Efficient Nonnegative Matrix Factorization with Random Projections

Fei Wang*   Ping Li†

Abstract

The recent years have witnessed a surge of interests in Nonnegative Matrix Factorization (NMF) in data mining and machine learning fields. Despite its elegant theory and empirical success, one of the limitations of NMF based algorithms is that it needs to store the whole data matrix in the entire process, which requires expensive storage and computation costs when the data set is large and high-dimensional. In this paper, we propose to apply the random projection techniques to accelerate the NMF process. Both theoretical analysis and experimental validations will be presented to demonstrate the effectiveness of the proposed strategy.

1 Introduction

Factorization of matrices is a fundamental technique for data analysis. There have already been a variety of powerful methods, such as *Principal Component Analysis* (*PCA*) [18] and *Singular Value Decomposition* (*SVD*) [13]. During the last decade, *Nonnegative Matrix Factorization* (*NMF*) [21] has aroused considerable interests from the machine learning and data mining fields; and NMF has been successfully applied in many areas, such as computer vision [14][28][10], information retrieval [36][9] and bioinformatics [19][11].

Basically, suppose we have a nonnegative data matrix $\mathbf{X} \in \mathbb{R}^{d \times n}$, where d is the dimensionality of the data points, n is the data set size. The goal of NMF is to factorize $\mathbf{X}$ into the product of two nonnegative matrices $\mathbf{F} \in \mathbb{R}^{d \times r}$ and $\mathbf{G} \in \mathbb{R}^{n \times r}$ (usually $r \ll \min(n, d)$) by minimizing the following loss

$$(1.1) \qquad J(\mathbf{F}, \mathbf{G}) = \left\| \mathbf{X} - \mathbf{F}\mathbf{G}^T \right\|_F^2$$

with $\| \cdot \|_F$ representing the matrix Frobenius norm. Some researchers have proposed other types of loss functions such as the matrix divergence [21].

During the last decade, many algorithm have been proposed to solve the NMF problem, such as *Alternating Least Squares* [20], *Multiplicative Updates* [21] and *Projected Gradient Descent* [29]. According to [20], all

---

*Department of Statistical Science, Cornell University.
†Department of Statistical Science, Cornell University.

these methods can be derived from the following *two-block coordinate descent* framework [5]:

- Initialize $\mathbf{G}$ with a nonnegative matrix $\mathbf{G}^{(0)}$; $t \leftarrow 0$

- Repeat until a stopping criterion is satisfied

  − Find $\mathbf{F}^{(t+1)}$: $J(\mathbf{F}^{(t+1)}, \mathbf{G}^{(t)}) \leqslant J(\mathbf{F}^{(t)}, \mathbf{G}^{(t)})$
  − Find $\mathbf{G}^{(t+1)}$: $J(\mathbf{F}^{(t+1)}, \mathbf{G}^{(t+1)}) \leqslant J(\mathbf{F}^{(t+1)}, \mathbf{G}^{(t)})$

To enhance the applicability of those NMF based algorithms, Ding et al. [12] generalized the traditional NMF algorithm to handle data with both nonnegative and negative entries; and they named the proposed algorithm **Semi-NMF**. The goal of semi-NMF is to factorize $\mathbf{X}$ (whose entries could be either nonnegative or negative) into the product of two matrices $\mathbf{F} \in \mathbb{R}^{d \times r}$ and $\mathbf{G} \in \mathbb{R}^{n \times r}$, where only $\mathbf{G}$ is constrained to be nonnegative. According to the analysis in [12], semi-NMF is equivalent to a relaxed version of K-means clustering, where r is the number of clusters, $\mathbf{F} = [\mathbf{f}_1, \mathbf{f}_2, \cdots, \mathbf{f}_r]$ is composed of the cluster centers, and $\mathbf{G}$ is the cluster indicator matrix with $\mathbf{G}_{ij}$ being the possibility of datum $\mathbf{x}_i$ belonging to cluster j. The Semi-NMF algorithm has also successfully applied to template matching [33], gene clustering [32] and relational learning [35].

Despite their theoretical and empirical success, one of the main limitations of NMF based algorithms is that they need to store the data matrix $\mathbf{X}$ in the main memory throughout the computation process. This can be prohibitive when the data set is large and high dimensional. For example, in web scale data mining, one may commonly encounter data sets of size $n = 10^{10}$ web pages and each may be represented by $d = 10^5$ dimensions (using single words) or 10^{15} dimensions (using 3 contiguous words, i.e., 3-shingles).

As a dimension reduction technique, **Random Projection** [34][23][24][17] provides an efficient mechanism to compress the data. Basically, the random projection method makes use of a random projection matrix $\mathbf{R} \in \mathbb{R}^{k \times d}$ to project the original d-dimensional data matrix $\mathbf{X}$ into a k-dimensional space by

$$(1.2) \qquad \tilde{\mathbf{X}} = \frac{1}{\sqrt{k}} \mathbf{R}\mathbf{X} \qquad \in \mathbb{R}^{k \times n} \; (k \ll d)$$

The much smaller matrix $\widetilde{\mathbf{X}}$ can be used to approximate some measures in the original d-dimensional space, e.g., it has been proved that matrix $\widetilde{\mathbf{X}}$ can preserve the data vector norms and all pairwise distances of $\mathbf{X}$ in expectations [34]. In recent years, the random projection technique has successfully been applied to solve many computational data analysis problems, such as principal component analysis [7], singular value decomposition [7], least squares [8] and manifold learning [16].

In this paper, we propose to incorporate the random projection technique into the nonnegative matrix factorization process. Specifically, as there are both negative and nonnegative values in the random matrix $\mathbf{R}$, we first investigate the issue of semi-NMF with random projections, where $\mathbf{X}$ is not restricted to be nonnegative. Then we propose a dual random projection method to solve the original NMF problem. We also provide experimental results on applying our algorithm to three real world high dimensional dense data sets.

The rest of this paper is organized as follows. Section 2 introduces how to make use of random projection to accelerate the semi-NMF procedure. The detailed algorithm on NMF with random projections will be introduced in Section 3. Section 4 presents the experimental results of the proposed algorithms, followed by the conclusions and discussions in Section 5.

2 Semi-NMF with Random Projections

In this section we introduce how to apply random projection techniques to reduce the storage and computation burden of semi-NMF. First, we briefly review the basic semi-NMF algorithm.

2.1 An Overview of Semi-NMF As we stated in the introduction, semi-NMF also aims to factorize $\mathbf{X}$ into the product of $\mathbf{F}$ and $\mathbf{G}^T$, where only $\mathbf{G}$ is restricted to be nonnegative. The semi-NMF procedure also complies with the block-coordinate procedure: It starts with a random nonnegative $\mathbf{G}^{(0)}$, and then find an $\mathbf{F}^{(0)}$ such that

$$(2.3) \quad \mathbf{F}^{(0)} = \underset{\mathbf{F}}{\arg\min} J\left(\mathbf{F}, \mathbf{G}^{(0)}\right) = \left\| \mathbf{X} - \mathbf{F}\left(\mathbf{G}^{(0)}\right)^T \right\|_F^2$$

Since semi-NMF does not constrain $\mathbf{F}$ to be nonnegative, (2.3) is just an ordinary least square problem, and its solution would be

$$(2.4) \quad \mathbf{F}^{(0)} = \mathbf{X}\mathbf{G}^{(0)} \left[\left(\mathbf{G}^{(0)}\right)^T \mathbf{G}^{(0)} \right]^{-1}$$

The computational complexity of computing $\mathbf{F}^{(0)}$ would be $O(dnr + 2nr^2 + r^3)$. Then semi-NMF will find a $\mathbf{G}^{(1)}$ such that

$$(2.5) \quad J\left(\mathbf{F}^{(0)}, \mathbf{G}^{(1)}\right) \leqslant J\left(\mathbf{F}^{(0)}, \mathbf{G}^{(0)}\right)$$

which can be achieved by one of the following two ways.

- *Multiplicative Update* [12], which updates $\mathbf{G}^{(1)}$ by

$$(2.6)$$
$$\mathbf{G}_{ij}^{(1)} \longleftarrow \mathbf{G}_{ij}^{(0)} \sqrt{ \frac{ (\mathbf{X}^T\mathbf{F}^{(0)})_{ij}^+ + [\mathbf{G}^{(0)}((\mathbf{F}^{(0)})^T\mathbf{F}^{(0)})^-]_{ij} }{ (\mathbf{X}^T\mathbf{F}^{(0)})_{ij}^- + [\mathbf{G}^{(0)}((\mathbf{F}^{(0)})^T\mathbf{F}^{(0)})^+]_{ij} } }$$

  whose computational complexity can reach $O(ndr + rd^2 + n^2 r)$, which would be extremely high when n and d are both large. Then the algorithm iterates between Eq.(2.4) and Eq.(2.6) to find $\mathbf{F}^{(1)}$, $\mathbf{G}^{(2)}$ and so on and so forth, until the procedure converges. The convergence property has been proved in [12]. One issue should be mentioned here is that since the optimal $\mathbf{F}^{(t)}$ can be expressed in a closed form as in Eq.(2.4), we can update $\mathbf{G}^{(t+1)}$ each time without explicitly compute $\mathbf{F}^{(t)}$, by

$$(2.7)$$
$$\mathbf{G}_{ij}^{(t+1)} \longleftarrow \mathbf{G}_{ij} \sqrt{ \frac{ (\mathbf{K}\mathbf{\Theta}^{(t)})_{ij}^+ + [\mathbf{G}^{(t)}((\mathbf{\Theta}^{(t)})^T\mathbf{K}\mathbf{\Theta}^{(t)})^-]_{ij} }{ (\mathbf{K}\mathbf{G}^{(t)})_{ij}^- + [\mathbf{G}^{(t)}((\tilde{\mathbf{G}}^{(t)})^T\mathbf{K}\mathbf{G}^{(t)})^+]_{ij} } }$$

  where $\mathbf{K} = \mathbf{X}^T\mathbf{X}$ is the Gram matrix, and

$$\mathbf{\Theta}^{(t)} = \mathbf{G}^{(t)} \left[\left(\mathbf{G}^{(t)}\right)^T \left(\mathbf{G}^{(t)}\right) \right]^{-1}$$

  Whenever $\mathbf{F}^{(t)}$ is needed, we can just calculate it by Eq.(2.4) using $\mathbf{G}^{(t)}$.

- *Nonnegative Least Square* [20], which obtains $\mathbf{G}^{(1)}$ by solving the following optimization problem

$$(2.8) \quad \mathbf{G}^{(1)} = \underset{\mathbf{G} \geqslant 0}{\arg\min} J(\mathbf{F}^{(0)}, \mathbf{G}) = \left\| \mathbf{X} - \mathbf{F}^{(0)}\mathbf{G}^T \right\|_F^2$$

  which can be solved, for example, by the active set method [3][20]. Similar to Eq.(2.7), we can also obtain $\mathbf{G}^{(t+1)}$ by solving

$$(2.9) \quad \mathbf{G}^{(t+1)} = \underset{\mathbf{G} \geqslant 0}{\arg\min} J(\mathbf{G}) = \left\| \mathbf{X} - \mathbf{X}\mathbf{\Theta}^{(t)}\mathbf{G}^T \right\|_F^2$$

  without explicitly computing $\mathbf{F}^{(t)}$.

2.2 Semi-NMF with Random Projections As we have mentioned, when the data set is large and high dimensional, updating $\mathbf{F}$ and $\mathbf{G}$ would be time and space expensive. Thus, we propose to first left multiply $\mathbf{X} \in \mathbb{R}^{d \times n}$ with a matrix $\widetilde{\mathbf{R}} = \frac{1}{\sqrt{k}}\mathbf{R} \in \mathbb{R}^{k \times d}$ ($k \ll d$), with $\mathbf{R} \in \mathbb{R}^{k \times d}$ being a normal random matrix[1], to

---

[1] We say $\mathbf{R}$ is a normal random matrix if each of its entry r_{ij} is chosen independently from a standard normal $\mathcal{N}(0,1)$. Actually $\mathbf{R}$ can be a random matrix of other types as in [1][27].

compress its dimensionality from d to k, and then perform semi-NMF on $\widetilde{\mathbf{X}} = \widetilde{\mathbf{R}}\mathbf{X} \in \mathbb{R}^{k \times n}$, i.e., solve the following optimization problem

$$(2.10) \qquad \min_{\widetilde{\mathbf{G}} \geqslant 0, \widetilde{\mathbf{F}}} \left\| \widetilde{\mathbf{X}} - \widetilde{\mathbf{F}}\widetilde{\mathbf{G}}^T \right\|_F^2$$

where $\widetilde{\mathbf{F}} \in \mathbb{R}^{k \times r}$ and $\widetilde{\mathbf{G}} \in \mathbb{R}^{n \times r}$. In this case, the size of $\widetilde{\mathbf{G}}$ would still be the same as $\mathbf{G}$ in Eq.(1.1).

To solve problem (2.10), we still initialize $\widetilde{\mathbf{G}}^{(0)} = \mathbf{G}^{(0)}$, and obtain $\widetilde{\mathbf{F}}^{(0)}$ by solving the following (unconstrained) least squares problem

$$(2.11) \quad \widetilde{\mathbf{F}}^{(0)} = \underset{\widetilde{\mathbf{F}}}{\arg\min} \widetilde{J}(\widetilde{\mathbf{F}}, \widetilde{\mathbf{G}}^{(0)}) = \left\| \widetilde{\mathbf{X}} - \widetilde{\mathbf{F}} \left(\widetilde{\mathbf{G}}^{(0)} \right)^T \right\|_F^2$$

whose solution is

$$(2.12) \qquad \widetilde{\mathbf{F}}^{(0)} = \widetilde{\mathbf{X}}\widetilde{\mathbf{G}}^{(0)} \left[\left(\widetilde{\mathbf{G}}^{(0)} \right)^T \widetilde{\mathbf{G}}^{(0)} \right]^{-1} = \widetilde{\mathbf{R}}\mathbf{F}^{(0)}$$

where $\mathbf{F}^{(0)}$ is computed as in Eq.(2.4).

Therefore, to obtain $\widetilde{\mathbf{G}}^{(1)}$, we just need to solve

$$(2.13) \qquad \widetilde{\mathbf{G}}^{(1)} = \underset{\mathbf{G} \geqslant 0}{\arg\min} \widetilde{J}(\widetilde{\mathbf{F}}^{(0)}, \mathbf{G}) = \left\| \widetilde{\mathbf{X}} - \widetilde{\mathbf{F}}^{(0)}\mathbf{G}^T \right\|_F^2$$

Combining Eq.(2.13) with Eq.(2.12) yields

$$(2.14) \quad \widetilde{\mathbf{G}}^{(1)} = \underset{\mathbf{G} \geqslant 0}{\arg\min} \widetilde{J}(\widetilde{\mathbf{F}}^{(0)}, \mathbf{G}) = \left\| \widetilde{\mathbf{R}}\mathbf{X} - \widetilde{\mathbf{R}}\mathbf{F}^{(0)}\mathbf{G}^T \right\|_F^2$$

which is a compressed nonnegative least square regression problem [8][30]. According to [2], we have the following Theorem.

THEOREM 2.1. *[2]. Let $\mathbf{R} = (r_{ij})$ be a random $k \times d$ matrix, such that each entry r_{ij} is chosen independently according to $\mathcal{N}(0, 1)$. For any vector fixed $\mathbf{u} \in \mathbb{R}^d$ and any $0 < \epsilon < 1$, let $\widetilde{\mathbf{u}} = \widetilde{\mathbf{R}}\mathbf{u} = \frac{1}{\sqrt{k}}\mathbf{R}\mathbf{u}$. Then, $E(\|\widetilde{\mathbf{u}}\|^2) = \|\mathbf{u}\|^2$ and with the probability of at least $1 - e^{-(\epsilon^2 - \epsilon^3)\frac{k}{4}}$*

$$(2.15) \qquad \left| \|\widetilde{\mathbf{u}}\|^2 - \|\mathbf{u}\|^2 \right| \leqslant \epsilon \|\mathbf{u}\|^2 \qquad\qquad \square$$

Applying Theorem 2.1 to matrix $\mathbf{U}^{(0)} = \mathbf{X} - \mathbf{F}^{(0)}\mathbf{G}^T$, we obtain the following corollary.

COROLLARY 2.1. *Let $\widetilde{\mathbf{R}} = \frac{1}{\sqrt{k}}\mathbf{R}$ and $\mathbf{R}$ be a normal random matrix, then for any particular $\mathbf{G}$, $E(\|\widetilde{\mathbf{U}}^{(0)}\|_F^2) = \|\mathbf{U}^{(0)}\|_F^2$ with $\widetilde{\mathbf{U}}^{(0)} = \widetilde{\mathbf{R}}\mathbf{U}^{(0)}$, i.e., $E\left[\widetilde{J}\left(\widetilde{\mathbf{F}}^{(0)}, \mathbf{G} \right) \right] = J\left(\mathbf{F}^{(0)}, \mathbf{G} \right)$. Moreover, with the probability of at least $1 - e^{-(\epsilon^2 - \epsilon^3)\frac{k}{4}}$ with $0 < \epsilon < 1$, we have that*

$$(2.16) \quad \left| \widetilde{J}\left(\widetilde{\mathbf{F}}^{(0)}, \mathbf{G} \right) - J\left(\mathbf{F}^{(0)}, \mathbf{G} \right) \right| \leqslant \epsilon J\left(\mathbf{F}^{(0)}, \mathbf{G} \right) \qquad \square$$

Corollary 2.1 tells that, in the expectation sense, the random projection procedure would not change the objective value $J(\mathbf{F}^{(0)}, \mathbf{G})$. Moreover, we have the following theorem on the quality-of-approximation result of the final solutions.

THEOREM 2.2. *Let $\mathbf{G}_{opt} = \arg\min_{\mathbf{G} \geqslant 0} J(\mathbf{F}^{(0)}, \mathbf{G})$, and $\widetilde{\mathbf{G}}_{opt} = \arg\min_{\mathbf{G} \geqslant 0} \widetilde{J}(\widetilde{\mathbf{F}}^{(0)}, \mathbf{G})$. Then with the probability of at least $1 - e^{-(3\epsilon^2 - \epsilon^3)\frac{k}{108}}$ with $0 < \epsilon < 1$, we have that*

$$(2.17) \qquad J(\mathbf{F}^{(0)}, \widetilde{\mathbf{G}}_{opt}) \leqslant (1 + \epsilon)J(\mathbf{F}^{(0)}, \mathbf{G}_{opt})$$

Proof. See Appendix I.$\square$

From Corollary 2.1 and Theorem 2.2, we know that solving problem (2.13) is approximately the same as solving (2.8). As introduced in Section 2.1, the whole semi-NMF process actually only involves the successive updating of the $\mathbf{G}$ matrix. Therefore at each iteration, we can solve problem (2.13) instead of (2.8) (or update $\widetilde{\mathbf{G}}^{(t)}$ to descend $\widetilde{J}(\widetilde{\mathbf{F}}^{(t-1)}, \widetilde{\mathbf{G}}^{(t)})$).

Algorithm 1 summarizes the basic procedure of our semi-NMF with random projections algorithm.

Algorithm 1 SEMI-NMF WITH RANDOM PROJECTIONS

Require: Data Matrix $\mathbf{X} \in \mathbb{R}^{d \times n}$, Projection Matrix $\widetilde{\mathbf{R}} \in \mathbb{R}^{k \times d}$, Positive Integer r, Number of Iterations T
1: Construct the projected matrix $\widetilde{\mathbf{X}} = \widetilde{\mathbf{R}}\mathbf{X} \in \mathbb{R}^{k \times n}$
2: Randomly initialize $\widetilde{\mathbf{G}}^{(0)} \in \mathbb{R}^{n \times r}$ to be a nonnegative matrix
3: **for** $t = 1 : T$ **do**
4:     Construct

$$\widetilde{\mathbf{F}}^{(t)} = \widetilde{\mathbf{X}}\widetilde{\mathbf{G}}^{(t-1)} \left[\left(\widetilde{\mathbf{G}}^{(t-1)} \right)^T \left(\widetilde{\mathbf{G}}^{(t-1)} \right) \right]^{-1}$$

5:     Find $\widetilde{\mathbf{G}}^{(t)}$ with $\widetilde{J}(\widetilde{\mathbf{F}}^{(t)}, \widetilde{\mathbf{G}}^{(t)}) \leqslant \widetilde{J}(\widetilde{\mathbf{F}}^{(t)}, \widetilde{\mathbf{G}}^{(t-1)})$
6: **end for**
7: Output $\widetilde{\mathbf{G}}^{(T)}$ and $\widetilde{\mathbf{F}}^{(T)}$

---

3 NMF with Dual Random Projections

In this section we will introduce how to apply random projection to the regular NMF procedure. First we briefly review the basic NMF problem and algorithm.

3.1 An Overview of NMF The goal of NMF [21] is to factorize a nonnegative matrix $\mathbf{X}$ into the product of two (low rank) nonnegative matrices $\mathbf{F}$ and $\mathbf{G}$ by minimizing the Frobenius loss in Eq.(1.1). The NMF algorithm starts with initial nonnegative $\mathbf{G}^{(0)}$ and $\mathbf{F}^{(0)}$, and then seeks a nonnegative $\mathbf{F}^{(1)}$ such that

$$(3.18) \qquad J(\mathbf{G}^{(0)}, \mathbf{F}^{(1)}) \leqslant J(\mathbf{G}^{(0)}, \mathbf{F}^{(0)})$$

where $J(\mathbf{G}, \mathbf{F})$ is just the Frobenius loss as defined in Eq.(1.1). After obtaining $\mathbf{F}^{(1)}$, we then seek a $\mathbf{G}^{(1)}$ such that

$$(3.19) \qquad J(\mathbf{G}^{(1)}, \mathbf{F}^{(1)}) \leqslant J(\mathbf{G}^{(0)}, \mathbf{F}^{(1)})$$

The above process is repeated iteratively to find $\mathbf{F}^{(2)}, \mathbf{G}^{(2)}$... till convergence.

To achieve this goal, we can apply either the multiplicative update [21] or the alternating nonnegative least squares [4][20] method. One of the most well-known algorithms for the first type of method is Lee and Seung's multiplicative update rules, which updates $\mathbf{F}$ and $\mathbf{G}$ by

$$(3.20) \qquad \mathbf{G}_{ij} \quad \longleftarrow \quad \mathbf{G}_{ij} \frac{(\mathbf{X}^T \mathbf{F})_{ij}}{(\mathbf{G} \mathbf{F}^T \mathbf{F})_{ij}}$$

$$(3.21) \qquad \mathbf{F}_{ij} \quad \longleftarrow \quad \mathbf{F}_{ij} \frac{(\mathbf{X} \mathbf{G})_{ij}}{(\mathbf{F} \mathbf{G}^T \mathbf{G})_{ij}}$$

For the second type of method, [20] proposes to apply the active set method [3] to solve the nonnegative least square problems

$$(3.22) \qquad \min_{\mathbf{F}} \quad J(\mathbf{F}, \mathbf{G}^{(t)}) = \left\| \mathbf{X} - \mathbf{F}(\mathbf{G}^{(t)})^T \right\|_F^2$$

$$(3.23) \qquad \min_{\mathbf{G}} \quad J(\mathbf{F}^{(t)}, \mathbf{G}) = \left\| \mathbf{X} - \mathbf{F}^{(t)} \mathbf{G}^T \right\|_F^2$$

alternatingly. [20] showed that their method can converge much faster than the multiplicative update method, and usually achieve lower Frobenius loss.

One interesting issue should be mentioned here is that we can also apply semi-NMF style updating rules (Eq.(2.6)) to solve the NMF problem, as the problem of updating $\mathbf{G}$ with $\mathbf{F}$ fixed in semi-NMF is exactly the same as the problem of updating $\mathbf{G}$ with $\mathbf{F}$ fixed (and updating $\mathbf{F}$ with $\mathbf{G}$ fixed) in NMF. In this sense, since $\mathbf{X}, \mathbf{F}, \mathbf{G}$ are all required to be nonnegative in NMF, there would be no negative parts in $\mathbf{X}^T \mathbf{F}$ and $\mathbf{F}^T \mathbf{F}$; and hence the updating rules for $\mathbf{G}$ and $\mathbf{F}$ become

$$(3.24) \qquad \mathbf{G}_{ij} \quad \longleftarrow \quad \mathbf{G}_{ij} \sqrt{\frac{(\mathbf{X}^T \mathbf{F})_{ij}}{(\mathbf{G} \mathbf{F}^T \mathbf{F})_{ij}}}$$

$$(3.25) \qquad \mathbf{F}_{ij} \quad \longleftarrow \quad \mathbf{F}_{ij} \sqrt{\frac{(\mathbf{X} \mathbf{G})_{ij}}{(\mathbf{F} \mathbf{G}^T \mathbf{G})_{ij}}}$$

The convergence of the above updating rules is guaranteed by the following theorem.

THEOREM 3.1. *Update* $\mathbf{G}$ *and* $\mathbf{F}$ *using Eq.(3.24) and Eq.(3.25) to minimize* $J(\mathbf{F}, \mathbf{G}) = \left\| \mathbf{X} - \mathbf{F} \mathbf{G}^T \right\|_F^2$ *will finally converge.*

Proof. The above theorem can be directly derived from the convergence proof of the semi-NMF updating rules

in [12], where we first make use of the auxiliary function method to prove update one part (i.e., $\mathbf{F}$ or $\mathbf{G}$) with the other part fixed can monotonically decrease the objective function value $J(\mathbf{F}, \mathbf{G})$ (the form of the constructed auxiliary function is exactly the same as Eq.(18) in [12] with the negative part dropped). Since $J(\mathbf{F}, \mathbf{G})$ is obviously bounded so the algorithm will finally converge.□

Interestingly, the "new part" to be multiplied to the old $\mathbf{G}$ (or $\mathbf{F}$) in Eq.(3.24) (or Eq.(3.25)) is just the square root of the counterpart in Eq.(3.20) (or Eq.(3.21)). We thus expect that the updating rules in Eq.(3.24) and Eq.(3.25) will converge slower than the rules in Eq.(3.20) and Eq.(3.21).

3.2 NMF with Dual Random Projections

Applying random projections to NMF is not that straightforward as in semi-NMF, because the nonnegativity constraint would no longer hold if we multiply $\mathbf{X}$ by $\widetilde{\mathbf{R}}$. This is our motivation to decompose the NMF with random projection problem into the following two subproblems at each iteration step t:

- Find a nonnegative $\mathbf{G}^{(t)}$ such that
  $\widetilde{J}_G(\mathbf{F}^{(t-1)}, \mathbf{G}^{(t)}) \leqslant \widetilde{J}_G(\mathbf{F}^{(t-1)}, \mathbf{G}^{(t-1)})$

- Find a nonnegative $\mathbf{F}^{(t)}$ such that
  $\widetilde{J}_F(\mathbf{F}^{(t)}, \mathbf{G}^{(t)}) \leqslant \widetilde{J}_F(\mathbf{F}^{(t-1)}, \mathbf{G}^{(t-1)})$

where

$$\widetilde{J}_G(\mathbf{F}, \mathbf{G}) = \left\| \widetilde{\mathbf{R}}_d \mathbf{X} - \widetilde{\mathbf{R}}_d \mathbf{F} \mathbf{G}^T \right\|_F^2$$

$$\widetilde{J}_F(\mathbf{F}, \mathbf{G}) = \left\| \widetilde{\mathbf{R}}_n \mathbf{X}^T - \widetilde{\mathbf{R}}_n \mathbf{G} \mathbf{F}^T \right\|_F^2$$

and $\widetilde{\mathbf{R}}_d \in \mathbb{R}^{k_1 \times d} = \frac{1}{\sqrt{k_1}} \mathbf{R}_{k_1 \times d}$ $(k_1 \ll d)$, $\widetilde{\mathbf{R}}_n \in \mathbb{R}^{k_2 \times n} = \frac{1}{\sqrt{k_2}} \mathbf{R}_{k_2 \times n}$ $(k_2 \ll n)$ with $\mathbf{R}_{a \times b}$ being the normal random matrix of size $a \times b$.

From Corollary 2.1 and Theorem 2.2 we know that the minimization of $\widetilde{J}_G(\mathbf{F}, \mathbf{G})$ $(\widetilde{J}_F(\mathbf{F}, \mathbf{G}))$ with respect to $\mathbf{G}$ $(\mathbf{F})$ is approximately equivalent to the minimization of $J(\mathbf{F}, \mathbf{G})$ with respect to $\mathbf{G}$ $(\mathbf{F})$. This allows us to obtain an approximate solution to the original NMF by iteratively solving the following two small scale nonnegative least square problems:

$$(3.26) \qquad \mathbf{G}^{(t)} = \underset{\mathbf{G} \geqslant 0}{\operatorname{argmin}} \widetilde{J}_G(\mathbf{F}^{(t-1)}, \mathbf{G})$$

$$(3.27) \qquad \mathbf{F}^{(t)} = \underset{\mathbf{F} \geqslant 0}{\operatorname{argmin}} \widetilde{J}_F(\mathbf{F}, \mathbf{G}^{(t)})$$

The above two problems may be efficiently solved via the active set method [3][20]. As an alternative, by noting that the above two problems are exactly the same as the $\mathbf{G}$-update step in semi-NMF and using Eq.(2.6), we can also apply the following multiplicative update

rules to successively update $\mathbf{F}$ and $\mathbf{G}$:

$$(3.28) \quad \mathbf{G}_{ij} \longleftarrow \mathbf{G}_{ij} \sqrt{\frac{(\widetilde{\mathbf{X}}_d^T \widetilde{\mathbf{F}})_{ij}^+ + [\mathbf{G}(\widetilde{\mathbf{F}}^T \widetilde{\mathbf{F}})^-]_{ij}}{(\widetilde{\mathbf{X}}_d^T \widetilde{\mathbf{F}})_{ij}^- + [\mathbf{G}(\widetilde{\mathbf{F}}^T \widetilde{\mathbf{F}})^+]_{ij}}}$$

$$(3.29) \quad \mathbf{F}_{ij} \longleftarrow \mathbf{F}_{ij} \sqrt{\frac{(\widetilde{\mathbf{X}}_n \widetilde{\mathbf{G}})_{ij}^+ + [\mathbf{F}(\widetilde{\mathbf{G}}^T \widetilde{\mathbf{G}})^-]_{ij}}{(\widetilde{\mathbf{X}}_n \widetilde{\mathbf{G}})_{ij}^- + [\mathbf{F}(\widetilde{\mathbf{G}}^T \widetilde{\mathbf{G}})^+]_{ij}}}$$

where $\widetilde{\mathbf{X}}_d = \widetilde{\mathbf{R}}_d \mathbf{X}$, $\widetilde{\mathbf{X}}_n = \mathbf{X}\widetilde{\mathbf{R}}_n^T$, $\widetilde{\mathbf{F}} = \widetilde{\mathbf{R}}_d \mathbf{F}$, $\widetilde{\mathbf{G}} = \widetilde{\mathbf{R}}_n \mathbf{G}$. These updating rules can be viewed as the randomized approximation of the rules in Eq.(3.24) and Eq.(3.25).

Algorithm 2 NMF with Random Projections

Require: Data Matrix $\mathbf{X} \in \mathbb{R}^{d \times n}$, Projection Matrix $\widetilde{\mathbf{R}} \in \mathbb{R}^{k \times d}$, Positive Integer r, Number of Iterations T
1: Construct the projected matrix $\widetilde{\mathbf{X}}_d \in \mathbb{R}^{k_1 \times n}$ and $\widetilde{\mathbf{X}}_n \in \mathbb{R}^{d \times k_2}$
2: Randomly initialize $\mathbf{G}^{(0)} \in \mathbb{R}^{n \times r}$ and $\mathbf{F}^{(0)} \in \mathbb{R}^{d \times r}$ to be nonnegative matrices
3: **for** $t = 1 : T$ **do**
4:    Find a nonnegative $\mathbf{G}^{(t)}$ such that $\widetilde{J}_G(\mathbf{F}^{(t-1)}, \mathbf{G}^{(t)}) \leqslant \widetilde{J}_G(\mathbf{F}^{(t-1)}, \mathbf{G}^{(t-1)})$
5:    Find a nonnegative $\mathbf{F}^{(t)}$ such that $\widetilde{J}_F(\mathbf{F}^{(t)}, \mathbf{G}^{(t)}) \leqslant \widetilde{J}_F(\mathbf{F}^{(t-1)}, \mathbf{G}^{(t-1)})$
6: **end for**
7: Output $\widetilde{\mathbf{G}}^{(T)}$ and $\widetilde{\mathbf{F}}^{(T)}$

---

4 Experiments

This section presents a set of experiments on applying our nonnegative matrix factorization with random projection method using three real world data sets. The basic information of the data are summarized in Table 1; and more detailed information will be available in their respective sub-sessions.

Table 1: Data Set Information

Name	Dimension(d)	Size(n)	# Class
Harvard [6]	12600	203	5
Gisette [15]	5000	6000	2
COIL [31]	16384	7200	100

4.1 Harvard Microarray Data Set The Harvard Microarray data set [6] has been used in [22] for testing sparse random projection algorithms. It contains 203 samples (specimens) in 12600 gene dimensions, including 139 lung adenocarcinomas (12 of which may be suspicious), 17 normal samples, 20 pulmonary carcinoids, 21 sqamous cell lung carcinomas, and 6 SCLC cases.

We test the effectiveness of our random projection method on both semi-NMF and NMF. The number of columns r of both $\mathbf{F}$ and $\mathbf{G}$ is set to 5, which is equal to the actual number of clusters. Results with 4 different initializations will be reported.

4.1.1 Semi-NMF with Random Projections We conduct the multiplicative update and active set method to solve the semi-NMF problem. Both algorithms use the same randomly initialized $\mathbf{G}^{(0)}$. The number of iterations is set to 500 for multiplicative update, and 200 for active set method (since it converges faster, as can be seen from the experiments). The Frobenius loss at step t is computed as

$$(4.30) \qquad J(\mathbf{F}^{(t)}, \mathbf{G}^{(t)}) = \|\mathbf{X} - \mathbf{F}^{(t)}(\mathbf{G}^{(t)})^T\|_F^2$$

For the random projection method, we generate a $k \times d$ normal random matrix $\mathbf{R}$, where k varies from 50 to 1000 (which is very small compared to $d = 12600$ for this data set), and then run Algorithm 1 with $T = 500$ for the multiplicative method (or $T = 200$ for the active set method). For the convenience of comparison, we use the same $\mathbf{G}^{(0)}$ as in ordinary semi-NMF. For each k value, we conduct 100 independent runs with the same $\mathbf{G}^{(0)}$ and report the statistics of performance measures. The Frobenius loss at step t is computed as

$$(4.31) \qquad J'(\mathbf{F}'^{(t)}, \widetilde{\mathbf{G}}^{(t)}) = \|\mathbf{X} - \mathbf{F}'^{(t)}(\widetilde{\mathbf{G}}^{(t)})^T\|_F^2$$

where $\widetilde{\mathbf{G}}^{(t)}$ is obtained by solving the compressed nonnegative least square problem (2.13) with $\widetilde{\mathbf{F}}^{(t)}$ fixed (using multiplicative update or active set), and

$$\mathbf{F}'^{(t)} = \mathbf{X}\widetilde{\mathbf{G}}^{(t)} \left[\left(\widetilde{\mathbf{G}}^{(t)} \right)^T \widetilde{\mathbf{G}}^{(t)} \right]^{-1} \text{ as in Eq.(2.4)}.$$

As noted in [12], the final $\mathbf{G}$ of semi-NMF or NMF can also be viewed as a relaxed cluster indicator matrix, where $\mathbf{G}_{ij}$ represents the possibility that data point $\mathbf{x}_i$ belongs to cluster j. Therefore we also make use of $\mathbf{G}$ to calculate the clustering accuracy [35] on $\mathbf{X}$. The predicted cluster membership for data point $\mathbf{x}_i$ is determined according to $c_i = \underset{j}{\operatorname{argmax}} \ \mathbf{G}_{ij}$.

Fig.1 and Fig.2 respectively plot the variation of the Frobenius loss and the clustering accuracy with respect to the number of iterations for the semi-NMF method using multiplicative rules in Eq.(2.4) and Eq.(2.6) (referred to as **multiplicative semi-NMF**) with 4 different random initializations of $\mathbf{G}$. The solid lines in these figures are the lines with different projected dimensionality, which are averaged over 100 independent runs (i.e., we generate 100 different $\widetilde{\mathbf{R}}$ independently and compute the corresponding curves and average them). The dotted line corresponds to the curves calculated by original multiplicative semi-NMF without random projection. The initializations of $\mathbf{G}$ are set to be the same for semi-NMF with or without random projections.

From Fig.1 we can see that with the increase of the projected dimensionality, the Frobenius loss curve of the random projection method becomes closer to the
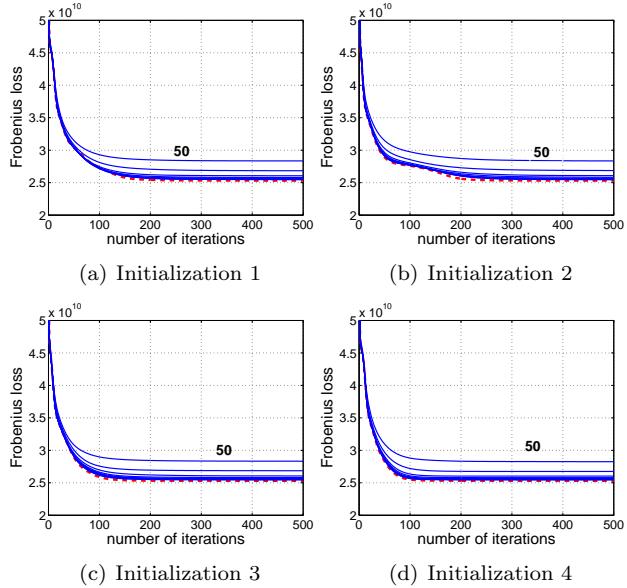
Figure 1: **Frobenius loss** variation over **500** iterations on **Harvard Microarray** data using **Multiplicative Semi-NMF** with 4 different initializations of **G**. Dotted line is the plot of original semi-NMF. Solid lines are the averaged (over 100 independent runs) plots of semi-NMF with random projections ($k = 50$ to 1000 from top to bottom).
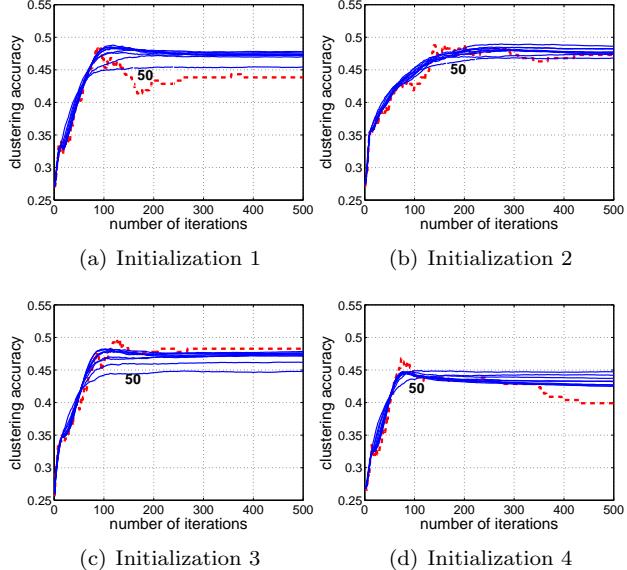

Figure 2: **Clustering accuracy** variation over **500** iterations on **Harvard Microarray** data using **Multiplicative Semi-NMF** with 4 different random initializations of **G**. The dotted line is the plot of original semi-NMF. The solid lines are the averaged (over 100 independent runs) plots of semi-NMF with random projections ($k = 50$ to 1000).

original curve; and when $k > 100$, the gap between the random projection curve and original curve diminishes.

Fig.2 shows the variation of the clustering accuracies with respect to the number of iterations. While there are usually some fluctuations during the iteration process, the curves with random projections are more smooth because they are averaged curves. We notice that it is not always the case that the clustering accuracy increases with increasing iterations. In other words, more iterations would not necessarily lead to better clustering results. Interestingly, for this data set, random projection may even produce better clustering results.
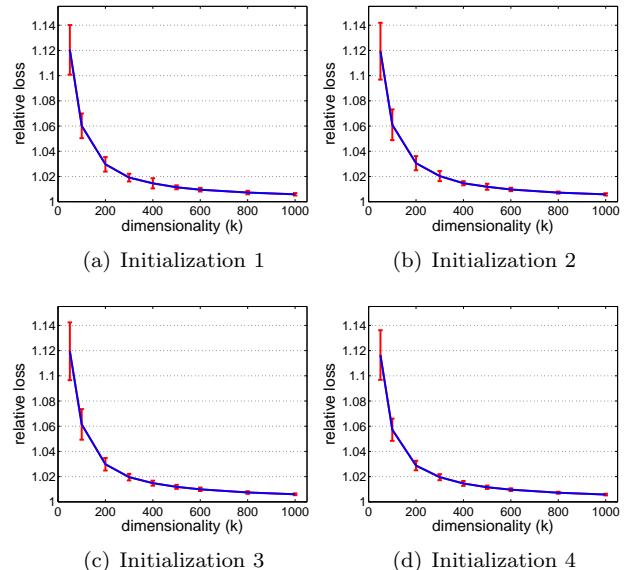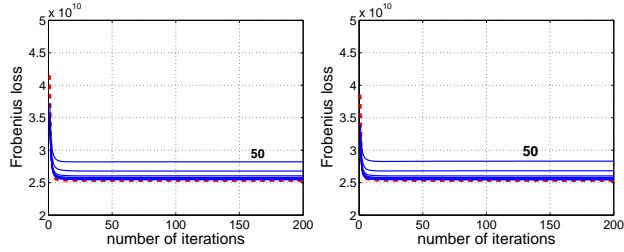

Figure 3: **Relative loss** on **Harvard Microarray** data using **Multiplicative Semi-NMF** using 4 different random initializations of **G**. The y-axis corresponds to the final relative loss after 500 multiplicative update iterations, and the x-axis represents different projected dimensions k (50 to 1000). The solid lines are averaged 100 independent runs with the standard deviation shown as error bars.
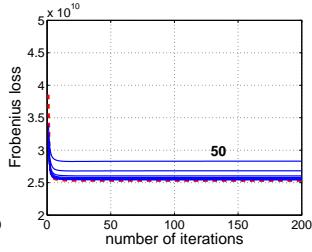
Fig.3 shows the final (after $T = 500$ iterations) relative loss (averaged over 100 independent runs with standard deviation shown as error bars) vs. projected dimensionality. Here, the relative loss after T iterations at a specific projected dimension is computed as

$$(4.32) \qquad r(T) = J'(\mathbf{F}'^{(T)}, \widetilde{\mathbf{G}}^{(T)})/J(\mathbf{F}^{(T)}, \mathbf{G}^{(T)})$$
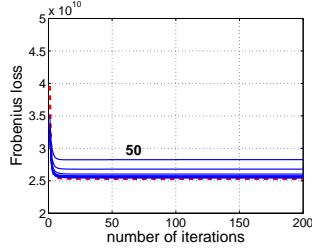
Clearly, the closer $r(T)$ to 1, the better the approximation will be. From Fig.3 we can see that using more projected dimensions will lead to better and more stable approximations, and the gap between $J'(\mathbf{F}'^{(T)}, \widetilde{\mathbf{G}}^{(T)})$ ($T = 500$, $k = 1000$) and the original $J(\mathbf{F}^{(T)}, \mathbf{G}^{(T)})$ is very small (less than $0.5\% \times J(\mathbf{F}^{(T)}, \mathbf{G}^{(T)})$).
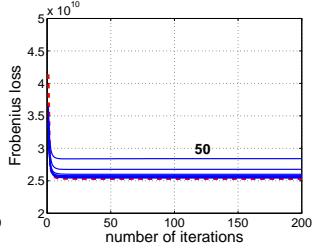
(a) Initialization 1   (b) Initialization 2
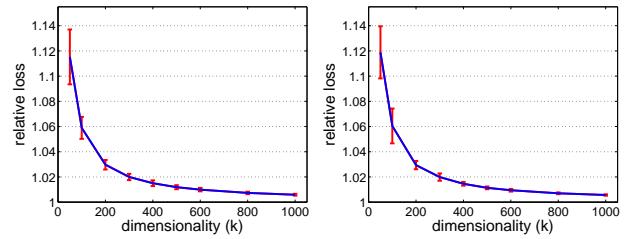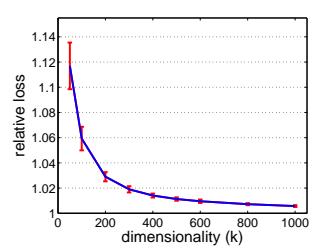
(c) Initialization 3   (d) Initialization 4

Figure 4: **Frobenius loss** over **200** iterations on **Harvard Microarray** data using **Active Set Semi-NMF** with 4 different random initializations of **G**. Dotted line is the plot of original semi-NMF. Solid lines are the averaged (over 100 independent runs) plots of semi-NMF with random projections ($k = 50$ to 1000 from top to bottom).
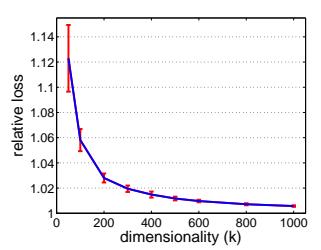

(a) Initialization 1   (b) Initialization 2
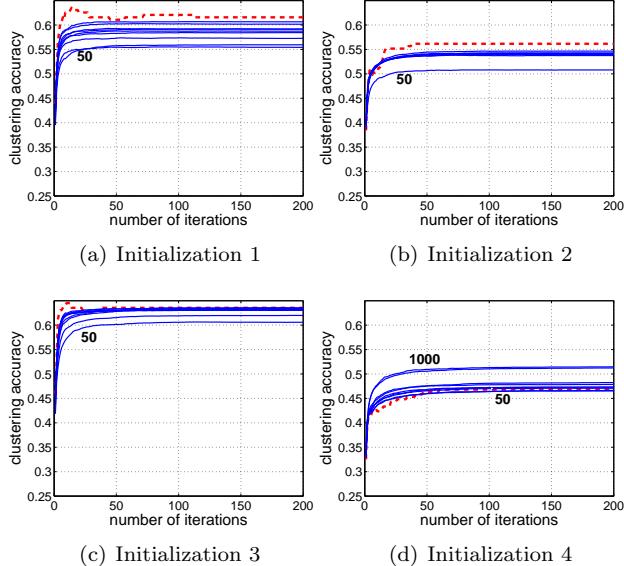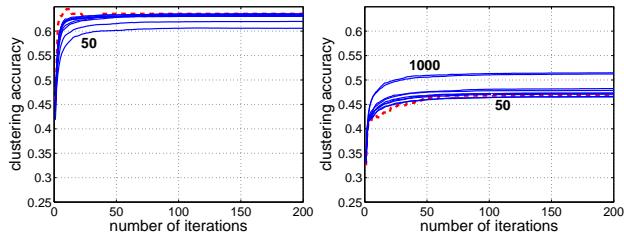
(c) Initialization 3   (d) Initialization 4

Figure 6: **Relative loss** on **Harvard Microarray** data using **Active Set Semi-NMF** with 4 different random initializations of **G**. The y-axis denotes the final relative loss after 500 alternative nonnegative least squares iterations, and the x-axis represents different projected dimensions k (50 to 1000). The solid lines are averaged 100 independent runs with the standard deviation shown as error bars.

Fig.4 to Fig.6 demonstrate the performances of semi-NMF (with/without random projections) using the active set method [20] (referred to as **Active Set semi-NMF**). We use the same initializations as in multiplicative semi-NMF methods. Comparing Fig.4 with Fig.1 confirms that the active set method converges much faster than the multiplicative method, as claimed in [20]. Usually less than 50 steps is enough for the Harvard microarray data set. Moreover, Fig.4 and Fig.6 demonstrate that our random projection method can perform sufficiently well after $k = 600$ as the relative Frobenius loss is less than 0.5%.

(a) Initialization 1   (b) Initialization 2

(c) Initialization 3   (d) Initialization 4

Figure 5: **Clustering accuracy** variation over **200** iterations on the **Harvard Microarray** data using **Active Set Semi-NMF** with 4 different random initializations of **G**. The dotted line is the plot of original semi-NMF. The solid lines are the averaged (over 100 independent runs) plots of semi-NMF with random projections ($k = 50$ to 1000).

4.1.2 NMF with Random Projections We test the performance of NMF algorithm with random projections. The NMF algorithm with multiplicative updates (which is referred to as **Multiplicative NMF**) and alternating least squares using active set [20] (which is abbreviated as **Active Set NMF**) are both tested.

For multiplicative NMF, we randomly initialize **F** and **G** and perform updates for 500 iterations. Note that there are two types of rules for multiplicative NMF, i.e., using Lee and Seung's rules Eq.(3.21) and Eq.(3.20) [21] (denoted as **LS Multiplicative NMF**) or the

square root rules Eq.(3.25) and Eq.(3.24) (denoted as **Square-Root Multiplicative NMF**). For the convenience of comparison, we use the same initializations for different methods. For **active set NMF**, we first randomly initialize $\mathbf{G}$ (using the same initializations as in multiplicative NMF), and then alternatingly solve problem (3.27) and problem (3.26) for 200 iterations.

We first compare the convergence rates of the above 3 different NMF methods in Fig.7, which suggests that **active set NMF** converges the fastest, followed by **LS multiplicative NMF** (which is also consistent with the claim in [20]). The **square-root multiplicative NMF** converges the slowest.

(a) Initialization 1   (b) Initialization 2

(c) Initialization 3   (d) Initialization 4
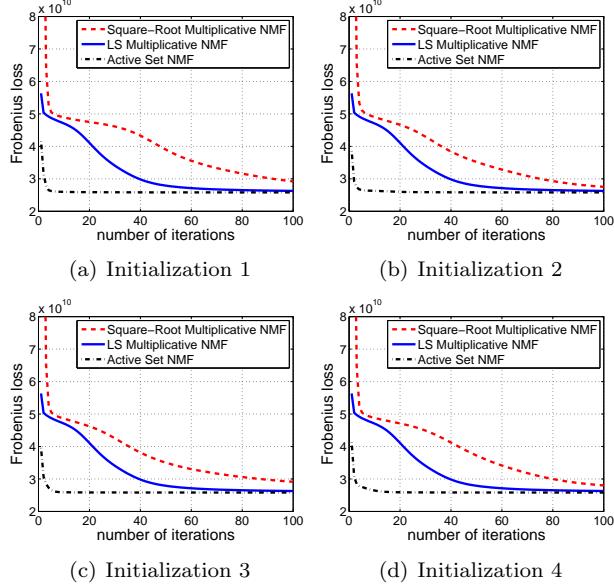
Figure 7: Comparisons of **Frobenius norm** variations of different **NMF** algorithms on **Harvard Microarray** data. Curves in the same figure use the same initializations.

We also test the performances of multiplicative and active set NMF with random projections methods, using the same initializations as in original NMF algorithms. In our experiments, we only reduce the scale of the problem when solving $\mathbf{G}$, and leave the problem of solving $\mathbf{F}$ in its original scale. This is because this gene data set is highly imbalanced, where there are only 203 samples but with dimension 12600. Since $\mathbf{G}$ is only with size 203×5, we do not need to compress it when solving $\mathbf{F}$. As mentioned in Section 3.2, applying Eq.(3.28) and Eq.(3.29) iteratively to update $\mathbf{G}$ and $\mathbf{F}$ iteratively can be viewed as an approximation of the square-root multiplicative NMF algorithm. Therefore we compare our multiplicative NMF with random projection algorithm with the original square root multiplicative NMF.

(a) Initialization 1   (b) Initialization 2

(c) Initialization 3   (d) Initialization 4
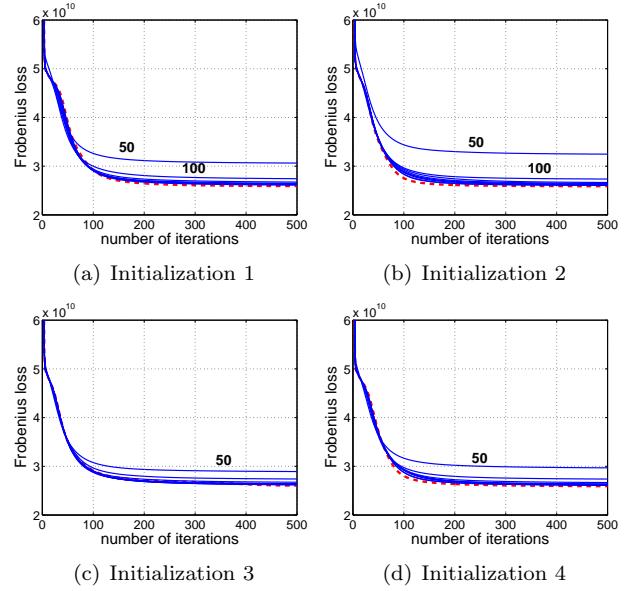
Figure 8: **Frobenius loss** variation over **500** iterations on the **Harvard Microarray** data set using **Square-Root Multiplicative NMF** with 4 different random initializations of $\mathbf{F}$ and $\mathbf{G}$. The dotted line is the plot of original NMF. The solid lines are the averaged (over 100 independent runs) plots of NMF with random projections ($k_1 = 50$ to 1000 from top to bottom).

(a) Initialization 1   (b) Initialization 2

(c) Initialization 3   (d) Initialization 4
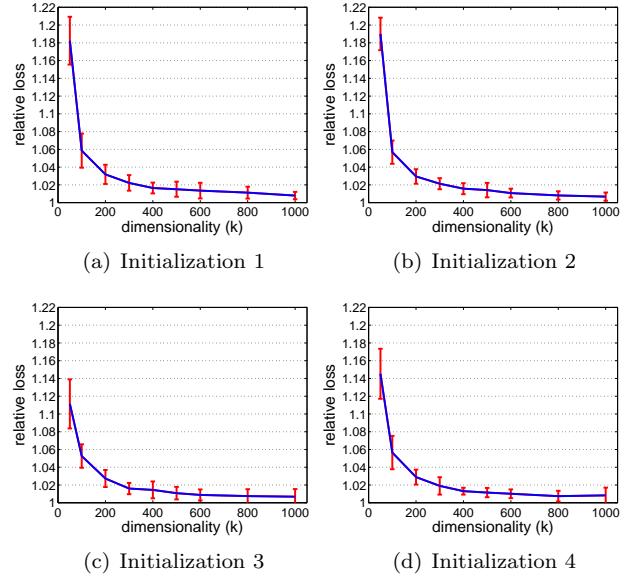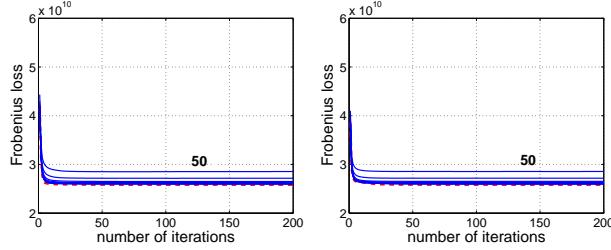
Figure 9: **Relative loss** on **Harvard Microarray** data using **Square-Root Multiplicative NMF** with 4 different random initializations of $\mathbf{G}$. The y-axis corresponds to the final relative loss after 500 multiplicative update iterations, and the x-axis represents different projected dimensions k (50 to 1000). The solid lines are averaged 100 independent runs with the standard deviation shown as error bars.

(a) Initialization 1   (b) Initialization 2

(c) Initialization 3   (d) Initialization 4
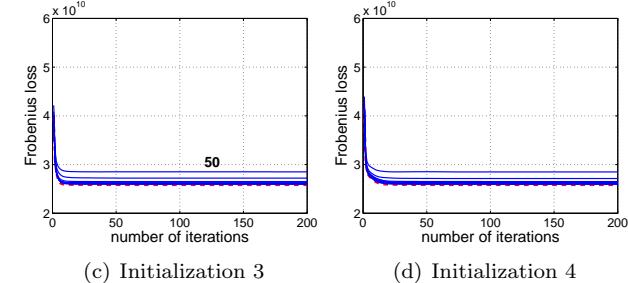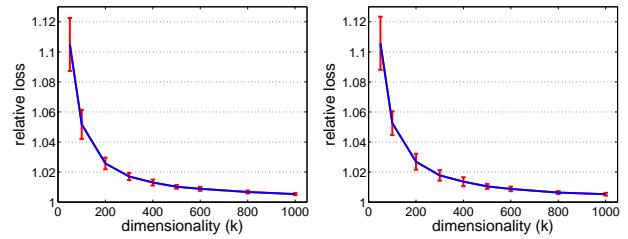
Figure 10: **Frobenius loss** variation over **500** iterations on **Harvard Microarray** data using **Active Set NMF** with 4 different random initializations of $\mathbf{F}$ and $\mathbf{G}$. The dotted line is the plot of original NMF. The solid lines are the averaged (over 100 independent runs) plots of NMF with random projections ($k_1 = 50$ to 1000 from top to bottom).

The Frobenius loss variation is plotted in Fig.8 (where the solid curves are averaged over 100 independent runs), and the relative loss curve is shown in Fig.11. When k_1 approaches 1000 (usually $k_1 = 500$ is enough), the performance would be quite close for multiplicative NMF with and without random projections. The same conclusion can also be drawn from the results of active set NMF with/without random projections, which are shown in Fig.10 and Fig.9. Moreover, comparing Fig.9 with Fig.11, we can find that the active set NMF with random projection method is more stable as the standard deviation is smaller.

4.2   Gisette Data Set Gisette [15] is a handwritten digit recognition problem. The task is to separate the highly confusable digits "4" and "9". The digits have been size-normalized and centered. The original data were modified and the final data set contains 3000 positive examples and 3000 negative examples with dimension 5000.

4.2.1   Semi-NMF with Random Projections Similar to the experiments in the last section, we first compare the performances of the multiplicative/active set semi-NMF algorithm with and without random projections under 2 different random initializations of $\mathbf{G}$. The Frobenius norm variations and relative loss are

(a) Initialization 1   (b) Initialization 2
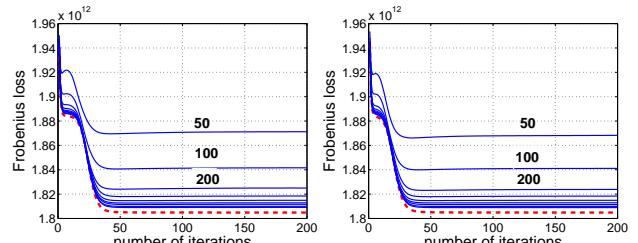

(c) Initialization 3   (d) Initialization 4

Figure 11: **Relative loss** on the **Harvard Microarray** data set using **Active Set NMF** with 4 different random initializations of $\mathbf{G}$. The y-axis corresponds to the final relative loss after 500 multiplicative update iterations, and the x-axis represents different projected dimensions k (50 to 1000). The solid lines are averaged 100 independent runs with the standard deviation shown as error bars.

shown in Fig.12 and Fig.13 for multiplicative semi-NMF with random projections, and Fig.14, Fig.15 for active set semi-NMF with random projections (after 200 iterations). The solid random projection related curves are averaged over 100 independent runs. From the figures we can also observe that the semi-NMF algorithms with random projections would perform very close to the original algorithms when $k \geqslant 500$.


(a) Initialization 1   (b) Initialization 2

Figure 12: **Frobenius loss** variation over **200** iterations on the **Gisette** data set using **Multiplicative Semi-NMF** with 2 different random initializations of $\mathbf{G}$. Dotted line is the plot of original semi-NMF. Solid lines are the averaged (over 100 independent runs) plots of semi-NMF with random projections ($k = 50$ to 1000 from top to bottom).

(a) Initialization 1   (b) Initialization 2
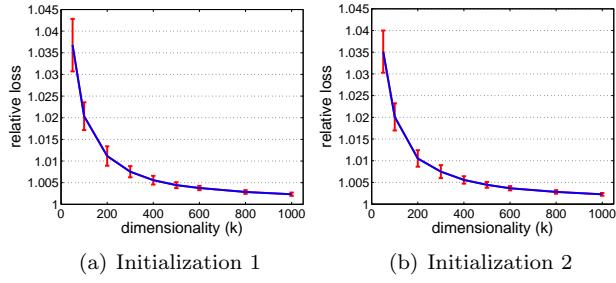
Figure 13: **Relative loss** on the **Gisette** data set using **Multiplicative Semi-NMF** with 2 different random initializations of **G**. The y-axis denotes the final relative loss after 200 multiplicative update iterations, and the x-axis represents different projected dimensions k (50 to 1000). The solid lines are averaged 100 independent runs with the standard deviation shown as error bars.


(a) Initialization 1   (b) Initialization 2
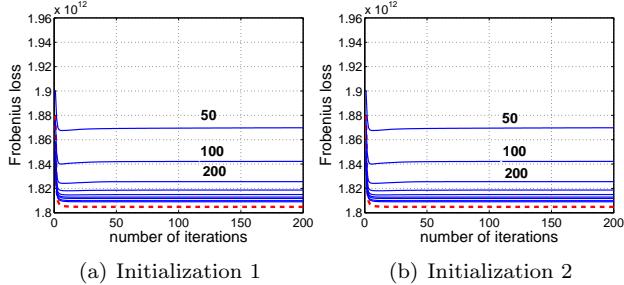
Figure 14: **Frobenius loss** variation over **200** iterations on the **Gisette** data set using **Active Set Semi-NMF** with 2 different random initializations of **G**. The dotted line is the plot of original semi-NMF. The solid lines are the averaged (over 100 independent runs) plots of semi-NMF with random projections ($k = 50$ to 1000 from top to bottom).


(a) Initialization 1   (b) Initialization 2
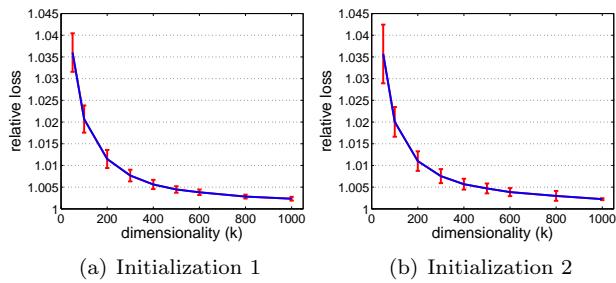
Figure 15: **Relative loss** on the **Gisette** data set using **Active Set Semi-NMF** with 2 different random initializations of **G**. The y-axis denotes the final relative loss after 200 multiplicative update iterations, and the x-axis represents different projected dimensions k (50 to 1000). The solid lines are averaged 100 independent runs with the standard deviation shown as error bars.

4.2.2 NMF with Random Projections We perform the same testing of NMF series algorithms on Gisette data as on Harvard microarray data in Section 4.1.2. Fig.16 compares the convergence rates of different NMF algorithms (over 100 iterations), and the result is consistent with the results in Fig.7. For the random projection based methods, we set $k_1 = k_2 = k$ for simplicity, i.e., the column dimensionality are set to the same when updating **G** and **F**. Fig.17 and Fig.18 show the Frobenius loss variation of the square-root multiplicative NMF and active set NMF with random projection method. We can also observe that using $k \geqslant 500$ projections would be enough for the random projection method to perform sufficiently well as their original counterparts.

(a) Initialization 1   (b) Initialization 2
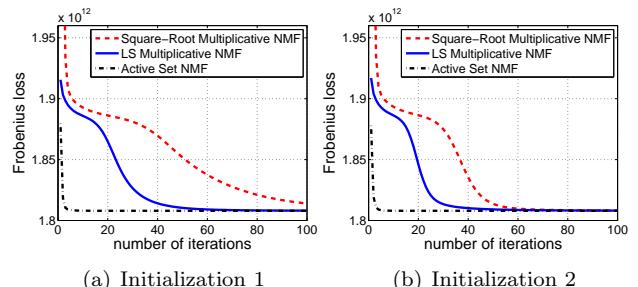
Figure 16: Comparisons of **Frobenius norm** variations of different **NMF** algorithms on **Gisette** data over 100 iterations. Curves in the same figure use the same initializations.

(a) Initialization 1   (b) Initialization 2
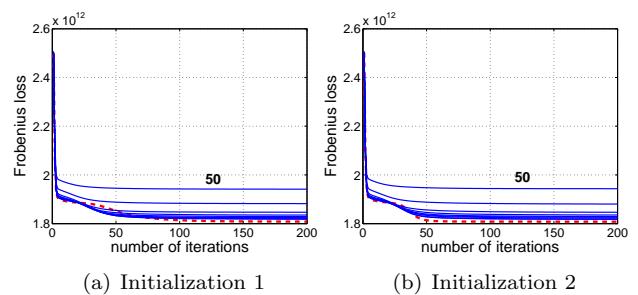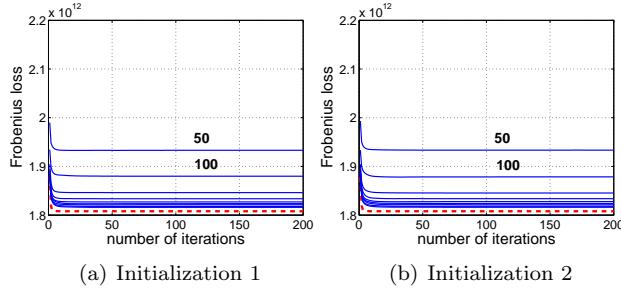
Figure 17: **Frobenius loss** variation over **200** iterations on the **Gisette** data set using **Square-Root Multiplicative NMF** with 2 different random initializations of **G**. Dotted line is the plot of original NMF. Solid lines are the averaged (over 100 independent runs) plots of NMF with random projections ($k_1 = k_2 = k = 50$ to 1000 from top to bottom).
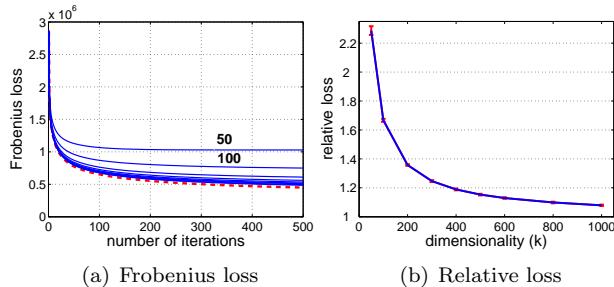
(a) Initialization 1　　　　(b) Initialization 2

Figure 18: **Frobenius loss** variation over **200** iterations on the **Gisette** data set using **Active Set NMF** with 2 different random initializations of **G**. The dotted line is the plot of original NMF. The solid lines are the averaged (over 100 independent runs) plots of NMF with random projections ($k_1 = k_2 = 50$ to 1000 from top to bottom).

4.3 COIL Data Set COIL-100 [31] is an object recognition data set, which contains the pictures of 100 different objects. Each object has 72 pictures taken from different angles. All pictures are of size 128x128, with a total of 16384 pixels.
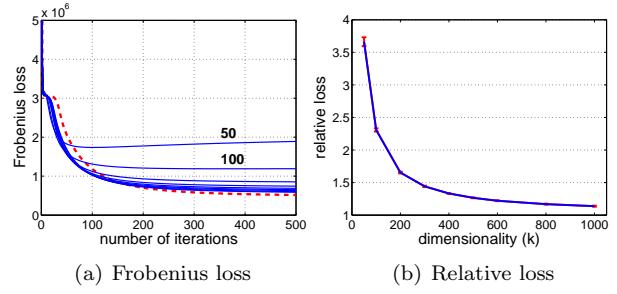
We only tested multiplicative semi-NMF and NMF with/without random projections on this data set, as we found that the active set method is too time consuming on this data set. Fig.19 and Fig.20 show the results of Frobenius loss variation and relative loss of semi-NMF and NMF methods respectively with one random initialization of **F** and **G**. From the figures we can see that even for this large data set, random projection with $k = 500$ (for NMF with random projection, we also set $k_1 = k_2 = k$) can still generate satisfactory results.


(a) Frobenius loss　　　　(b) Relative loss

Figure 19: **Frobenius loss** variation over **500** iterations and **relative loss** on the **COIL** data set using **Multiplicative Semi-NMF** with **G** randomly initialized. The projected dimensions $k_1 = k_2 = k$ (50 to 1000). The solid lines are averaged 100 independent runs.


(a) Frobenius loss　　　　(b) Relative loss

Figure 20: **Frobenius loss** variation over **500** iterations and **relative loss** on the **COIL** data set using **Square-Root Multiplicative NMF** with **G** randomly initialized. The projected dimensions $k_1 = k_2 = k$ (50 to 1000). The solid lines are averaged 100 independent runs.

5 Conclusions and Discussions

In this paper we propose to apply the random projection strategy to make nonnegative matrix factorization more efficient. Extensive experimental results are provided to validate the effectiveness of the proposed strategy.

However, methods based on random projections do not take into account data sparsity, while many real-world large-scale data sets are highly sparse. We are currently exploring sampling/sketching methods (e.g., Conditional Random Sampling (CRS) [25][26]) which were specifically designed for sparse data.

Acknowledgement

Appendix I: Proof of Theorem 2.2

Using Corollary 2.1, let $\xi = \epsilon/3$, we have that with the probability of at least $1 - e^{-(\xi^2 - \xi^3)\frac{k}{4}}$

$$(1-\xi)J(\mathbf{F}^{(0)}, \mathbf{G}_{opt}) \leqslant \widetilde{J}(\widetilde{\mathbf{F}}^{(0)}, \mathbf{G}_{opt}) \leqslant (1+\xi)J(\mathbf{F}^{(0)}, \mathbf{G}_{opt})$$

$$(1-\xi)J(\mathbf{F}^{(0)}, \widetilde{\mathbf{G}}_{opt}) \leqslant \widetilde{J}(\widetilde{\mathbf{F}}^{(0)}, \widetilde{\mathbf{G}}_{opt}) \leqslant (1+\xi)J(\mathbf{F}^{(0)}, \widetilde{\mathbf{G}}_{opt})$$

Therefore

$$J(\mathbf{F}^{(0)}, \widetilde{\mathbf{G}}_{opt}) \leqslant \frac{1}{1-\xi}\widetilde{J}(\widetilde{\mathbf{F}}^{(0)}, \widetilde{\mathbf{G}}_{opt})$$

$$\leqslant \frac{1}{1-\xi}\widetilde{J}(\widetilde{\mathbf{F}}^{(0)}, \mathbf{G}_{opt}) \leqslant \frac{1+\xi}{1-\xi}J(\mathbf{F}^{(0)}, \mathbf{G}_{opt})$$

$$\leqslant (1+3\xi)J(\mathbf{F}^{(0)}, \mathbf{G}_{opt}) \leqslant (1+\epsilon)J(\mathbf{F}^{(0)}, \mathbf{G}_{opt}) \qquad \square$$

References

[1] D. Achlioptas. Database-friendly random projections: Johnson-lindenstrauss with binary coins. *Journal of Computer and System Sciences*, 66(4):671–687, 2003.

[2] R. I. Arriaga and S. Vempala. An algorithmic theory of learning: Robust concepts and random projection. In *FOCS*, 1999.

[3] M. H. Van Benthem and M. R. Keenan. Fast algorithm for the solution of large-scale non-negativity constrained least squares problems. *J. Chemometrics*, 18:441–450, 2004.

[4] M. W. Berry, M. Browne, A. N. Langville, P. V. Pauca, and R. J. Plemmons. Algorithms and applications for approximate nonnegative matrix factorization. *Comput. Stat. & Data Analysis*, 52(1):155–173, 2007.

[5] D. P. Bertsekas. *Nonlinear Programming*. Belmont, MA, 1999.

[6] A. Bhattacharjee, W. G. Richards, J. Staunton, and et al. Classification of human lung carcinomas by mrna expression profiling reveals distinct adenocarcinoma subclasses. *Proceedings of National Academy of Sciences*, 98(24):13790–13795, 2001.

[7] E. Bingham and H. Mannila. Random projection in dimensionality reduction: applications to image and text data. In *SIGKDD*, pages 245–250, 2001.

[8] P. Drineas C. Boutsidis. Random projections for the nonnegative least-squares problem. *Linear Algebra and Its Applications*, 431:760–771, 2009.

[9] G. Chen, F. Wang, and C. Zhang. Collaborative filtering using orthogonal nonnegative matrix trifactorization. *Journal of Information Processing and Management*, 45(3):368–379, 2009.

[10] P. Cui, F. Wang, L. Sun, and S.-Q. Yang. A joint matrix factorization approach to unsupervised action categorization. In *ICDM*, pages 767–772, 2008.

[11] K. Devarajan. Nonnegative matrix factorization: An analytical and interpretive tool in computational biology. *PLoS Comput Biol*, 4(7):e1000029+, 2008.

[12] C. Ding, T. Li, and M. I. Jordan. Convex and seminonnegative matrix factorizations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2009.

[13] G. H. Golub and C. F. Van Loan. *Matrix Computations, 3rd ed.* Johns Hopkins, 1996.

[14] D. Guillamet, M. Bressan, and J. Vitrià. A weighted non-negative matrix factorization for local representations. In *CVPR*, pages 942–947, 2001.

[15] I. Guyon, S. Gunn, M. Nikravesh, and L. Zadeh (Eds.). *Feature Extraction, Foundations and Applications*. Studies in Fuzziness and Soft Computing. Physica-Verlag, Springer., 2006.

[16] C. Hegde, M. B. Wakin, and R. G. Baraniuk. Random projections for manifold learning. In *Advances in Neural Information Processing Systems*, 2007.

[17] P. Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *Journal of the ACM*, 53(3):307–323, 2006.

[18] I. T. Jolliffe. *Principal Component Analysis, 2nd ed.* Springer, 2002.

[19] H. Kim and H. Park. Sparse non-negative matrix factorizations via alternating non-negativity-constrained least squares for microarray data analysis. *Bioinformatics*, 23(12):1495–1502, 2007.

[20] H. Kim and H. Park. Nonnegative matrix factorization based on alternating nonnegativity constrained least squares and active set method. *SIAM Journal on Matrix Analysis and Applications*, 30(2):713–730, 2008.

[21] D. D. Lee and H. S. Seung. Learning the parts of objects by nonnegative matrix factorization. *Nature*, 401:788–791, 1999.

[22] P. Li. Very sparse stable random projections for dimension reduction in l_α $(0 < \alpha \leqslant 2)$ norm. In *SIGKDD*, pages 440–449, 2007.

[23] P. Li. Computationally efficient estimators for dimension reductions using stable random projections. In *ICDM*, pages 403–412, 2008.

[24] P. Li. Improving compressed counting. In *UAI*, 2009.

[25] P. Li and K. W. Church. Using sketches to estimate associations. In *HLT/EMNLP*, pages 708–715, 2005.

[26] P. Li, K. W. Church, and T. Hastie. One sketch for all: Theory and application of conditional random sampling. In *Advances in Neural Information Processing System*, pages 953–960, 2008.

[27] P. Li, T. Hastie, and K. W. Church. Very sparse random projections. In *SIGKDD*, pages 287–296, 2006.

[28] S. Z. Li, X. W. Hou, H. J. Zhang, and Q. S. Cheng. Learning spatially localized, parts-based representation. In *CVPR*, pages 207–212, 2001.

[29] C. J. Lin. Projected gradient methods for nonnegative matrix factorization. *Neural Computation*, 19(10):2756–2779, 2007.

[30] O. A. Maillard and R. Munos. Compressed least square regression. In *Advances in Neural Information Processing Systems*, 2009.

[31] S. A. Nene, S. K. Nayar, and H. Murase. Columbia object image library (coil-100). *Technical Report CUCS-006-96*, 1996.

[32] Q. Qi, Y. Zhao, M. Li, and R. Simon. Non-negative matrix factorization of gene expression profiles: a plug-in for brb-arraytools. *Bioinformatics*, 25(4):545–547, February 2009.

[33] J. Le Roux, A. de Cheveigne, and L. C. Parra. Adaptive template matching with shift-invariant semi-nmf. In *Advances in Neural Information Processing Systems*, pages 921–928. MIT Press, 2008.

[34] S. S. Vempala. *The Random Projection Method*, volume 65. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 2004.

[35] F. Wang, T. Li, and C. Zhang. Semi-supervised clustering via matrix factorization. In *SDM*, pages 1–12, 2008.

[36] W. Xu, X. Liu, and Y. Gong. Document clustering based on non-negative matrix factorization. In *SIGIR*, pages 267–273, 2003.