb{R}^{m\times n}$ in $O(mn)$ space and supporting following operations:*

- *Querying and updating an entry in $O(\log m + \log n)$ time;*
- *Sampling from $A_{(i)}$ for any $i \in [m]$ in time $O(\log n)$;*
- *Sampling from $A^{(j)}$ for any $j \in [n]$ in time $O(\log m)$;*
- *Finding $\|A\|_F$, $\left\|A_{(i)}\right\|$ and $\left\|A^{(j)}\right\|$ in time $O(1)$;*
- *Sampling $\tilde{A}$ and $\tilde{A}'$ in time $O(\log m)$ and $O(\log n)$, respectively.*

This data structure can be easily implemented via Lemma 2.2, we can just use two arrays of BST to store all rows and columns of A and use two extra BSTs store $\tilde{A}$ and $\tilde{A}'$.

2.3 Low-rank Approximations in Sample Model

FKV algorithm is a Monte-Carlo algorithm [Frieze *et al.*, 2004] that returns approximate singular vectors of given matrix A in *matrix sample model*. The low-rank approximation of A can be reconstructed by approximate singular vectors. The query and sample complexity of FKV algorithm are independent of size of A. FKV algorithm outputs a short 'description' of $\hat{V}$, which is approximate to a right singular vectors V of matrix A. Similarly, FKV algorithm can output a description of approximate left singular vectors $\hat{U}$ of A by inputting A^T. Let FKV (A, k, ϵ, δ) denote the FKV algorithm, where A is a matrix given by sample model, k is the rank of approximate

matrix of A, ϵ is error parameter, and δ is the failure probability. The FKV algorithm is described in Theorem 2.4.

Theorem 2.4 (Low-rank Approximations, [Frieze *et al.*, 2004]). *Given matrix $A \in \mathbb{R}^{m\times n}$ in matrix sample model, $k \in \mathbb{N}$ and $\epsilon, \delta \in (0,1)$,* FKV *algorithm outputs the description of the approximate right singular vectors $\hat{V} \in \mathbb{R}^{n\times k}$ in $O(poly(k, 1/\epsilon, \log\frac{1}{\delta}))$ samples and queries of A with probability $1-\delta$, which satisfies*

$$\left\|A\hat{V}\hat{V}^T - A\right\|_F^2 \leq \min_{D:rank(D)\leq k} \|A-D\|_F^2 + \epsilon\|A\|_F^2 .$$

Especially, if A is a matrix with rank k exactly, Theorem 2.4 also implies an inequality: $\left\|A\hat{V}\hat{V}^T - A\right\|_F \leq \sqrt{\epsilon}\|A\|_F$.

Description of $\hat{V}$. Note that FKV algorithm does not output the approximate right singular vectors $\hat{V}$ directly since their lengths are linear of n. It returns a description of $\hat{V}$, which consists of three components: the row index sets $T := \{i_t \in [m] | t \in [p]\}$, a vector set $U := \{u^{(j)} \in \mathbb{R}^p | j \in [k]\}$ which are singular vectors of a submatrix sampled from A , and its corresponding singular values $\Sigma := \{\sigma^{(j)} | j \in [k]\}$, where $p = O(poly(k, \frac{1}{\epsilon}))$. In fact, $\hat{V}^{(i)} := A_T u^{(i)}/\sigma_i$ for $i \in [k]$. Given a description of $\hat{V}$, we can *sample* from $\hat{V}^{(i)}$ in time $O(poly(k, \frac{1}{\epsilon}))$ for $i \in [k]$ [Tang, 2018] and *query* its entry in time $O(poly(k, \frac{1}{\epsilon}))$.

Definition 2.5 (α-orthonormal). *Given $\alpha > 0$, $\hat{V} \in \mathbb{R}^{n\times k}$ is called α-approximately orthonormal if $1 - \alpha/k \leq \left\|\hat{V}^{(i)}\right\|^2 \leq 1 + \alpha/k$ for $i \in [k]$ and $|\hat{V}^{(s)}\hat{V}^{(t)}| \leq \alpha/k$ for $s \neq t \in [k]$.*

The next lemma presents some properties of α-approximate orthonormal vectors.

Lemma 2.6 (Properties of α-orthonormal Vectors, [Tang, 2018]). *Given a set of k α-approximately orthonormal vectors $\hat{V} \in \mathbb{R}^{n\times k}$, then there exists a set of k orthonormal vectors $V \in \mathbb{R}^{n\times k}$ spanning the columns of $\hat{V}$ such that*

$$\left\|V - \hat{V}\right\|_F \leq \alpha/\sqrt{2} + c_1\alpha^2, \tag{1}$$

$$\left\|\Pi_{\hat{V}} - \hat{V}\hat{V}^T\right\|_F \leq c_2\alpha, \tag{2}$$

where $\Pi_{\hat{V}} := VV^T$ represents the orthonormal projector to image of $\hat{V}$ and $c_1, c_2 > 0$ are constants.

Lemma 2.7 ([Frieze *et al.*, 2004]). *The output vectors $\hat{V} \in \mathbb{R}^{n\times k}$ of* FKV *(A, k, ϵ, δ) is $\epsilon k/16$-approximate orthonormal.*

3 Fast Anchors Seeking Algorithm

In this section, we present a randomized algorithm for SNMF which is called Fast Anchors Seeking (FAS) Algorithm. Especially, the input $A \in \mathbb{R}^{m\times n}_{\geq 0}$ of FAS is given by matrix sample model which is realized via a data structure described in Section 2. FAS returns the indices of anchors in time polynomial logarithmic to the size of matrix.

3.1 Description of Algorithm

Recall that SNMF aims to factorize $A = FA_R$ where R is the index set of anchors. In this paper, an additional constraint is added: the sum of entries in any row of F is 1. Namely, any data point of A resides in convex hull which is the set of all convex combination of A_R. In fact, normalizing each row of matrix A by ℓ_1-norm is valid, since the anchors remain unchanged. Moreover, Instead of storing ℓ_1-normalized matrix A, we can just maintain the ℓ_1-norms for all rows and columns.

The Quantum Divide-and-Conquer Anchoring (QDCA) is a quantum algorithm for SNMF which achieves exponential speedup than any classical algorithms [Du *et al.*, 2018]. After projecting any convex hull into an 1-dimensional space, the geometric information is partially preserved. Especially, the anchors in 1-dimensional projected subspace are still anchors in the original space. The main idea of QDCA is quantizing random projection step in DCA. It decomposes SNMF into several subproblems: projecting A onto a set of random unit vectors $\{\beta_i \in \mathbb{R}^n\}_{i=1}^s$ with $s = O(k\log k)$, *i.e.*, computing $A\beta_i \in \mathbb{R}^m$. Such a matrix-vector product can be efficiently implemented by Quantum Principle Component Analysis (QPCA). And then it returns a $\log m$-qubits quantum state whose amplitudes are proportional to entries of $A\beta_i$. Measurement of quantum state outcomes an index $j \in [m]$ which obeys distribution $\mathcal{D}_{A\beta_i}$. Thus, we can prepare $O(poly\log m)$ copies of quantum states, measure each of them in computational basis and record the most frequent index. By repeating procedure above with $s = O(k\log k)$ times, we could successively identify all anchors with high probability.

As discussed above, the core and most costly procedure is to simulate $\mathcal{D}_{A\beta_i}$. At the first sight, traditional algorithms can not achieve exponential speedup on account of limits of computational model. In QDCA, vectors are encoded into quantum states and we can sample the entries with probability proportional to their magnitudes by measurements. This quantum state preparation overcomes the bottleneck of traditional computational model. Based on divided-and-conquer scheme and sample model (See Section 2.2), we present Fast Anchors Seeking (FAS) Algorithm inspired by QDCA. Designing FAS is quite hard and non-trivial although FAS and QDCA have the same scheme. Indeed, we can simulate $\mathcal{D}_{A\beta_i}$ directly by rejection sampling technology. However, the number of iterations of rejection sampling is unbounded. To overcome this difficulty, we translate matrix A into its approximation $\hat{U}\hat{U}^TA$, where the columns $\hat{U} \in \mathbb{R}^{m\times k}$ consists of k approximate left singular vectors of matrix A and $k = rank(A)$. Next, it is obvious that $y = \hat{U}^TA\beta_i \in \mathbb{R}^k$ is a short vector and we can estimate its entries one by one (see Lemma 3.5) efficiently. Now the problem becomes to simulate $\mathcal{D}_{\hat{U}y}$ and it can be done by Lemma 3.4 .

Given an error parameter $\epsilon/2$, the method described above will result in $\left\|A\beta_i - \hat{U}\hat{U}^TA\beta_i\right\| < \epsilon\|A\|_F\left\|\beta_i\right\|/2$ via Theorem 2.4, which implies $\left\|\mathcal{D}_{A\beta_i} - \mathcal{D}_{\hat{U}\hat{U}^TA\beta_i}\right\|_{TV} \leq \epsilon\|A\|_F\left\|\beta_i\right\|/\left\|A\beta_i\right\|$. Namely, the method above introduces an unbounded error in form $\epsilon\|A\|_F\left\|\beta_i\right\|/\left\|A\beta_i\right\|$ if β_i is arbitrary vector in entire space $\mathbb{R}^n$. Fortunately, this issue can be solved by generating random vectors $\{\beta_i\}_{i=1}^s$ lying in row space of A instead of those lying in entire space $\mathbb{R}^n$. To generate uniform random unit vectors on the row space of A, we need to find a basis of row space of A. If $V \in \mathbb{R}^{n\times k}$ is a set of orthonormal basis of the row space of A (the space spanned by the right singular vectors), and x_i is uniform random unit vector on $\mathbb{S}^{k-1}$, then $\beta = Vx_i$ is a unit random vector in row space of A. Moreover, FKV algorithm will figure out approximate singular vectors $\hat{V}$ for V, that can

$$\mathcal{D}_{A\beta_i} \xleftarrow{(1)\ \beta_i = Vx_i} \mathcal{D}_{AVx_i} \overset{(2)\text{ Lemma 2.6, Lemma 3.2}}{\underset{\text{span}\{\hat{V}^{(i)}\} = \text{span}\{A_{(i)}\}}{\leftsquigarrow}} \mathcal{D}_{A\hat{V}x_i} \overset{(3)\text{ Theorem 2.4}}{\underset{\hat{U}\hat{U}^TA \approx A}{\leftsquigarrow}} \mathcal{D}_{\hat{U}\hat{U}^TA\hat{V}x_i} \overset{(4)\text{ Lemma 3.5}}{\underset{\tilde{M} \approx \hat{U}^TA\hat{V},\ \tilde{y}_i = \tilde{M}x_i}{\leftsquigarrow}} \mathcal{D}_{\hat{U}\tilde{y}_i} \overset{(5)\text{ Lemma 3.4}}{\underset{\text{rejection sampling}}{\leftsquigarrow}} \mathcal{O}_i$$

Figure 2: An illustration for how to approximate distribution $\mathcal{D}_{A\beta_i}$. $\mathcal{O}_i$ represents the final distribution which approximate $\mathcal{D}_{A\beta_i}$. $\leftsquigarrow$ represents 'approximate' and $\leftarrow$ represents 'equal to' in a sense of total variation distance. To prove the upper bound for $\|\mathcal{D}_{A\beta_i}, \mathcal{O}_i\|_{TV}$, we introduce several medium distributions $\mathcal{D}_{A\hat{V}x_i}$, $\mathcal{D}_{\hat{U}\hat{U}^TA\hat{V}x_i}$ and $\mathcal{D}_{\hat{U}\tilde{y}}$. From left to right, (1) use a set of right singular vectors $V \in \mathbb{R}^{n\times k}$ to generate the random unit vector lying in the row space of A; (2) however, since V cannot be gained efficiently, use FKV algorithm to figure out an approximation $\hat{V}$ given by a 'short desription'; (Lemma 2.6, Lemma 3.2) (3) translate $A\hat{V}x_i$ into $\hat{U}\hat{U}^TA\hat{V}x_i$ since $A \approx \hat{U}\hat{U}^TA$ (Theorem 2.4), where $\hat{U}$ is the approximate left singular vectors generated by FKV given by a 'short desription'. (4) return an estimation $\tilde{M}$ of $M = \hat{U}^TA\hat{V} \in \mathbb{R}^{k\times k}$ by estimating each entry by Lemma 3.5 and then approximate Mx_i (denoted as $\tilde{y}_i$); (5) finally, the rejection sampling works for $\hat{U}\tilde{y}_i$ by Lemma 3.4 since $\hat{U}$ is approximately orthonormal.

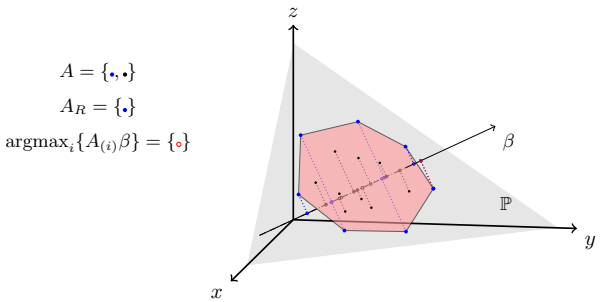

Figure 3: An illustration of finding an anchor of A with rank $k = 3$. The 3-dimensional space represents the row space of A and A's data points (blue and black points) lie on the ℓ_1-normalized plane $\mathbb{P}$. The blue points also stand for the anchors of A. A random vector β is picked up from row space of A and then data points are projected on β. The anchors of the projected space on β are still the anchors of A, such that implies that the red point with the maximum absolute projection component on β is an anchor of A.

help us make an approximate $\hat{\beta}_i = \hat{V}x_i$ for β_i. Therefore, we will estimate distribution $\mathcal{D}_{\hat{U}\hat{U}^TA\hat{V}x_i}$ instead of $\mathcal{D}_{A\beta_i}$. Based on Corollary 3.5, $\hat{U}^TA\hat{V}$ can be estimated efficiently. According to Lemma 3.4, $\hat{U}y$ can be sampled efficiently, thus we can treat $\tilde{y}$ as estimation of $\hat{U}^TA\hat{V}x_i$ (see Figure 2).

Once we can simulate distribution $\mathcal{D}_{AVx_i}$, we can figure out the index of the largest component of vector AVx_i by picking up $O(poly\log m)$ samples (Theorem 3.6). Moreover, according to [Zhou *et al.*, 2013], by repeating this procedure with $O(k\log k)$ times, we can find all anchors of A with high probability (For single step of random projection, see Figure 3).

3.2 Analysis

Now, we propose our main theorem and analyze the correctness and complexity of our algorithm FAS.

Theorem 3.1 (Main Result)**.** *Given separable non-negative matrix $A \in \mathbb{R}^{m\times n}_{\geq 0}$ in matrix sample model, the rank k, condition number κ and a constant $\delta \in (0,1)$, Algorithm 1 returns the indices of anchors with probability at least $1-\delta$ in time*

$$O\left(poly\left(k, \kappa, \log\frac{1}{\delta}, \log(mn)\right)\right).$$

Correctness

In this subsection, we will analyze the correctness of Algorithm 1. Firstly, we show that the columns of V defined in Lemma 2.6 form a basis of row space of matrix A, which is necessary to generate unit vector in row space of A. The next, we prove that for each $i \in [s]$, distribution $\mathcal{O}_i$ is ϵ-close to distribution $\mathcal{D}_{AVx_i}$ in total variant distance. Once again, we show how to gain the index of largest component of AVx_i from distribution $\mathcal{O}_i$. Finally, by $O(k\log k)$ random projection, it is enough for us to gain all anchors of matrix A.

The following lemma tells us the approximate singular vectors outputted by FKV spans the row space of matrix A. And combining with Lemma 2.6, it gives us that V also spans the same space, i.e., V forms an orthonormal basis of row space of matrix A.

Lemma 3.2. *Let $\hat{V}$ be the output of algorithm* FKV (A, k, ϵ, δ).

Algorithm 1 Fast Anchors Seeking Algorithm

Input: Separable non-negative matrix $A \in \mathbb{R}^{m\times n}_{\geq 0}$ in matrix sample model, $k = rank(A)$, condition number κ, a constant $\delta \in (0,1)$ and $s = O(k\log k)$.
Output: The index set R of anchors for A.

1: Initialize $R = \emptyset$.
2: Set $\epsilon < 2\sqrt{2\log(4\log^2(m)/\delta)/\log^2(m)}$.
3: Set $\epsilon_V = O\left(\min\left\{\epsilon/\sqrt{k}\kappa, 1/k\kappa^2\right\}\right)$, $\delta_V = 1-(1-\delta)^{\frac{1}{4}}$.
4: Run FKV $(A, k, \epsilon_V, \delta_V)$ and output the description of approximate right singular vectors $\hat{V}$.
5: Set $\epsilon_U = O\left(\min\left\{\frac{\epsilon}{k}, \frac{1}{k\kappa^2}\right\}\right)$, $\delta_U = 1-(1-\delta)^{\frac{1}{4}}$.
6: Run FKV $(A^T, k, \epsilon_U, \delta_U)$ and output the description of approximate left singular vectors $\hat{U}$.
7: By Lemma 3.5, estimate $M := \hat{U}^TA\hat{V}$ with relative error $\zeta = O\left(\epsilon/k^2\kappa\right)$ and failure probability $\eta = 1-(1-\delta)^{\frac{1}{4}}$, and denote the result as $\tilde{M}$.
8: **for** $i = 1$ to s **do**
9: Generate a unit random vector $x_i \in \mathbb{R}^k$.
10: Directly compute $\tilde{y}_i = \tilde{M}x_i$.
11: By rejection sampling (Algorithm 2), simulate distribution $\mathcal{D}_{\hat{U}\tilde{y}}$ with failure probability $\gamma = 1-(1-\delta)^{\frac{1}{4s}}$ and pick up $O(poly\log m)$ samples.
{Let $\mathcal{O}_i$ denotes the actual distribution which simulates $\mathcal{D}_{\hat{U}\tilde{y}_i}$ }
12: $R \leftarrow R \cup \{l\}$, where l is the most frequently index appearing in $O(poly\log m)$ samples.
13: **end for**
14: Return R

If $\epsilon < \frac{1}{k\kappa^2}$, then with probability $1-\delta$, we obtain

$$\text{span}\left\{\hat{V}^{(i)}\middle| i \in [l], l \le k\right\} = \text{span}\left\{A_{(i)}\middle| i \in [m]\right\}.$$

Proof. By contradiction, we assume that $\text{span}\left\{\hat{V}^{(i)}\middle| i \in [k]\right\} \neq \text{span}\left\{A_{(i)}\middle| i \in [m]\right\}$, which implies that there exists a unit vector $x \in \text{span}\left\{A_{(i)}\middle| i \in [m]\right\}$ and $x \perp \text{span}\left\{\hat{V}^{(i)}\middle| i \in [k]\right\}$. Then we can obtain $\left\|Ax - A\hat{V}\hat{V}^T x\right\| = \|Ax\| \ge \sigma_{\min}(A)$ since $\hat{V}^T x = \vec{0}$. And according to Theorem 2.4, we have

$$\left\|Ax - A\hat{V}\hat{V}^T x\right\| \le \sqrt{\epsilon}\|A\|_F \le \sqrt{\epsilon k}\kappa\sigma_{\min}(A).$$

Thus $\sigma_{\min}(A) \le \sqrt{\epsilon k}\kappa\sigma_{\min}(A)$, which makes a contradiction if $\epsilon < 1/k\kappa^2$. □

By Lemma 3.2, we can generate an approximate random vector in the row space of A with probability $1-\delta$ in time $O(poly(k, 1/\epsilon, \log 1/\delta))$ by FKV (A, k, ϵ, δ). Firstly, we obtain the description of approximate right singular vectors by FKV algorithm, where the error parameter ϵ is bounded by rank k and condition number κ (see in Lemma 3.2). Secondly, we generate a random unit vector $x_i \in \mathbb{R}^k$ as a coordinate vector referring to a set of orthonormal vectors in Lemma 2.6. Let V denotes the matrix defined in Lemma 2.6, then it is obvious that its columns form the right singular vectors for matrix A. That is, $\hat{\beta}_i = \hat{V}x_i$ is an approximate vector of a random vector $\beta = Vx_i$. Next, we show that total variant distance between O_i and $\mathcal{D}_{AVx_i}$ is bounded by constant ϵ. For convenience, we assume that each step in Algorithm 1 succeeds and the final success probability will be given in next subsection.

Lemma 3.3. *For all $i \in [s]$, $\left\|O_i, \mathcal{D}_{AVx_i}\right\|_{TV} \le \epsilon$ holds simultaneously with probability $1-\delta$.*

In the rest, without ambiguity, we use notations O, x instead of O_i, x_i. By applying triangle inequality, we divide the left part of inequality into four parts (the intuition idea please ref Figure 2):

$$\begin{aligned}\left\|\mathcal{D}_{AVx}, O\right\|_{TV} \le & \underbrace{\left\|\mathcal{D}_{AVx}, \mathcal{D}_{A\hat{V}x}\right\|_{TV}}_{①} + \underbrace{\left\|\mathcal{D}_{A\hat{V}x}, \mathcal{D}_{\hat{U}\hat{U}^T A\hat{V}x}\right\|_{TV}}_{②} \\ & + \underbrace{\left\|\mathcal{D}_{\hat{U}\hat{U}^T A\hat{V}x}, \mathcal{D}_{\hat{U}\tilde{M}x}\right\|_{TV}}_{③} + \underbrace{\left\|\mathcal{D}_{\hat{U}\tilde{M}x}, O\right\|_{TV}}_{④}.\end{aligned}$$

Thus, we only need to prove that ①, ②, ③, ④ $< \frac{\epsilon}{4}$, respectively. In addition, given $u, v \in \mathbb{R}^n$, if $\|u - v\| \le \frac{\epsilon}{2}\|u\|$, then $\|\mathcal{D}_u, \mathcal{D}_v\|_{TV} \le \epsilon$. For ①, ② and ③, we only show their ℓ_2-norm version, *i.e.*,

- $\|AVx - A\hat{V}x\| \le \frac{\epsilon}{8}\|AVx\|$;
- $\|A\hat{V}x - \hat{U}\hat{U}^T A\hat{V}x\| \le \frac{\epsilon}{8}\|A\hat{V}x\|$;
- $\|\hat{U}\hat{U}^T A\hat{V}x - \hat{U}\tilde{M}x\| \le \frac{\epsilon}{8}\|\hat{U}\hat{U}^T A\hat{V}x\|$.

For convenience, in the rest part, let $\alpha_U = \epsilon_U k/16$ and $\alpha_V = \epsilon_V k/16$ represent approximate ratio for orthonormality of $\hat{U}$ and $\hat{V}$ based on Lemma 2.7, respectively.

Before we start our proof, we list two tools which are used to prove ③ and ④, respectively.

Based on rejection sampling, Lemma 3.4 shows that sampling from linear combination of α-approximately orthogonal vectors can be quickly realized without knowledge of norms of these vectors (see Algorithm 2).

Algorithm 2 Rejection Sampling for $\mathcal{D}_{\hat{U}y}$

Input: A set of approximately orthonormal vectors $\hat{U}^{(j)} \in \mathbb{R}^m$ (for $j = 1, \ldots, k$) in *vector sample model* and a vector $y \in \mathbb{R}^k$.
Output: A sample s subjecting to $\mathcal{D}_{\hat{U}y}$.

1: Sample j according to probability proportional to $\|\hat{U}^{(j)} y_j\|$.
2: Sample s from $\mathcal{D}_{\hat{U}^{(j)}}$.
3: Compute $r_s = \frac{(\hat{U}y)_s^2}{k\sum_i (y_i \hat{U}_{ij})}$.
4: Accept s with probability r_s, otherwise, restart.

Lemma 3.4 ([Tang, 2018])**.** *Given a set of α-approximately orthonormal vectors $\hat{V} \in \mathbb{R}^{n\times k}$ in vector sample model, and an input vector $w \in \mathbb{R}^k$, there exists an algorithm outputting a sample from a distribution $\frac{\alpha}{1-\alpha}$-close to $\mathcal{D}_{\hat{V}w}$ with probability $1-\gamma$ using $O(k^2 \log\frac{1}{\gamma}(1 + O(\alpha)))$ queries and samples.*

Lemma 3.5. *Given $A \in \mathbb{R}^{m\times n}$ in* matrix sample model *and $L \in \mathbb{R}^{k_1\times m}$ and $R \in \mathbb{R}^{n\times k_2}$ in query model, let $M = LAR$, then we can output a matrix $\tilde{M} \in \mathbb{R}^{k_1\times k_2}$, with probability $1-\eta$, such that*

$$\left\|M - \tilde{M}\right\|_F \le \zeta\|A\|_F\|L\|_F\|R\|_F$$

by $O\left(k_1 k_2 \frac{1}{\zeta^2}\log\frac{1}{\eta}\right)$ queries and samples.

Proof. Let $M_{ij} = L_{(i)}AR^{(j)}$ with $i \in [k_1]$ and $j \in [k_2]$. In [Tang, 2018], there exists an algorithm that outputs an estimation of M_{ij} ($\tilde{M}_{ij}$) to precision $\zeta\|A\|_F\left\|L_{(i)}\right\|\left\|R^{(j)}\right\|$ with probability $1-\eta'$ in time $O\left(\frac{1}{\zeta^2}\log\frac{1}{\eta'}\right)$. Let $\eta' = 1-(1-\eta)^{1/(k_1k_2)}$. We can output $\tilde{M}$ with probability $1-\eta$ utilizing $O\left(k_1k_2\frac{1}{\zeta^2}\log\frac{1}{1-(1-\eta)^{1/k^2}}\right) = O\left(k_1k_2\frac{1}{\zeta^2}\log\frac{1}{\eta}\right)$ queries and samples respectively where $\tilde{M}$ satisfies

$$\begin{aligned}\left\|M - \tilde{M}\right\|_F^2 &= \sum_{i\in[k_1], j\in[k_2]} |M_{ij} - \tilde{M}_{ij}|^2 \\ &\le \sum_{i\in[k_1], j\in[k_2]} \zeta^2\|A\|_F^2\left\|L_{(i)}\right\|^2\left\|R^{(j)}\right\|^2 \\ &= \zeta^2\|A\|_F^2 \sum_{i\in[k_1]}\left\|L_{(i)}\right\|^2 \sum_{j\in[k_2]}\left\|R^{(j)}\right\|^2 \\ &= \zeta^2\|A\|_F^2\|L\|_F^2\|R\|_F^2.\end{aligned}$$

□

proof of Lemma 3.3. **Upper bound for** ①. By Lemma 3.2, Vx is a unit random vector sampled from the row space of A with probability $1-\delta_V := (1-\delta)^{\frac{1}{4}}$ if $\epsilon_V < \frac{1}{k\kappa^2}$. From Eq. (1) in Lemma 2.6, with probability $1-\delta_V$

$$\left\|AVx - A\hat{V}x\right\| \le \|A\|_F\left\|V - \hat{V}\right\|_F\|x\| \le (\alpha_V/\sqrt{2} + c_1\alpha_V^2)\|A\|_F.$$

Combing with $\|A\|_F \le \sqrt{k}\kappa\sigma_{\min}(A) \le \sqrt{k}\kappa\|AVx\|$, we gain

$$\left\|AVx - A\hat{V}x\right\| \le (\alpha_V/\sqrt{2} + c_1\alpha_V^2)\sqrt{k}\kappa\|AVx\|. \quad (3)$$

Eq. (3) satisfies $\left\|AVx - A\hat{V}x\right\| \le \frac{\epsilon}{8}\|AVx\|$ with $\epsilon_V = O\left(\min\left\{\frac{\epsilon}{\sqrt{k}\kappa}, \frac{1}{k\kappa^2}\right\}\right)$.

Upper bound for ②. According to Lemma 3.2, the columns of $\hat{U}$ span a space equal to the column space of A if $\epsilon_U \leq \frac{1}{k\kappa^2}$ with probability $1-\delta_U := (1-\delta)^{\frac{1}{4}}$. Let $\Pi_{\hat{U}}$ denote the orthonormal projector to image of $\hat{U}$ (column space of A). Similarly, $\Pi^{\perp}_{\hat{U}}$ denotes the orthonormal projector to the orthogonal space of column space of $\hat{U}$.

$$\begin{aligned}&\left\|A\hat{V}x-\hat{U}\hat{U}^TA\hat{V}x\right\|\\=&\left\|(\Pi_{\hat{U}}+\Pi^{\perp}_{\hat{U}})A\hat{V}x-\hat{U}\hat{U}^TA\hat{V}x\right\|=\left\|\Pi_{\hat{U}}A\hat{V}x-\hat{U}\hat{U}^TA\hat{V}x\right\|\\\leq&\left\|\Pi_{\hat{U}}-\hat{U}\hat{U}^T\right\|_F\left\|A\hat{V}x\right\|\leq c_2\alpha_U\left\|A\hat{V}x\right\|,\end{aligned}\tag{4}$$

based on Eq. (2) in Lemma 2.6. If $\epsilon_U = O\left(\min\left\{\frac{\epsilon}{k},\frac{1}{k\kappa^2}\right\}\right)$, $\left\|A\hat{V}x-\hat{U}\hat{U}^TA\hat{V}x\right\|\leq\frac{\epsilon}{8}\left\|A\hat{V}x\right\|$.

Upper bound for ③. When ϵ_U and ϵ_V are discussed above, with probability $1-\eta := (1-\delta)^{\frac{1}{4}}$, we have

$$\left\|\hat{U}\hat{U}^TA\hat{V}x\right\|\geq\left(1-\frac{\epsilon}{8}\right)\left\|A\hat{V}x\right\|\geq\left(1-\frac{\epsilon}{8}\right)^2\|AVx\|.\tag{5}$$

According to Lemma 3.5, we obtain

$$\begin{aligned}\left\|\hat{U}^TA\hat{V}-\tilde{M}\right\|_F&\leq\zeta\left\|\hat{U}\right\|_F\left\|\hat{V}\right\|_F\|A\|_F\\&\leq\zeta k^{1.5}\kappa\sqrt{(1+\frac{\alpha_U}{k})(1+\frac{\alpha_V}{k})}\|AVx\|.\end{aligned}\tag{6}$$

Combining Eq. (5) and Eq. (6), the following holds

$$\begin{aligned}&\left\|\hat{U}\hat{U}^TA\hat{V}x-\hat{U}\tilde{M}x\right\|\leq\left\|\hat{U}\right\|_F\left\|\hat{U}^TA\hat{V}-\tilde{M}\right\|_F\\&\leq\zeta k^2\kappa\sqrt{(1+\frac{\alpha_U}{k})^2(1+\frac{\alpha_V}{k})}\|AVx\|\\&\leq\zeta k^2\kappa\sqrt{(1+\frac{\alpha_U}{k})^2(1+\frac{\alpha_V}{k})}/(1-\frac{\epsilon}{8})^2\left\|\hat{U}\hat{U}^TA\hat{V}x\right\|.\end{aligned}\tag{7}$$

If $\zeta = O\left(\frac{\epsilon}{k^2\kappa}\right)$, then $\left\|\hat{U}\hat{U}^TA\hat{V}x-\hat{U}\tilde{M}x\right\| < \frac{\epsilon}{8}\left\|\hat{U}\hat{U}^TA\hat{V}x\right\|$ holds.

Upper bound for ④. Since $\epsilon_U = O\left(\min\left\{\frac{\epsilon}{k},\frac{1}{k\kappa^2}\right\}\right)$ as discussed before, directly taking usage of Lemma 3.4, with probability $1-\gamma := (1-\delta)^{\frac{1}{4s}}$ we have

$$\left\|\mathcal{D}_{\hat{U}\tilde{y}},\mathcal{O}\right\|_{TV}\leq\frac{\alpha_U}{1-\alpha_U}<\frac{\epsilon}{4}.\tag{8}$$

Hence, Algorithm 1 generates a distribution O_i which satisfies $\left\|O_i,\mathcal{D}_{AVx_i}\right\|_{TV}\leq\epsilon$ for s random unit vectors generated simultaneously with probability $1-\delta$. □

The following theorem tells us how to find the largest component of AVx_i from distribution O_i.

Theorem 3.6 (Restatement of Theorem 1 in [Du *et al.*, 2018])**.** *Let $\mathcal{D}$ be a distribution over $[m]$ and $\mathcal{D}'$ is another distribution simulating $\mathcal{D}$ with total variant error ϵ. Let $x_1,\ldots,x_N$ be examples independently sampled from $\mathcal{D}'$ and N_i be the number of examples taking value of i. Let $\mathcal{D}_{\max}=\max\{\mathcal{D}_1,\ldots,\mathcal{D}_m\}$ and $\mathcal{D}_{secmax}=\max\{\mathcal{D}_1,\ldots,\mathcal{D}_m\}\backslash\mathcal{D}_{\max}$. If $\mathcal{D}_{\max}-\mathcal{D}_{secmax}>2\sqrt{2\log(4N/\delta)/N}+\epsilon$, then, for any $\delta>0$, with a probability at least $1-\delta$, we have*

$$\arg\max_i\{N_i|1\leq i\leq N\}=\arg\max_i\{p_i|1\leq i\leq m\}.$$

As mentioned in [Du *et al.*, 2018], the assumption about the gap between $\mathcal{D}_{max}$ and $\mathcal{D}_{secmax}$ is easy to satisfy in practice. By choosing $N=\log^2 m$ and $\epsilon<2\sqrt{2\log(4\log^2 m/\delta)/\log^2 m}$, we have $\mathcal{D}_{max}-\mathcal{D}_{secmax}>4\sqrt{2\log(4\log^2 m/\delta)/\log^2 m}$, which will converge to zero as m goes to infinity.

To estimate the number of random projections we need, we denote p_i^* the probability that after random projection β, a data point $A_{(i)}$ is identified as an anchor in subspace, i.e.,

$$p_i^*=\Pr_{\beta}(i=\text{argmax}_i\{(A\beta)_i\}).$$

In [Zhou *et al.*, 2014], if $p_i^*>k/\alpha$ for a constant α, with $s=\frac{3}{\alpha}k\log k$ random projections, all anchors can be found with probability at least $1-k\exp(-\alpha s/3k)$.

Complexity and Success Probability

Note that Algorithm 1 involves operations that query and sample from matrix A, $\hat{U}$ and $\hat{V}$, but those operations can be implemented in $O(\log(mn)poly(k,\kappa,1/\epsilon))$ time. Thus, in the following analysis, we just ignore the time complexity of those operations but multiple it to the final time complexity.

The running time and failure probability mainly concentrates on lines 4, 6, 7 and 11 in Algorithm 1. The running time of lines 4 and 6 are $O\left(poly(k,1/\epsilon_V,\log 1/\delta_V)\right)$ and $O\left(poly(k,\frac{1}{\epsilon_U},\log\frac{1}{\delta_U})\right)$, respectively, according to Theorem 2.4. And line 7 takes $O\left(k^2\frac{1}{\zeta^2}\log\frac{1}{\eta}\right)$ to estimate matrix $\tilde{M}$ according to Lemma 3.5. And line 11 with s iterations totally spends $O\left(sk^2\log\frac{1}{\gamma}poly\log m\right)$. In the perspective of failure probability, lines 4, 6 and 7 take the same failure probabilities $(1-\eta)^{\frac{1}{4}}$. And line 11 takes $(1-\eta)^{\frac{1}{4s}}$ for each iteration.

Above all, the time complexity of FAS is $O\left(poly\left(k,\kappa,\log\frac{1}{\delta},\log mn\right)\right)$. The success probability is $1-\delta$.

4 Conclusion

This paper presents a classical randomized algorithm FAS which dramatically reduces the running time to find anchors of low-rank matrix. Especially, we achieve exponential speedup when the rank is logarithmic of the input scale. Although our algorithm running in polynomial of logarithm of matrix dimension, it still has a bad dependence on rank k. In the future, we plan to improve its dependence on rank as well as analyze its noise tolerance.

Acknowledgements

Part of this work is done during Y. Li's visit at the Institute of Quantum Computing, Baidu Inc.. This work is supported in part by the National Natural Science Foundation of China Grants No. 61433014, 61832003, 61761136014, 61872334, 61502449, the Strategic Priority Research Program of Chinese Academy of Sciences Grant No. XDB28000000, and Anhui Initiative in Quantum Information Technologies, Grant No. AHY150100. Y. Li is supported by ERC Consolidator Grant 615307-QPROGRESS. Y. Li thanks Runyao Duan for hosting his visit at the Institute of Quantum Computing, Baidu Inc..

References

[Arora *et al.*, 2012a] Sanjeev Arora, Rong Ge, Ravindran Kannan, and Ankur Moitra. Computing a nonnegative matrix factorization–provably. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 145–162. ACM, 2012.

[Arora *et al.*, 2012b] Sanjeev Arora, Rong Ge, and Ankur Moitra. Learning topic models–going beyond svd. In *Foundations of Computer Science (FOCS), 2012 IEEE 53rd Annual Symposium on*, pages 1–10. IEEE, 2012.

[Chia *et al.*, 2018] Nai-Hui Chia, Han-Hsuan Lin, and Chunhao Wang. Quantum-inspired sublinear classical algorithms for solving low-rank linear systems. *arXiv preprint arXiv:1811.04852*, 2018.

[Ding *et al.*, 2010] Chris HQ Ding, Tao Li, and Michael I Jordan. Convex and semi-nonnegative matrix factorizations. *IEEE transactions on pattern analysis and machine intelligence*, 32(1):45–55, 2010.

[Donoho and Stodden, 2004] David Donoho and Victoria Stodden. When does non-negative matrix factorization give a correct decomposition into parts? In *Advances in neural information processing systems*, pages 1141–1148, 2004.

[Du *et al.*, 2018] Yuxuan Du, Tongliang Liu, Yinan Li, Runyao Duan, and Dacheng Tao. Quantum divide-and-conquer anchoring for separable non-negative matrix factorization. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 2093–2099. AAAI Press, 2018.

[Elhamifar and Vidal, 2009] Ehsan Elhamifar and René Vidal. Sparse subspace clustering. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2790–2797. IEEE, 2009.

[Elhamifar *et al.*, 2012] Ehsan Elhamifar, Guillermo Sapiro, and Rene Vidal. See all by looking at a few: Sparse modeling for finding representative objects. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1600–1607. IEEE, 2012.

[Esser *et al.*, 2012] Ernie Esser, Michael Moller, Stanley Osher, Guillermo Sapiro, and Jack Xin. A convex model for nonnegative matrix factorization and dimensionality reduction on physical space. *IEEE Transactions on Image Processing*, 21(7):3239–3252, 2012.

[Frieze *et al.*, 2004] Alan Frieze, Ravi Kannan, and Santosh Vempala. Fast monte-carlo algorithms for finding low-rank approximations. *Journal of the ACM (JACM)*, 51(6):1025–1041, 2004.

[Gillis and Vavasis, 2014] Nicolas Gillis and Stephen A Vavasis. Fast and robust recursive algorithmsfor separable nonnegative matrix factorization. *IEEE transactions on pattern analysis and machine intelligence*, 36(4):698–714, 2014.

[Gilyén *et al.*, 2018] András Gilyén, Seth Lloyd, and Ewin Tang. Quantum-inspired low-rank stochastic regression with logarithmic dependence on the dimension. *arXiv preprint arXiv:1811.04909*, 2018.

[Guan *et al.*, 2012] Naiyang Guan, Dacheng Tao, Zhigang Luo, and John Shawe-Taylor. Mahnmf: Manhattan non-negative matrix factorization. *stat*, 1050:14, 2012.

[Hofmann, 2017] Thomas Hofmann. Probabilistic latent semantic indexing. In *ACM SIGIR Forum*, volume 51, pages 211–218. ACM, 2017.

[Hsieh and Dhillon, 2011] Cho-Jui Hsieh and Inderjit S Dhillon. Fast coordinate descent methods with variable selection for non-negative matrix factorization. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1064–1072. ACM, 2011.

[Kim and Park, 2008] Jingu Kim and Haesun Park. Toward faster nonnegative matrix factorization: A new algorithm and comparisons. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 353–362. IEEE, 2008.

[Kumar *et al.*, 2013] Abhishek Kumar, Vikas Sindhwani, and Prabhanjan Kambadur. Fast conical hull algorithms for near-separable non-negative matrix factorization. In *International Conference on Machine Learning*, pages 231–239, 2013.

[Lee and Seung, 1999] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788, 1999.

[Lee and Seung, 2001] Daniel D Lee and H Sebastian Seung. Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems*, pages 556–562, 2001.

[Lin, 2007] Chih-Jen Lin. Projected gradient methods for nonnegative matrix factorization. *Neural computation*, 19(10):2756–2779, 2007.

[Mahoney and Drineas, 2009] Michael W Mahoney and Petros Drineas. Cur matrix decompositions for improved data analysis. *Proceedings of the National Academy of Sciences*, pages pnas–0803205106, 2009.

[Recht *et al.*, 2012] Ben Recht, Christopher Re, Joel Tropp, and Victor Bittorf. Factoring nonnegative matrices with linear programs. In *Advances in Neural Information Processing Systems*, pages 1214–1222, 2012.

[Tang, 2018] Ewin Tang. A quantum-inspired classical algorithm for recommendation systems. *arXiv preprint arXiv:1807.04271*, 2018.

[Vavasis, 2009] Stephen A Vavasis. On the complexity of nonnegative matrix factorization. *SIAM Journal on Optimization*, 20(3):1364–1377, 2009.

[Zhou *et al.*, 2013] Tianyi Zhou, Wei Bian, and Dacheng Tao. Divide-and-conquer anchoring for near-separable nonnegative matrix factorization and completion in high dimensions. In *2013 IEEE 13th International Conference on Data Mining*, pages 917–926. IEEE, 2013.

[Zhou *et al.*, 2014] Tianyi Zhou, Jeff A Bilmes, and Carlos Guestrin. Divide-and-conquer learning by anchoring a conical hull. In *Advances in Neural Information Processing Systems*, pages 1242–1250, 2014.