AFRL-AFOSR-JP-TR-2017-0050

Dimensionality Reduction in Big Data with Nonnegative Matrix Factorization

Tu-Bao Ho
JAPAN ADVANCED INSTITUTE OF SCIENCE AND TECHNOLOGY

06/20/2017
Final Report

Air Force Research Laboratory
AF Office Of Scientific Research (AFOSR)/ IOA
Arlington, Virginia 22203
Air Force Materiel Command

FORM SF 298

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From - To)*
22-06-2017	Final	08 Apr 2015 to 07 Apr 2017

4. TITLE AND SUBTITLE
Dimensionality Reduction in Big Data with Nonnegative Matrix Factorization

5a. CONTRACT NUMBER

5b. GRANT NUMBER
FA2386-15-1-4006

5c. PROGRAM ELEMENT NUMBER
61102F

6. AUTHOR(S)
Tu-Bao Ho

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
JAPAN ADVANCED INSTITUTE OF SCIENCE AND TECHNOLOGY
1-1, ASAHIDAI
NOMI, ISHIKAWA, 923-1211 JP

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AOARD
UNIT 45002
APO AP 96338-5002

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/AFOSR IOA

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
AFRL-AFOSR-JP-TR-2017-0050

12. DISTRIBUTION/AVAILABILITY STATEMENT
A DISTRIBUTION UNLIMITED: PB Public Release

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Nonnegative matrix factorization (NMF) is a linear powerful technique for dimension reduction. It reduces the dimensions of data making learning algorithms faster and more effective. Although NMF and its applications have been developed for more than a decade, they still have limitations in modeling and performance. In this study, the PI's team designed rich models and fast algorithms for NMF. They found solutions for four different problems: 1. Accelerated parallel and distributed algorithm for NMF with Frobenius norm with L1 L2 regularizations. The algorithm is based on the proposed accelerated anti-lopsided algorithm for nonnegative least squares. It attained fast overbounded guaranteed convergence in the space of passive variables, where convex parameter and Lipschitz constant L are bounded. 2. Fast parallel randomized algorithm NMF with Kullback-Leibler divergence. The proposed algorithm has fast convergence, and utilize the sparse properties of data, model and representation. In addition, the experiments indicate that sparse models and sparse representation are archived for large sparse datasets, which is a significant milestone in this research problem.
3. New models of simplicial NMF and simplicial nonnegative matrix tri-factorization with Frobenius norm and fast parallel algorithm. The new models have more concise interpretability via these values of factor matrices. The proposed algorithms based on a combination of alternating direction and Frank-Wolfe's scheme attain linear convergence, low iteration complexity, and easily controlled sparsity. The experiments indicate that the proposed model and algorithm outperform the NMF model and its state-of-the-art algorithms.
4. New model of simplicial NMF with Kullback-Leibler divergence and fast parallel algorithm. The proposed algorithm as guaranteed instance inference with sub-linear convergence, and easy sparsi

15. SUBJECT TERMS
Machine Learning

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			KNOPP, JEREMY
Unclassified	Unclassified	Unclassified	SAR	38	**19b. TELEPHONE NUMBER** *(Include area code)* 315-227-7006

Final Report for AOARD Grant FA2386-15-1-4006 "Dimensionality Reduction in Big Data with Nonnegative Matrix Factorization"

Name of Principal Investigator: Tu Bao Ho

Date: April 6, 2017

E-mail: bao@jaist.ac.jp

Institution: Japan Advanced Institute of Science and Technology

Mailing Address: JAIST, 1-1 Asahidai, Nomi City, Ishikawa, 923-1292 Japan

Phone and Fax: 81-761-51-1730

Period of Performance: 4/1/2015 - 3/31/2017

Abstract

Nonnegative matrix factorization (NMF) is a linear powerful technique for dimension reduction. It reduces dimension of data making learning algorithms faster and more effective. Although NMF and its applications have been developed for more than a decade, they still have limitations in modeling and performance.

In this study, we design rich models and fast algorithms for NMF. We have found solutions for the following four problems:

1. Accelerated parallel and distributed algorithm for NMF with Frobenius norm with L_1 L_2 regularizations. The algorithm is based on the proposed accelerated anti-lopsided algorithm for nonnegative least squares. It attained fast overbounded guaranteed convergence $O([(1-\frac{\mu}{L})(1-\frac{\mu}{rL})^{2r}]^k)$ in the space of passive variables, where convex parameter μ and Lipschitz constant L are bounded as $\frac{1}{2} \leq \mu \leq L \leq r$.

2. Fast parallel randomized algorithm NMF with Kullback-Leibler divergence with L_1 L_2 regularizations. The proposed algorithm has fast convergence, and utilize the sparse properties of data, model and representation. In addition, the experiments indicate that sparse models and sparse representation are archived for large sparse datasets, which is a significant milestone in this research problem.

3. New models of simplicial NMF and simplicial nonnegative matrix tri-factorization with Frobenius norm and fast parallel algorithm. The new models have more concise interpretability via these values of factor matrices. The proposed algorithms based on a combination of alternating direction and Frank-Wolfe's scheme attain linear convergence $O(1/k)$, low iteration complexity, and easily controlled sparsity. The experiments indicate that the proposed model and algorithm outperform the NMF model and its state-of-the-art algorithms.

4. New model of simplicial NMF with Kullback-Leibler divergence and fast parallel algorithm. The proposed algorithm has guaranteed instance inference with sublinear convergence $O(1/k)$, and easy sparsity control. The experiments indicate that this approach can achieve high sparse representation with higher accuracy in comparison with similar approaches.

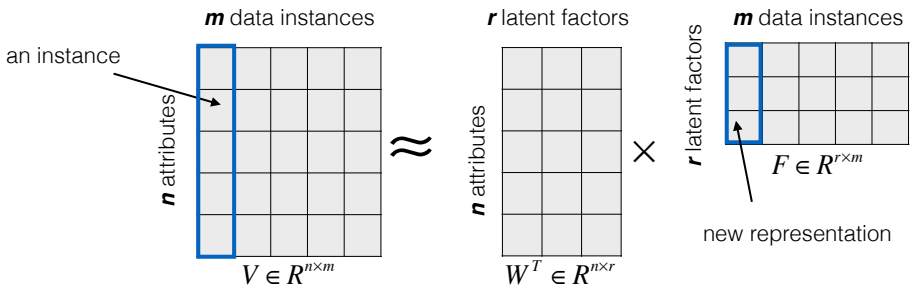

Figure 1: Nonnegative matrix factorization

1. Introduction

1.1 Nonnegative matrix factorization

Nonnegative matrix factorization (NMF) is a powerful technique widely used in applications of data mining, signal processing, computer vision, bioinformatics, etc. Fundamentally, NMF has two main purposes. First, it reduces dimension of data making learning algorithms faster and more effective as they often work less effectively due to the curse of dimensionality. Second, NMF helps extracting latent components and learning part-based representation, which are the significant distinction from other dimension reduction methods such as Principal Component Analysis (PCA), Independent Component Analysis (ICA), Vector Quantization (VQ), etc. This feature originates from transforming data into lower dimension of latent components and non-negativity constraints.

Mathematically, nonnegative matrix factorization (NMF), see Figure 1, is formulated as follows:

Definition 1 *Given a dataset consisting of m vectors in n dimensions $V = [V_1, V_2, ..., V_m] \in R_+^{n \times m}$, where each vector presents a data instance. NMF seeks to approximately factorize V into a product of two nonnegative factorizing matrices W^T and F, where $W \in R_+^{r \times n}$ and $F \in R_+^{r \times m}$ are coefficient matrix and latent component matrix, respectively, $V \approx W^T F$.*

In another mathematical way of understanding NMF, the factorization is the transformation of vectors in dataset V to a new space of vectors in W^T to obtain new representation F. Specifically, determining F is considered the projection of datasets into a new space of latent factors W^T. $W^T F$ is the result of projecting back into the original space of new coefficients F.

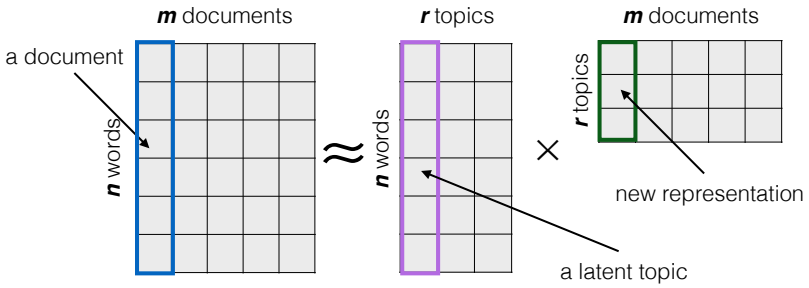

Figure 2: A collection of documents represented by weighted words can be analyzed as product of two matrices, one is latent factors represented by weighted words and the other is new representation of documents by linear combination of topics

Furthermore, the whole of information in the original space can not be kept because m, n is often bigger than r. To retain the properties of V, the factorization needs to guarantee having the least information loss via minimizing an objective function $D(V\|W^TF) = \sum_{i=1}^{n}\sum_{j=1}^{m} d(V_{ij}\|[W^TF]_{ij})$; where $d(x\|y)$ can be considered a function to measure the difference between x and y.

In practice, many functions are employed depending on the properties of datasets. For examples, Frobenius norm is suitable for image datasets, KullbackLeibler (KL) divergence is employed for count sparse datasets such as documents, and Itakura-Saito (IS) divergence is more suitble for decomposing a power spectrogram in musical applications. In this research, we focus on two of the most popular divergences:

- Frobenius norm: $J(V\|W^TF) = \|V - W^TF\|_2^2 = \sum_{i=1}^{n}\sum_{1}^{m}(V_{ij} - W_i^TF_j)^2$
- KL divergence: $J(V\|W^TF) = \sum_{i=1}^{n}\sum_{1}^{m}(V_{ij}\log(\frac{V_{ij}}{W_i^TF_j}) - V_{ij} + W_i^TF_j)$

Regularizations are added to improve the quality of the factorization. Specially, L_1 regularization $(\beta_1\|W\|_1 + \beta_2\|F\|_1)$ is used in the objective function to enhance sparsity, and L_2 regularization $(\alpha_1\|W\|_2^2 + \alpha_2\|F\|_2^2)$ is used to enhance smoothness.

1.2 Motivation of our work

Although nonnegative matrix factorization and its applications have been developed for more than a decade, there remains many open problems for further studies. Some of the major challenges can be enumerated as follows:

Challenge 1: Rich models. Nonnegative matrix factorization is an ill-posed inverse problem. In other words, many local optimal solutions, called as stationary points,

exist. Hence, rich models are needed that can concisely interpret the relationship observed data and latent factors, and have the high quality of sparsity and smoothness. Rich models will specially describe latent factors and new coefficients to enhance the factorization results. In addition, model-based approach using a concise specification language to describe NMF provides significant advantages in creating highly tailored models for specific scenarios, rapidly prototyping and easily understanding various models [1].

Challenge 2: Fast convergence and low complexity. In practice, multiple iterative algorithms like EM algorithms requiring a number of iterations are employed for NMF because NMF is often an ill-posed inverse problem and its objective function depends on separated sets of variables W^T and F. The number of used iterations and the iteration complexity mainly influence the speed of algorithm performance. In addition, NMF is a non-convex optimization problem, yet finding new W and F when fixing one of them is convex. Hence, algorithms having fast guaranteed convergence and low complexity are preferred rather than heuristic unstable ones.

Challenge 3: Parallel and distributed computation. The development of technology and science has been generating huge data matrices, which makes it infeasible to factorize these matrices by a single core or a computer. Hence, the computation must be parallelized and distributed to reduce the running time. The major challenges are how to decompose the computation into small independent computing units, deal with the limited internal memory and various constraints of smoothness, sparsity, and orthogonality. Furthermore, emerging new computing models such as Hadoop and Spark leads to redesign parallel NMF algorithms for distributed computation.

Challenge 4: Sparsity of models and representation. Sparsity is a natural property of data, which can be measured by the number of non-zeros elements. For example, the number of words in a topic and the number of topics in a document should be small. Moreover, sparsity leads to save used internal memory, enhance the speed of algorithms, and reduce over-fitting problems. In nonnegative matrix factorization, algorithms need to attain sparse models W^T, sparse representation in F, and sparse computation that utilizes the sparsity of W^T and F to increase their speed.

In this research, we propose rich models and fast algorithms for nonnegative matrix factorization, see Figure 3. Concerning rich models, two rich models, namely NMF with L_1 L_2 regularizations and simplicial NMF that is extended from [2], are proposed to enhance sparsity, smoothness and interpretability. In comparison with

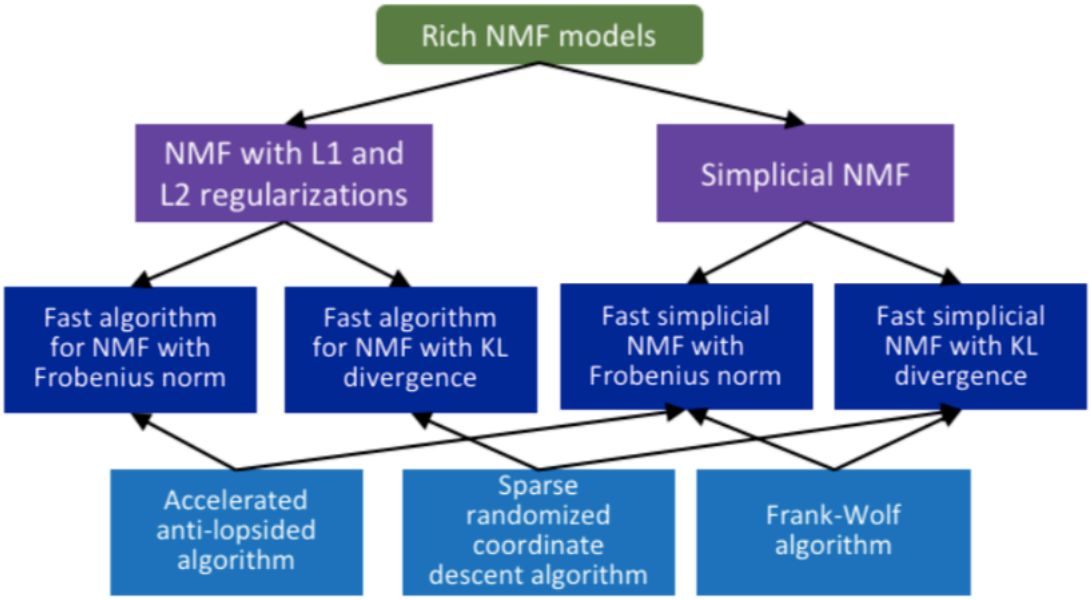

Figure 3: Scheme of our algorithms and their relations

the previous models, the proposed rich models are more general, robust and interpretable to deal with various data. Regarding to fast algorithms, we propose four fast parallel algorithms for four NMF variants of two rich models and two divergences. Furthermore, the proposed algorithms are carefully designed to achieve low complexity and fast convergence to deal with big data.

2. Accelerated parallel and distributed algorithms for NMF with Frobenius norm with L_1 L_2 regularizations

In the NMF with Frobenius norm, the quality of this factorization is controlled by this objective function:

$$D(V\|W^TF) = \|V - W^TF\|_2^2 + \alpha_1\|W\|_2^2 + \alpha_2\|F\|_2^2 + \beta_1\|W\|_1 + \beta_2\|F\|_1$$

To be fully parallel and distributed, the optimization is totally decomposed into non-negative quadratic programming (NQP) problems with the size of r, which is the smallest in comparison with other methods:

$$D(V\|W^TF) = \sum_{j=1}^{m} D(V_j\|W^TF_j) = \sum_{i=1}^{n} D(V_i\|F^TW_i)$$

The proposed method is basically based on iterative multiplicative update accelerated algorithm (Algorithm 1). To adapt to big data, we adjust Algorithm 1 to a parallel and distributed one using limited internal memory (Algorithm 2). This algorithm only requires limited internal memory of F^TF, W^TW and FV^T with the size of $O(r(r+n))$ in computing nodes, where $n \ll m$ for large datasets, and n is limited by the dimension of data.

Algorithm 1: Iterative Multiplicative Update Accelerated Algorithm

Input: Data matrix $V = \{V_i\}_{i=1}^m \in \mathcal{R}_+^{n\times m}$ and r.
Output: Latent components $W \in \mathcal{R}^{r\times n}$.

```
1 begin
2     Randomize r nonnegative latent components W ∈ R_+^{r×n} ;
3     repeat
4         E-step: Fixing W to find F^+ such that the accelerated condition is
            satisfied;
5         M-step: Fixing F to find W^+ such that the accelerated condition is
            satisfied;
6     until Convergence condition is satisfied;
```

The most major challenge of Algorithm 2 is to solve a large number of nonnegative quadratic programming (NQP) problems. Hence, Algorithm 2 employs accelerated anti-lopsided algorithm (Algorithm 3) to swiftly seek the approximate solutions. Specifically, Algorithm 3 contains four main steps:

- Part 1. Anti-lopsided transformation from Line 4 to Line 6: the variable vector x is transformed into a new space by $x = \varphi(y)$ as an inverse function. In the new space, the new equivalent objective function $g(y) = f(\varphi(y))$ has $\frac{\partial^2 g}{\partial y_i^2} = 1,\ \forall i$, or the acceleration of each variable equals 1. As a result, the role of variables become more balanced because the shape of the function becomes more spherical because $\frac{\partial^2 g}{\partial y_i^2} = 1,\ \forall i$, and $g(y)$ is convex. This part aims to make the post-processing parts more effective because it can implicitly exploit the second derivative information $\frac{\partial^2 g}{\partial y_i^2} = 1,\ \forall i$ to guarantee that μ and L are always bounded as $\frac{1}{2} \le \mu \le L \le r$.

- Part 2: Exact line search from Line 11 to Line 13: this part optimizes the objective function with a guarantee of over-bounded convergence rate $\mathcal{O}((1 - \frac{\mu}{L})^k)$ where $\frac{1}{2} \le \mu \le L \le r$ over the space of passive variables, which has a

complexity $\mathcal{O}(r^2)$. The part aims to reduce the objective functions exponentially and precisely, although it suffers from variable scaling problems and nonnegative constraints.

- Part 3. Greedy coordinate descent algorithm from Line 15 to Line 18 and repeated in Line 26: this part employs greedy coordinate descent using Gauss-Southwell rule with exact optimization to rapidly reduce the objective function with fast convergence $\mathcal{O}(1 - \frac{\mu}{rL})$ for each update [3, 4], which has a complexity of $\mathcal{O}(r^2)$. The part aims to reduce negative affects of variable scaling problems and nonnegative constraints, although it has zig-zagging problems because of optimizing the objective function over each single variable. Due to having fast convergence in practice and reducing negative affects of variable scaling problems and nonnegative constraints, this part is repeated one more time after Part 4.

- Part 4. Accelerated search from Line 22 to Line 25: This step performs a momentum search based on previous changes of variables in Part 2 and Part 3, which has a low complexity of $\mathcal{O}(r.nn(r))$ where $nn(r)$ is the number of negative elements in $(x_{k+1} - \alpha \triangle x)$, see Line 25 in Algorithm 3. This part relies on the global information of two distinct points to escape the local optimal information issues of the first derivative raised by the function complexity. This part originates from the idea that if the function is optimized from x_s to x_k by the exact line search and the coordinate descent algorithm, it is highly possible that the function value will be reduced along the vector $(x_k - x_s)$ because the NNLS objective function is convex and has (super) eclipse sharp.

Theorem 2 *The exact line search in Algorithm 3 linearly converges at the rate of* $\mathcal{O}(1 - \frac{\mu}{L})^k$ *in the sub-space of passive variables, where* $\frac{1}{2} \leq \mu \leq L \leq r$, r *is the dimension of solutions or the number of latent factors, and* r *is the number of iterations.*

Theorem 3 *Algorithm 3 has over-bounded linear convergence rate of* $\mathcal{O}([(1 - \frac{\mu}{L})(1 - \frac{\mu}{rL})^{2r}]^k)$ *in the sub-space of passive variables, where* $\frac{1}{2} \leq \mu \leq L \leq r$, $(1 - \frac{\mu}{L})(1 - \frac{\mu}{rL})^{2r} \leq (1 - \frac{1}{2n})(1 - \frac{1}{2rn})^{2r}$, r *is the dimension of solutions or the number of latent factors, and* k *is the number of iterations.*

Theorem 4 *The complexity of each iteration in Algorithm 2 using Algorithm 3 to solve NQP problems is* $\mathcal{O}((m+n)r^2 + mnr + \overline{k}(m+n)r^2)$. *In addition, it is* $\mathcal{O}((m+$

Algorithm 2: Parallel and Distributed Algorithm

Input: Data matrix $V = \{V_j\}_{j=1}^{m} \in \mathcal{R}_+^{nm}$ and r.
Output: Latent components $W \in \mathcal{R}_+^{rn}$.

begin
  Randomize r nonnegative latent components $W \in \mathcal{R}_+^{rn}$;
  repeat
    $Y = 0 \in \mathcal{R}^{rn}$ /* $Y = FV^T$*/;
    $H = 0 \in \mathcal{R}^{rr}$ /* $H = FF^T$ /*;
    $Q = WW^T \in \mathcal{R}^{rr}$;
    $maxStop = 0$;
    /*Parallel and distributed*/;
    for $j = 1$ *to* m **do**
      /*call Algorithm 3*/;
      $F_j \approx \underset{x \in \mathcal{R}^r \succeq 0}{\operatorname{argmin}} (\frac{x^T Q x}{2} - V_j^T W^T x)$;
      $Y = Y + F_j V_j^T$;
      $H = H + F_j F_j^T$;
    /*Parallel and distributed*/;
    for $i = 1$ *to* n **do**
      /*call Algorithm 3*/;
      $W_i \approx \underset{x \in \mathcal{R}^r \succeq 0}{\operatorname{argmin}} (\frac{x^T H x}{2} - {Y_i}^T x)$;
  until *Convergence condition is satisfied*;

Algorithm 3: Fast Combinational Algorithm for NQP

Input: $H \in \mathcal{R}^{r\times r}$ and $h \in \mathcal{R}^r$ and x_0

Output: $x \approx \underset{x\succeq 0}{\operatorname{argmin}} \frac{1}{2}x^T Hx + h^T x$

1 **begin**

2 /*Having a variable $maxStop = 0$ for each thread of computation */;

3 /*Re-scaling variables*/;

4 $Q = \frac{H}{\sqrt{diag(H)diag(H)^T}}$; $q = \frac{h}{\sqrt{diag(H)}}$;

5 /*Solving NQP: minimizing$f(x) = \frac{1}{2}x^T Qx + q^T x$*/;

6 $x = x_0 .* \sqrt{diag(H)}$;

7 $\nabla f = Qx + q$;

8 **repeat**

9 $x_s = x_{k-1}$ and $\nabla f_s = \nabla f$;

10 /*Exact line search over passive variables $P(x) = \{i \mid \nabla_i \bar{f}(x) < 0 \text{ or } \nabla_i \bar{f}(x) > 0 \text{ and } [x]_i > 0\}$*/;

11 $\nabla \bar{f} = \nabla f$; and $\nabla \bar{f}[x = 0 \text{ and } \nabla f > 0] = 0$;

12 $\alpha = \arg\min_{\alpha} f(x_k - \alpha \nabla \bar{f}) = \frac{\|\nabla \bar{f}\|_2^2}{\nabla \bar{f}^T Q \nabla \bar{f}}$;

13 $x_k = x_{k-1} - \alpha \nabla \bar{f}$; $\nabla f_k = \nabla f_k - \alpha Q \nabla \bar{f} - Q[x_k]_-$; $x_k = [x_k]_+$;

14 /*Greedy coordinate descent algorithm*/;

15 **for** *t=1 to n* **do**

16 $p = \underset{i\in P(x)}{\operatorname{argmax}} |\nabla_i f(x_k)|$;

17 $\triangle x_p = max(0, [x_k]_p - \frac{\nabla_p f}{Q_{pp}}) - [x_k]_p$;

18 $\nabla f = \nabla f + Q_p \triangle x_p$; $[x_k]_p = [x_k]_p + \triangle x_p$;

19 **if** $(\|\tilde{f}_k\|_2^2 \le \epsilon \|\tilde{f}_0\|_2^2)$ *or* $(\|\tilde{f}_k\|_2^2 \le maxStop)$ **then**

20 break;

21 /*Accelerated search carries a "momentum" based on the changes of variables in exact line search and greedy coordinate descent part*/;

22 $\triangle x = x_s - x_k$ /*x_s and ∇f_s are assigned in Line 9*/;

23 $\alpha = \underset{\alpha}{\operatorname{argmin}} f(x_k - \alpha \triangle x) = \frac{\nabla f^T \triangle x}{\triangle x^T Q \triangle x} = \frac{\nabla f^T \triangle x}{\triangle x^T (\nabla f_s - \nabla f)}$;

24 $x_k = x_k - \alpha \triangle x$;

25 $\nabla f = \nabla f - \alpha Q \triangle x - Q[x_k]_-$; $x_k = [x_k]_+$;

26 Repeat steps in the part of greedy coordinate descent algorithm;

27 **until** $(\|\tilde{f}_k\|_2^2 \le \epsilon \|\tilde{f}_0\|_2^2)$ *or* $(\|\tilde{f}_k\|_2^2 \le maxStop)$;

28 $maxStop = max(maxStop, \|\tilde{f}_k\|_2^2)$;

29 **return** $\frac{x_k}{\sqrt{diag(H)}}$

Table 1: Dataset Information

Data-sets	m	n	r	MaxIter
Faces	6977	361	60	300
Digits	6.10^4	784	80	300
Tiny Images	5.10^4	3,072	100	300
Nytimes	3.10^5	102,660	100,...,200	300

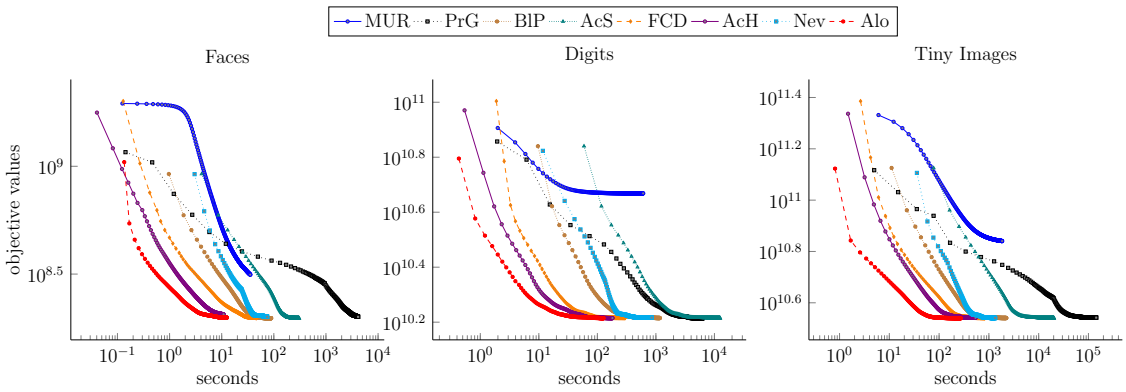

Figure 4: Objective function values $\|V - W^T F\|_2^2/2$ versus CPU seconds

$n)r^2 + rS(mn) + \overline{k}(m+n)r^2)$ *for sparse data, where* $S(mn)$ *is the number of non-zero elements in data matrix* V*.*

The experimental comparative evaluation is done on four datasets (see Table 1), and the proposed algorithm (Alo) is compared seven state-of-the-art belonging different research directions: Multiplicative Update Rule(MUR), Projected Gradient Methods(PrG), Block Principal Pivoting method(BlP), Fast Active-set-like method(AcS), Fast Coordinate Descent methods with variable selection(FCD), Accelerated Hierarchical Alternating Least Squares(AcH), and Nesterov's optimal gradient method(Nev).

Figure 4 and 5 show that the proposed algorithm converges fastest versus both time and iteration. Furthermore, the proposed algorithm attains the best optimal values in two large datasets, and it is stable when it takes only one iteration in average $\bar{k}$ to satisfy the stopping condition. These experiments indicate the strength and robustness of the proposed algorithm, specially for accelerated anti-lopsided algorithm for nonnegative quadratic programming problems.

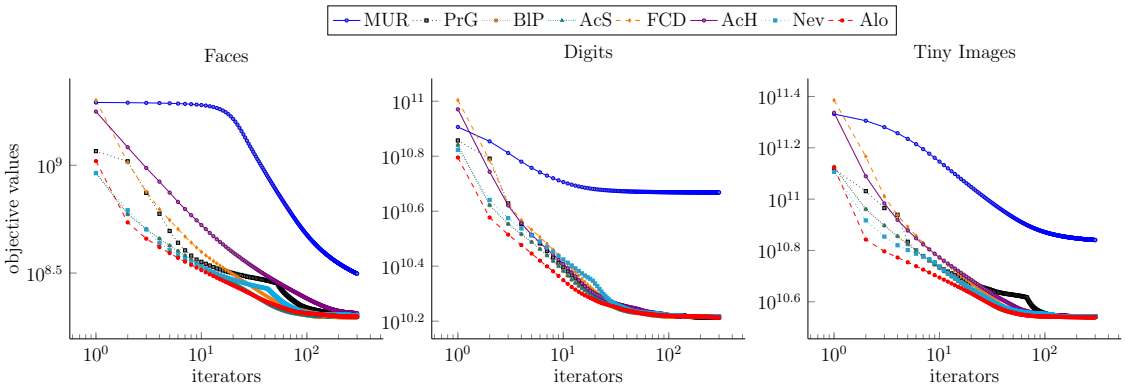

Figure 5: Objective function values $\|V - W^T F\|_2^2/2$ versus the iteration number

Table 2: Optimal Values of NMF solvers

Dataset	MUR	PrG	BlP	AcS	FCD	AcH	Nev	Alo
Faces ($\times 10^8$)	3.142	2.003	**1.975**	**1.975**	1.983	2.058	2.003	1.985
Digits ($\times 10^{10}$)	4.659	1.639	1.641	1.641	1.644	1.640	1.646	**1.639**
Tiny Images ($\times 10^{10}$)	6.925	3.483	3.472	3.472	3.474	3.476	3.473	**3.467**

Table 3: Average of Iteration Number $\bar{k}$

Dataset	MUR	PrG	BlP	AcS	FCD	AcH	Nev	Alo
Faces	1.00	321.12	1116.96	102.09	1.54	1.11	29.21	**1.01**
Digits	1.00	36.70	12503.75	305.94	1.00	1.05	23.36	**1.00**
Tiny Images	1.00	767.45	12869.12	1086.51	1.38	2.52	29.32	**1.02**

3. Fast parallel randomized algorithm NMF with Kullback-Leibler divergence with L_1 L_2 regularizations

In the NMF-KL problem with L_1 L_2 regularizations, the quality of this factorization is controlled by the objective function as follows:

$$D(V\|W^TF) = \sum_{i=1}^{n}\sum_{j=1}^{m}(V_{ij}\log\frac{V_{ij}}{W_i^TF_j} - V_{ij} + W_i^TF_j) + \alpha_1\|W\|_2^2 + \alpha_2\|F\|_2^2 + \beta_1\|W\|_1 + \beta_2\|F\|_1$$

We employ a multiple iterative update algorithm like EM algorithm, see Algorithm 4, because $D(V\|W^TF)$ is a non-convex function although it is a convex function when fixing one of two matrices W and F. This algorithm contain a *while* loop containing two main steps: the first one is to optimize the objective function by F when fixing W; and the another one is to optimize the objective function by W when fixing F. Furthermore, in this algorithm, we need to minimize Function 1, the decomposed elements of which can be independently optimized in Algorithm 4.

Algorithm 4: Iterative multiplicative update

Input: $V \in \mathcal{R}_+^{n\times m}, r$, and $\alpha_1, \alpha_2, \beta_1, \beta_2 \geq 0$
Output: $W \in \mathcal{R}_+^{n\times r}$, $F \in \mathcal{R}_+^{r\times m}$.

begin
  Randomize $W \in \mathcal{R}_+^{r\times n}$;
  Randomize $F \in \mathcal{R}_+^{r\times m}$;
  while *convergence condition is not satisfied* **do**
    ids = a randomized ordered set of values $\{1, 2, ..., r\}$;
    $sumW = W\mathbf{1}^n$;
    /*Optimizing the objective function by F when fixing W*/;
    for *j = 1 to m* **do**
      /*Call Algorithm 5 in parallel*/;
      F_j^{k+1} = Algorithm 5 (V_j, W^T, $sumW$, F_j^k, α_2, β_2, ids)
    $sumF = F\mathbf{1}^m$;
    /*Optimizing the objective function by W when fixing F*/;
    for *i = 1 to n* **do**
      /*Call Algorithm 5 in parallel*/;
      W_i^{k+1} = Algorithm 5 (V_i^T F^T, $sumW$, W_i^k, α_2, β_2, ids);
  return $(W^{k+1})^T$, F^{k+1};

$$D(V\|W^TF) = \sum_{i=1}^{m} D(V_i\|W^TF_i) = \sum_{j=1}^{n} D(V_j^T\|F^TW_j) \tag{1}$$

Specially, a number of optimization problems $D(V_i\|W^TF_i)$ or $D(V_j^T\|F^TW_j)$ in Algorithm 4 can be independently and simultaneously solved by Algorithm 5 in this form:

$$f(x) = D(v\|Ax) = \sum_{i=1}^{n}(v_i \log \frac{v_i}{[Ax]_i} - v_i + [Ax]_i) + \frac{\alpha}{2}\|x\|_2^2 + \beta|x|_1 \tag{2}$$

where $v \in \mathcal{R}_+^n, A \in \mathcal{R}_+^{n\times r}, x \in \mathcal{R}_+^r$

From Equation 2, the first and second derivative of the variable x_k are computed by Formula 3:

$$\Rightarrow \begin{cases} \nabla f_k & = -\sum_{i=1}^{n} v_i \frac{A_{ik}}{[Ax]_i} + \sum_{i=1}^{n} A_{ik} + \alpha x_k + \beta \\ \nabla^2 f_{kk} & = \sum_{i=1}^{n} v_i (\frac{A_{ik}}{[Ax]_i})^2 + \alpha \end{cases} \tag{3}$$

Algorithm 4 is designed on randomized coordinate descent algorithm because updating one element in x requires a heavy computation.

In theoretical analysis of the method we obtained the following result:

Theorem 5 *The complexity of Algorithm 5 is $\mathcal{O}(nnnz(r)+\bar{t}r(r+n+nnz(n)))$, where $nnz(r)$ is the number of non-zero elements in x, $nnz(n)$ is the number of non-zero elements in v, and $\bar{t}$ is the average number of iterations. Then, the complexity of a while iteration in Algorithm 4 is $\mathcal{O}(\bar{t}(mnr + (m+n)r^2))$*

The method is experimentally evaluated on the four datasets, see Table 4, in terms of convergence, sparsity for factor matrices, the use of internal memory. The proposed algorithm (Sparse Randomized Coordinate Descent(SRCD)) is compared with two state-of-the-art methods Cycle Coordinate Descent (CCD) and Multiplicative Update(MU).

Table 5 shows that our method uses less internal memory than the others. For the dense dataset Digits, the proposed algorithm SRCD uses more internal memory than the algorithm CCD because a considerable amount of memory is used for the indexing of matrices:

Figure 6 and 7 show the running time of Algorithm SRCD for 100 iterations with different number of latent component using 1 thread. Clearly, the running time

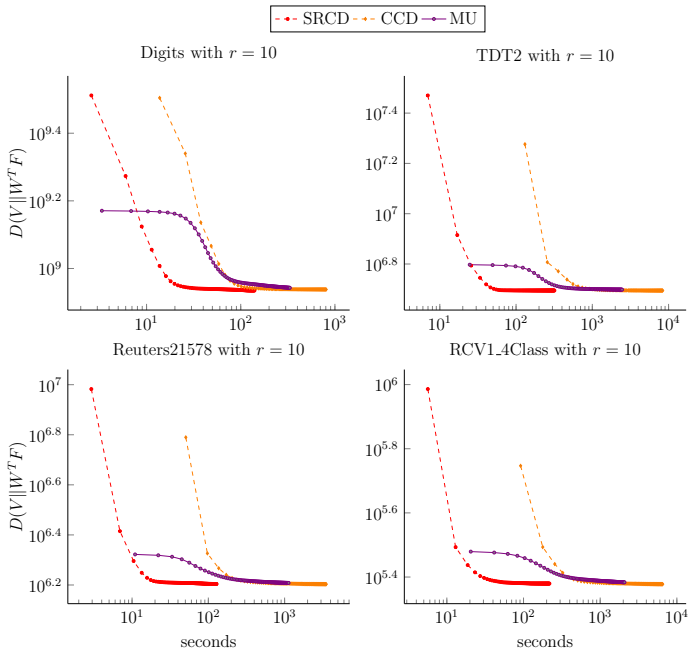

Figure 6: Objective value $D(V\|W^TF)$ versus running time with $r = 10$

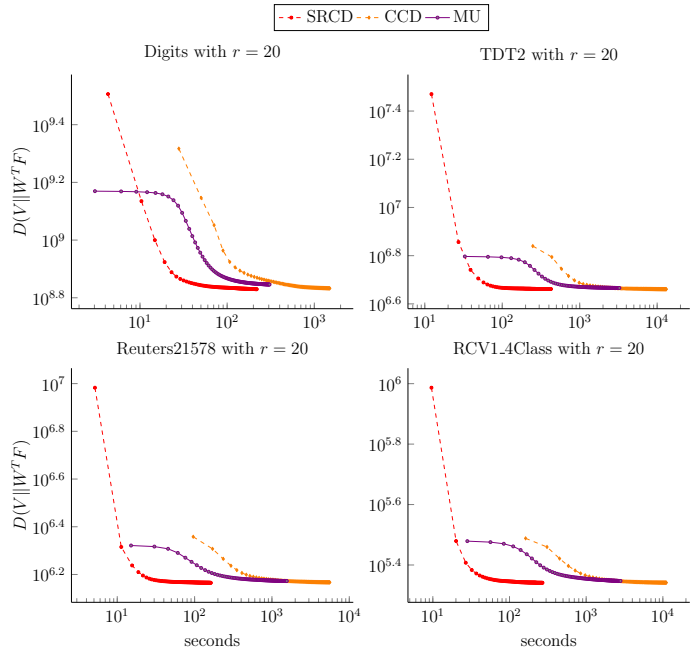

Figure 7: Objective value $D(V\|W^TF)$ versus running time with $r = 20$

Algorithm 5: Randomized coordinate descent algorithm for sparse datasets

Input: $v \in \mathcal{R}^n$, $A \in \mathcal{R}^{n \times k}$, $sumA$, $x \in \mathcal{R}^k$, $\alpha \geq 0$, $\beta \geq 0$, and variable order ids

Output: x is updated by

$$x \approx \underset{x \succeq 0}{\operatorname{argmin}} \sum_{i=1}^{n} -v_i \log([Ax]_i + \epsilon) + [Ax]_i + \frac{\alpha}{2}\|x\|_2^2 + \beta\|x\|_1.$$

```
begin
    Compute Ax ∈ R^n;
    for k in order ids do
        Compute ∇f_k and ∇²f_kk based on α, β, v, x, Ax, sumA and sparsity of
         v based on Formula 3;
        while (∇f_k < −ε) or (|∇f_k| > ε and x_k > ε) do
            △x = max(0, x_k − ∇f_k / ∇²f_kk) − x_k;
            Update Ax via Ax = Ax + △x A_k /*Re-computing Ax is removed*/;
            xs = x_k;
            x_k = x_k + △x;
            if (△x < ε_x xs) then
                break;
            Update ∇f_k and ∇²f_kk based on α, β, v, x, sumA and sparsity of v
             based on Formula 3;
    return x;
```

Table 4: Summary of datasets

Dataset (V)	n	m	nnz(V)	Sparsity (%)
Digits	784	60,000	8,994,156	80.8798
Reuters21578	8,293	18,933	389,455	99.7520
TDT2	9,394	36,771	1,224,135	99.6456
RCV1_4Class	9,625	29,992	730,879	99.7468

linearly increases, which fits the theoretical analyses about fast convergence and linear complexity for large sparse datasets.

Concerning the parallel algorithm, the running time of Algorithm SRCD for 100 iterations significantly decreases when the number of used threads increases in Figure 8 and 9. Furthermore, Table 9 indicates that highly sparse models and sparse representation can be attained for large sparse datasets, which is a significant milestone in researching this problem. In addition, the running time is acceptable for

Table 5: Used internal memory (GB) for $r = 10$

Datasets	MU	CCD	SRCD
Digits	1.89	**0.85**	1.76
Reuters21578	5.88	2.46	**0.17**
TDT2	11.51	5.29	**0.30**
RCV1_4Class	9.73	4.43	**0.23**

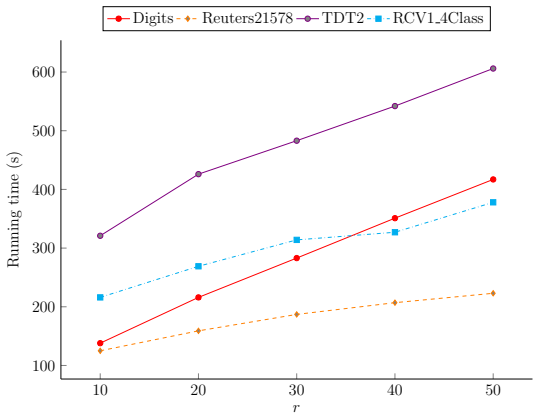

Figure 8: Running time of 100 iterations with different number of latent component using 1 thread

large applications. Hence, these results indicate that the proposed algorithm SRCD is feasible for large-scale applications.

The proposed algorithm attains the fastest convergence by means of removing the re-computation Ax over iteration, exploiting sparse properties of data, model and representation matrices, and utilizing the fast accessing speed of CPU cache. In addition, our method can stably run systems within limited internal memory by reducing internal memory requirements. In future research, we will generalize this algorithm for nonnegative tensor factorization.

Table 6: Sparsity (%) of (W, F) for the algorithm SRCD's results

	Digits	Reuters21578	TDT2	RCV1_4Class
$r = 10$	(74.3, 49.2)	(75.6, 71.6)	(68.5, 71.3)	(81.2, 74.0)
$r = 20$	(87.8, 49.7)	(84.2, 80.4)	(78.6, 81.1)	(88.4, 83.0)

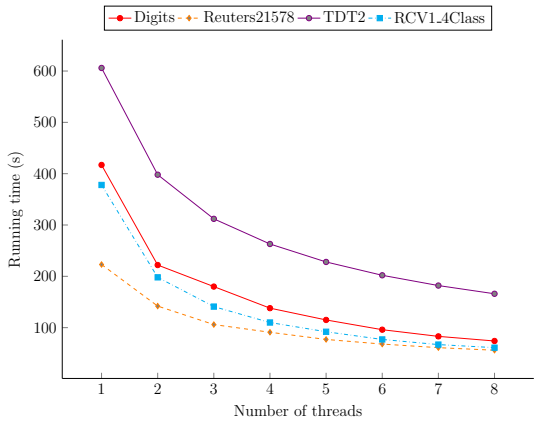

Figure 9: Running time of 100 iterations with $r = 50$ using different number of threads

4. New model of simplicial NMF with Frobenius norm and fast parallel algorithm

New NMF model, simplicial NMF, is proposed by adding simplicial constraints over coefficients $\sum_{k=1}^{r} F_{kj} = 1 \ \ \forall j$ that expresses the probabilistic combination role of latent components over data instances. The proposed fast parallel algorithm is derived from Frank-Wolfe algorithm for inferring instances has major advantages:

- **Sub-linear convergence**: The proposed algorithm guarantees sub-linear convergence $O(1/k)$ in the simplex of nonnegative variables.

- **Low iteration complexity**: The iteration complexity of proposed algorithm is $O(r)$ instead of $O(r^2)$ which is the iteration complexity of advanced guaranteed algorithm [5].

  Theorem 6 *The complexity of inferring a data instance is $\mathcal{O}(kr)$, and the complexity of inference step is $O(mnr + nr^2 + kmr)$, where k is the iteration number, n is the instance dimension, and m is the instance number.*

- **Sparsity control**: Sparsity of coefficients is easily controlled in simplicial NMF with Frobenius norm.

  Theorem 7 *Let f be a twice differentiable convex function over simplex $\triangle$ and denote $C_f = sup_{y,z \in \triangle; \tilde{y} \in [y,z]} (y - z).\nabla^2 f(\tilde{y}).(y - z)^T$. After l iterations, the Frank-Wolfe algorithm will find an approximate solution x_l with at most $(l + 1)$ non-zeros coefficients which satisfy*

$$max_{x \in \triangle} f(x_l) - f(x) \leq \frac{C_f}{l+1}$$

Algorithm 6: Iterative multiplicative update for Frobenius norm

Input: Data matrix $V = \{V_j\}_{j=1}^m \in R_+^{n \times m}$ and r.
Output: Coefficients $F \in R_+^{r \times m}$ and latent components $W \in R_+^{r \times n}$

begin
  Select randomly r components from m data instances;
  repeat
    $q = -WV$;
    $Q = WW^T$;
    Inference step: Fix W to find $F \approx \underset{F \in \mathcal{R}^{r \times m}}{\operatorname{argmin}} J(V \| W^T F)$;
    /*Call Algorithm 7 in parallel*/;
    for *j = 1 to m* **do**
      $F_j \approx \underset{x \ inR_+^r, x^T 1^r = 1}{\operatorname{argmin}} \|V_j - W^T x\|_2^2 = \underset{x \ inR_+^r, x^T 1^r = 1}{\operatorname{argmin}} \frac{1}{2} x^T Q x + q_j^T x$
    $q = -FV^T$;
    $Q = FF^T$;
    Learning step: Fix F to find $W \approx \underset{W \in \mathcal{R}^{r \times n}}{\operatorname{argmin}} J(V^T \| F^T W)$;
    /*Call solving NQP in parallel*/;
    for *i = 1 to n* **do**
      $W_i \approx \underset{x \ inR_+^r}{\operatorname{argmin}} \|V_i^T - F^T x\|_2^2 = \underset{x \ inR_+^r}{\operatorname{argmin}} \frac{1}{2} x^T Q x + q_i^T x$
    if *convergence condition is satisfied* **then**
      break;
  until *False*;

Our proposed method employs a multiple iterative update algorithm, see Algorithm 6, like EM algorithm containing two main step: (1) Inference step optimizes $D(V \| W^T F)$ by F to infer new coefficients of data instances, which calls Algorithm 7; (2) Learning step optimizes $D(V^T \| F^T W)$ by W to learn the parameters W of NMF model, which calls a nonnegative quadratic programming solver.

Algorithm 7 infers the coefficients of each instance, which is derived from Frank-Wolfe algorithm to achieve guaranteed convergence and to control effectively the sparsity of representation. Specially, the iteration complexity can be reduced into $O(r)$ instead of $O(r^2)$.

We investigate the effectiveness of our approach and the proposed algorithm for simplicial NMF (sNMF) with Frobenius norm by four criteria: interpretability, spar-

Algorithm 7: Inference for data instance x

Input: $Q \in \mathcal{R}^{r \times r}, q \in \mathcal{R}^r$.

Output: New coefficient $x \approx \underset{x \in \mathcal{R}^r}{\text{argmin}}\ f(x) = \underset{x \in \mathcal{R}^r}{\text{argmin}}\ \frac{1}{2} x^T Q x + q^T x$.

begin
- Choose $k = \arg\min_k \frac{1}{2} e_k^T Q e_k + q^T e_k$, where e_k is the k^{th} basis vector;
- Set $x = \mathbf{0}^k$; $x_k = 1$; $Qx = Qe_k$; $qx = q^T x$ and $\nabla f = Qx + q^T$;
- **repeat**
  - Select $k = \underset{k \in \{1..r\}}{\text{argmin}}\ \nabla f_k$;
  - Select $\alpha = \underset{\alpha}{\text{argmin}}\ f(\alpha e_k + (1-\alpha)x)$;
  - $\alpha = min(1, max(-1, max(\alpha, -\frac{f_k}{1-f_k})))$;
  - **if** $\alpha == 0$ **then**
    - break;
  - $Qx = (1-\alpha)Qx + \alpha Q e_k$;
  - $\nabla f = Qx + q$;
  - $qx = (1-\alpha)qx + \alpha q e_k$;
  - $x = (1-\alpha)x$;
  - $x_k = x_k + \alpha$;
- **until** *converged conditions are staisfied*;

sity of solutions, performance in classification tasks and loss information measure on 4000 random-selected samples from Digit datasets. Our algorithm (sNMF) for Frobenius form is compared to NMF [6], spNMF [7], oNMF[8], and cNMF [9]. The results indicate that the proposed algorithms can receive both highly sparse representation, see Figure 12 and 13; and high accuracy in classification, see Figure 14; while other algorithms can only achieve one of them.

Figure 12: Latent components with $K = 25$ for digit database (the left figure for NMF, and the right figure for sNMF)

5. New model of simplicial nonnegative matrix tri-factorization with Frobenius norm and fast parallel algorithm

Based on the previous research, existing NMF models remain the limitations in the terms of interpretability, guaranteed convergence, computational complexity, and sparse representation. Continuously, we propose to add simplicial constraints to the classical NMF model and to reformulate it into a new model called simplicial nonnegative matrix tri-factorization to have more concise interpretability via these values of factor matrices. Then, we propose an effective algorithm based on a combination of three-block alternating direction and Frank-Wolfe's scheme to attain linear convergence, low iteration complexity, and easily controlled sparsity. The experiments indicate that the proposed model and algorithm outperform the NMF model and its state-of-the-art algorithms.

Concerning the interpretability, NMF is a non-convex problem having numerous solutions as stationary points, which has rotational ambiguities [10]. Particularly, if $V \approx W^T F$ is a solution, $V \approx W^T[D_F F'] = [W^T D_F]F'$ are also equivalent solutions;

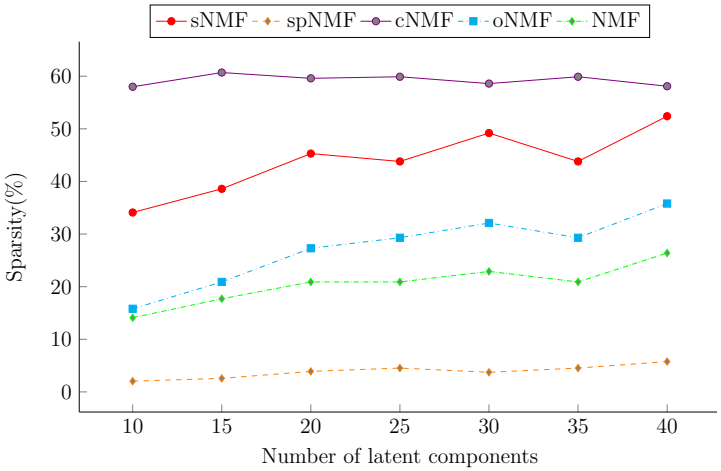

Figure 13: Sparsity of new coefficients for Frobenius norm with $r = 30$

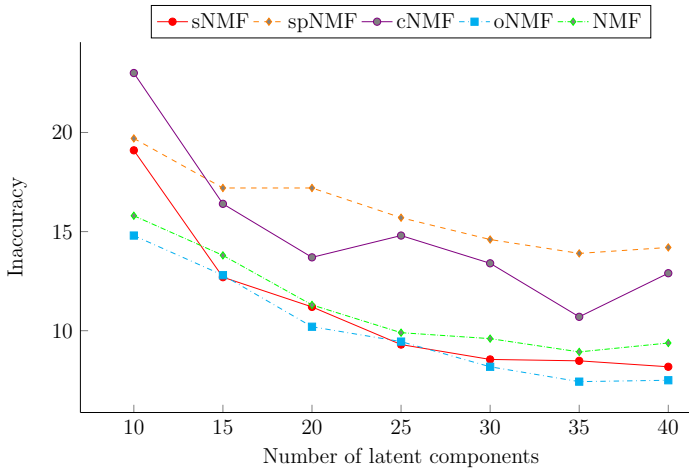

Figure 14: Inaccuracy for Digit Classification

where D_F is a positive diagonal matrix satisfying $F = D_F F'$. Hence, it does not consistently explain the roles of latent components over instances and the contributions of attributes over latent components.

To resolve the limitation, we propose a new formulation as SNMTF, in which the data matrix is factorized into a product of three matrices $V \approx W^T DF$, where D is a positive diagonal matrix, $\sum_{k=1}^{r} W_{ki} = 1 \forall i$, and $\sum_{k=1}^{r} F_{kj} = 1 \forall j$.

However, the scaling of factors via the diagonal matrix D can lead to the inconsistency of interpreting the factor matrices W and F, and the existence of bad stationary points such as $\lambda_k = 0 \Rightarrow W_{ki} = 0, F_{kj} = 0$. Hence, we restrict the formulation by adding the condition $\lambda_1 = ... = \lambda_r = \lambda$. This model can be considered as an extension of the probabilistic latent semantic indexing (PLSI) [11] for scaling data with an additional assumption that the weights of latent factors are the same. Remarkably, their roles of latent components over instances and of attributes over latent components are represented via the values of the factor matrices W and F. As a result, it is easier to recognize these roles than in the case that the weights of latent factors can be different. Therefore, the objective function of SNMTF with L_2 regularizations is written as follows:

$$\begin{cases} \min\limits_{W,D,F} & \left\{\Phi(W,D,F) := \frac{1}{2}\|V - W^T DF\|_F^2 + \frac{\alpha_1}{2}\|W\|_F^2 + \frac{\alpha_2}{2}\|F\|_F^2\right\} \\ \text{s.t.} & \sum\limits_{k=1}^{r} W_{ki} = 1 \ (i = 1, \cdots, n), \quad \sum\limits_{k=1}^{r} F_{kj} = 1 \ \ (j = 1, \cdots, m), \\ & W \in \mathbb{R}_+^{r \times n}, \quad F \in \mathbb{R}_+^{r \times m}, \quad D = \text{diag}(\lambda, \cdots, \lambda). \end{cases} \tag{4}$$

There are three significant remarks in this formulation. Firstly, adding simplicial constraints leads to more concise interpretability of the factor matrices and convenience for post processes such as neural network and support vector machine because the sum of attributes is normalized to 1. Secondly, L_1 regularization is ignored because $\|W\|_1$ and $\|F\|_1$ equal to a constant. Finally, the diagonal of $D = \text{diag}(\lambda, ..., \lambda)$ has the same value because of two main reasons: First, it is assumed that V_{ij} is generated by W_i, F_j and a scale as $V_{ij} \approx \lambda W_i^T F_j$; Second, it still retains the generalization of SNMTF which every solution of NMF can be equivalently represented by SNMTF.

We note that problem (4) is nonconvex due to the product $W^T DF$. We first propose to use three-block alternating direction method to decouple there three blocks. Then, we decompose the computation onto column of the matrix variables, which can be conducted in parallel. Finally, we apply Frank-Wolfe's algorithm [12] to solve the underlying subproblems.

We decouple the tri-product $W^T DF$ by the following alternating direction scheme, which is also called iterative multiplicative update:

$$\begin{cases} F^{t+1} & := \arg\min\limits_{F\in\mathbb{R}^{r\times m}} \left\{\Phi(W^t, D^t, F) : F \in \Delta_r^n\right\}, \\ W^{t+1} & := \arg\min\limits_{W\in\mathbb{R}^{r\times n}} \left\{\Phi(W, D^t, F^{t+1}) : W \in \Delta_r^m\right\}, \\ D^{t+1} & := \operatorname{argmin}\limits_{\lambda\in\mathbb{R}} \left\{\Phi(W^{t+1}, D, F^{t+1}) : D = \operatorname{diag}(\lambda, \cdots, \lambda)\right\}. \end{cases} \tag{5}$$

Clearly, both F-problem and W-problem in (5) are convex but are still constrained by simplex, while the D-problem is unconstrained. We now can solve the D-problem in the third line of (5). Due to the constraint $D = \operatorname{diag}(\lambda, \cdots, \lambda)$, this problem turns out to be an univariate convex program, which can be solved in a closed form as follows:

$$\lambda_{t+1} = \operatorname*{argmin}_{\lambda\in\mathbb{R}} \|V - {W^{t+1}}^T D F^{t+1}\|_2^2 = \frac{\langle V, {W^{t+1}}^T F^{t+1}\rangle}{\langle W^{t+1}{W^{t+1}}^T, F^{t+1}{F^{t+1}}^T\rangle}. \tag{6}$$

Then, we form the matrix D^{t+1} as $D^{t+1} = \operatorname{diag}(\lambda_{t+1}, \cdots, \lambda_{t+1})$.

If we look at the F-problem, then fortunately, it can be separated into n independent subproblems ($j = 1, \cdots, m$) of the form:

$$F_j^{t+1} = \arg\min_{F_j} \left\{\frac{1}{2}\|V_j - (W^t)^T D^t F_j\|_2^2 + \frac{\alpha_2}{2}\|F_j\|_2^2 : F_j \in \Delta_r\right\}. \tag{7}$$

The same trick is applied to the W-problem in the second line of (5). Now, we assume that we apply the well-known Frank-Wolfe algorithm to solve (7), then we can describe the full algorithm for solving (4) into Algorithm 8.

The stopping criterion of Algorithm 8 remains unspecified. Theoretically, we can terminate Algorithm 8 using the optimality condition of (4). However, computing this condition requires a high computational effort. We instead terminate Algorithm 8 if it does not significantly improve the objective value of (4) or the differences $W_{t+1} - W_t$ and $F_{t+1} - F_t$ and the maximum number of iterations.

Principally, we can apply any convex optimization method such as interior-point, active-set, projected gradient and fast gradient method to solve QP problems of the form (7). However, this QP problem (7) has special structure and is often sparse. In order to exploit its sparsity, we propose to use a Frank-Wolfe algorithm studied in [13] to solve this QP problem. Clearly, we can write (7) as follows:

$$x \approx \operatorname*{argmin}_{x\in\Delta_r} \frac{1}{2}\|v - A^T x\|_2^2 + \frac{\alpha}{2}\|x\|_2^2 = \operatorname*{argmin}_{x\in\Delta_r} \frac{1}{2}x^T Q x + q^T x \tag{8}$$

Algorithm 8: Iterative multiplicative update for Frobenius norm

Input: Data matrix $V = \{V_j\}_{j=1}^m \in \mathbb{R}_+^{n\times m}$, and r, $\alpha_1, \alpha_2 \geq 0, \beta \geq 0$.
Output: Coefficients $F \in R_+^{r\times m}$ and latent components $W \in R_+^{r\times n}$

begin
  Pick an arbitrary initial point $W \in \mathbb{R}_+^{r\times n}$ (e.g., random);
  repeat
    $q = -\lambda WV$; $Q = \lambda^2 WW^T + \alpha_1 \mathbb{I}$;
    /***Inference step:** Fix W and D to find new F;
    for *j = 1 to m* **do**
      $F_j \approx \underset{x\in\Delta_r}{\operatorname{argmin}} \left\{\frac{1}{2}x^T Qx + q_j^T x\right\}$/*Call Algorithm 9 in parallel */;
    $q = -\lambda FV^T$; $Q = \lambda^2 FF^T + \alpha_2 \mathbb{I}$;
    /***Learning step:** Fix F and D to find new W;
    for *i = 1 to n* **do**
      $W_i \approx \underset{x\in\Delta_r}{\operatorname{argmin}} \left\{\frac{1}{2}x^T Qx + q_i^T x\right\}$/*Call Algorithm 9 in parallel */;
    $\lambda = \underset{\lambda\in\mathbb{R}}{\operatorname{argmin}} \|V - W^T DF\|_2^2 = \frac{\langle V, W^T F\rangle}{\langle WW^T, FF^T\rangle}$;
  until *convergence condition is satisfied*;

Algorithm 9: Fast Algorithm for NQP with Simplicial Constraint

Input: $Q \in \mathbb{R}^{r\times r}, q \in \mathbb{R}^r$.
Output: New coefficient $x \approx \underset{x\in\Delta_r}{\operatorname{argmin}} f(x) = \frac{1}{2}x^T Qx + q^T x$.

begin
  Choose $k = \underset{k}{\operatorname{argmin}} \frac{1}{2}e_k^T Qe_k + q^T e_k$, where e_k is the k^{th} basis vector;
  Set $x = \mathbf{0}^k$; $x_k = 1$; $Qx = Qe_k$; $qx = q^T x$ and $\nabla f = Qx + q^T$;
  repeat
    Select $k = \underset{k\in\{1..r\}}{\operatorname{argmin}} \{\langle e_k - x, \nabla f\rangle\}$ *or* $\{\langle x - e_k, \nabla f\rangle | x_k > 0\}$;
    Select $\alpha = \underset{\alpha}{\operatorname{argmin}} f(\alpha e_k + (1-\alpha)x)$;
    $\alpha = min(1, max(\alpha, -\frac{x_k}{1-x_k}))$;
    $Qx = (1-\alpha)Qx + \alpha Qe_k$; $\nabla f = Qx + q$;
    $qx = (1-\alpha)qx + \alpha qe_k$;
    $x = (1-\alpha)x$; $x_k = x_k + \alpha$;
  until *converged conditions are staisfied*;

where $v = V_j, A = DW$, $Q = AA^T + \alpha$, and $q = -Av$. By applying the Frank-Wolfe algorithm form [13] to solve this problem, we obtain Algorithm 9 below.

In Algorithm 9, the first derivative of $f(x) = \frac{1}{2}x^TQx + q^Tx$ is computed by $\nabla f = Qx + q$. In addition, the steepest direction in the simplex is selected by this formula: $k = \underset{k\in\{1..r\}}{\operatorname{argmin}} \{\langle e_k - x, \nabla f\rangle\}$ *or* $\{\langle x - e_k, \nabla f\rangle | x_k > 0\}$.

For seeking the best variable α to minimize $f(\alpha x + (1-\alpha)e_k)$ where e_k is the k^{th} unit vector. Let consider $f(\alpha x + (1-\alpha)e_k)$, we have:

$$\begin{aligned} \frac{\partial f}{\partial \alpha}(\alpha = 0) &= (x - \mathbf{e}_k)^T(Qx + q) = x^T(Qx + q) - [Qx]_k - q_k \\ \frac{\partial^2 f}{\partial \alpha^2}(\alpha = 0) &= (x - \mathbf{e}_k)^TQ(x - \mathbf{e}_k) = x^TQx - 2[Qx]_k + Q_{kk}. \end{aligned} \tag{9}$$

Since f is a quadratic function of α, its optimal solution is $\alpha = \underset{\alpha\in[-\frac{x_k}{1-x_k},1]}{\operatorname{argmin}} f((1-\alpha)x + \alpha\mathbf{e}_k) = [-\frac{\nabla f_{\alpha=0}}{\nabla^2 f_{\alpha=0}}]_{[-\frac{x_k}{1-x_k},1]}$. The projection of solution over the interval $[-\frac{x_k}{1-x_k}, 1]$ is to guarantee $x_k \geq 0\ \forall k$. The updates $x = (1-\alpha)x$ and $x_k = x_k + \alpha$ are to retain the simplicial constraint $x \in \Delta_r$.

In Algorithm 9, duplicated computation is removed to reduce the iteration complexity into $\mathcal{O}(r)$ by maintaining Qx and q^Tx. This result is highly competitive with the state-of-the-art algorithm having a sub-linear convergence rate $\mathcal{O}(1/k^2)$ and complexity of $\mathcal{O}(r^2)$ [5].

This section discusses three important aspects of the proposed algorithm as convergence, complexity, and generalization. Concerning the convergence, setting $\alpha_1, \alpha_2 > 0$, based on Theorem 3 in Lacoste-Julien, S., & Jaggi, M. (2013) [13], since $f(x) = \frac{1}{2}x^TQx + q^Tx$ is smoothness and strongly convex, we have:

Theorem 8 *Algorithm 9 linearly converges as* $f(x_{k+1}) - f(x^*) \leq (1-\rho_f^{FW})^k\ (f(x_0) - f(x^*))$, *where* $\rho_f^{FW} = \{\frac{1}{2}, \frac{\mu_f^{FW}}{C_f}\}$, C_f *is the curvature constant of the convex and differentiable function* f, *and* μ_f^{FW} *is an affine invariant of strong convex parameter.*

Since Algorithm 9 always linearly converges and the objective function restrictedly decreases, Algorithm 6 always converges stationary points. Regarding the complexity of the proposed algorithm, we have:

Theorem 9 *The complexity of Algorithm 9 is* $\mathcal{O}(r^2 + \bar{t}r)$, *and the complexity of each iteration in Algorithm 6 is* $\mathcal{O}(mnr + (m+n)r^2 + \bar{t}(m+n)r)$.

Proof The complexity of the initial computation $Qx + q$ is $O(r^2)$, and each iteration in Algorithm 9 is $O(r)$. Hence, the complexity of Algorithm 9 is $\mathcal{O}(r^2 + \bar{t}r)$.

The complexity of main operators in Algorithm 6 as WV, FV^T, FF^T, and WW^T is $O(mnr + (m+n)r^2)$. Overall, the complexity of each iteration in Algorithm 6 is $\mathcal{O}(mnr + (m+n)r^2 + \bar{t}(m+n)r)$. ■

This is highly competitive with the guaranteed algorithm [5] having the complexity of $\mathcal{O}(mnr + (m+n)r^2 + \bar{t}(m+n)r^2)$. Furthermore, adding the simplicial constants into NMF does not reduce the generalization and flexibility of NMF:

Theorem 10 *Each solution of NMF can be equivalently transfered into SNMTF.*

Proof Assume that $V \approx W^T F$, which leads to the existence of λ large enough to satisfy $W^T F = W'^T D' F'$, where $D' = \text{diag}(\lambda, .., \lambda)$, $\sum_{k=1}^{r} W_{ki} < 1 \ \forall i$, and $\sum_{k=1}^{r} F_{kj} < 1 \ \forall j$. Therefore, $\exists W''$, D'', and F'': $W''^T D'' F'' = W'^T D' F' = W^T F$, where $W''_{ij} = W'_{ij}$, $F''_{ij} = F'_{ij}$, $W''_{r+1,i} = 0, W''_{r+2,i} = 1 - \sum_{k=1}^{r+1} W''_{ki} \ \forall i$, $F''_{r+1,j} = 1 - \sum_{k=1}^{r} F''_{kj}, F''_{r+2,j} = 0 \ \forall j$,$D'' = \text{diag}(\lambda, .., \lambda)$. $W''^T D'' F''$ is SNMTF of V. ■

The generalization is crucial to indicate the robustness and high flexibility of the proposed model in comparison with NMF models, although many constraints have been added to enhance the quality and interpretability of the NMF model.

The proposed algorithm **SNMTF** is compared with the following methods:

- **NeNMF** [5]: It is a guaranteed method, each alternative step of which sublinearly converges at $\mathcal{O}(1/k^2)$ that is highly competitive with the proposed algorithm.
- **LeeNMF** [14]: It is the original gradient algorithm for NMF.
- **PCA**: It is considered as a based-line method in dimensionality reduction, which is compared in classification and sparsity.

Datasets: We compared the selected methods in three typical datasets with different size, namely Faces[1], Digits[2] and Tiny Images [3].

Environment settings: We develop the proposed algorithm SNMTF in Matlab with embedded code C++ to compare them with other algorithms. We set system

1. `http://cbcl.mit.edu/cbcl/software-datasets/FaceData.html`
2. `http://yann.lecun.com/exdb/mnist/`
3. `http://horatio.cs.nyu.edu/mit/tiny/data/index.html`

Table 7: Dataset Information

Datasets	n	m	testing size	r	#class
Faces	361	$6,977$	24,045	30	2
Digits	784	6.10^4	10^4	60	10
Cifar-10	3,072	5.10^4	10^4	90	10

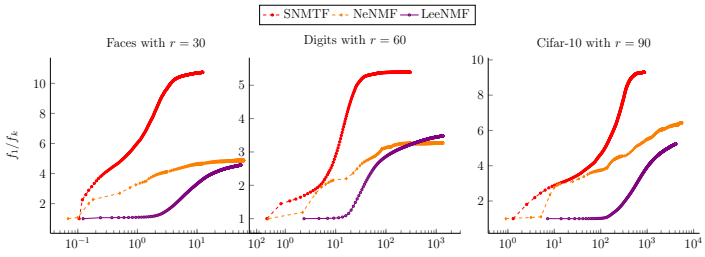

Figure 15: Convergence of loss information f_1/f_k versus time

parameters to use only six threads in the machine Mac Pro 6-Core Intel Xeon E5 3 GHz 32GB. The initial matrices W^0 and F^0 are set to the same values, the maximum number of iterations is 500. The source code is published on our homepage [4].

We investigate the convergence of the compared algorithms by $\frac{f_1}{f_k}$ because they have the different formulations and objective functions. Fig. 15 clearly shows that the proposed algorithm converges much faster than the other algorithms. The most steepest line of the proposed algorithm represents its fast convergence. This result is reasonable because the proposed algorithm has a faster convergence rate and lower complexity than the state-of-the-art algorithm NeNMF.

Concerning the classification performance, the training datasets with labels are used to learn gradient boosting classifiers [15, 16], one of the robust ensemble methods, to classify the testing datasets. The proposed algorithm outperforms the other algorithms and PCA over all the datasets. For the small and easy dataset Face, the result of the proposed algorithm is close to the results of NeNMF. However, for larger and more complex datasets Digit and Tiny Images, the proposed algorithm has much better accuracy than the other algorithms. Noticeably, the result of Tiny Images is much worse than the result of the other datasets because it is highly complicated and contains backgrounds. This classification result obviously represents the effectiveness of the proposed model and algorithm.

4. `http://khuongnd.appspot.com/`

Table 8: Classification inaccuracy (%)

Dataset	PCA	LeeNMF	NeNMF	SNMTF
Faces	6.7	3.3	2.7	**2.6**
Digits	30.15	12.16	3.9	**3.6**
Tiny Images	59.3	71.4	51.4	**50.1**

Table 9: Sparsity of factor matrices (%) of F, W, and (both F and W)

Dataset	PCA	LeeNMF	NeNMF	SNMTF
Faces	(0, 0.26, 0.24)	(0.58, 0, 0.55)	(7.64, **60.33** , 10.23)	(**9.78** , 59.98, **12.24**)
Digits	(1.37, 0, 0.02)	(31.24, 50.47, 31.49)	(41.20, 92.49, 41.86)	(**50.49** , **93.47** , **51.04**)
Tiny Images	(0, 0, 0)	(0.02, 0, 0.02)	(9.97, **86.58** , 14.40)	(**11.71** , 85.29, **15.97**)

We investigate the sparsity of factor matrices F, W, and the sparsity average in both F and W. For the dataset Digit, the proposed algorithm outperforms in all these measures. For the other datasets Faces and Tiny Images, it has better representation F and more balance between sparsity of F and W. Frankly speaking, in these datasets, achieving more sparse representation F is more meaningful than achieving more sparse model W because F is quite dense but W is highly sparse, which is a reason to explain why SNMTF has the best classification result.

This research proposes a new model of NMF as SNMTF with L_2 regularizations, which has more concise interpretability of the role of latent components over instances and attributes over latent components while keeping the generalization in comparison with NMF. We design a fast parallel algorithm with guaranteed convergence, low iteration complexity, and easily controlled sparsity to learn SNMTF, which is derived from Frank-Wolfe algorithm [13]. Furthermore, the proposed algorithm is convenient to parallelize and distribute, and control sparsity of both new representation F and model W. Based on the experiments, the new model and the proposed algorithm outperform the NMF model and its state-of-the-art algorithms in three significant aspects of convergence, classification, and sparsity. Therefore, we strongly believe that SNMTF is highly potential for many applications and extensible for nonnegative tensor factorization.

6. New model of simplicial NMF with Kullback-Leibler divergence and fast parallel algorithm

This study extends simplicial nonnegative matrix factorization for KL divergence by proposing a fast parallel algorithm having guaranteed sub-linear convergence in inference for large sparse datasets to archive sparse representation. The proposed algorithm has significant properties as follows:

- **Sub-linear convergence**: The proposed algorithm guarantees sub-linear convergence in the simplex of nonnegative variables. Remarkably, no existing algorithm for NMF with KL divergence has this convergence guarantee.

- **Low complexity**: The iteration complexity of proposed algorithm is $\mathcal{O}(k[r.nnz(n) + n\log\frac{1}{\epsilon}])$ which is feasible for large sparse datasets. Furthermore, the computation for learning step is extremely fast with the iteration complexity because of the data sparsity ($nnz(n) \ll n$).

  Theorem 11 *Consider Algorithm 11 to infer a data instance having n-dimension by r latent components with k iterations. Then, its complexity is* $\mathcal{O}(k[r.nnz(n)+ n\log\frac{1}{\epsilon}])$*, where* $nnz(n)$ *is the number of non-zero elements in* v*.*

- **Sparsity control**: Sparsity is easily controlled for simplicial NMF with KL divergence because the proposed algorithm is derived from Frank-Wolfe algorithm.

Our proposed method employs a multiple iterative update algorithm, see Algorithm 10, like EM algorithm containing two main step: (1) Inference step optimizes $D(V\|W^TF)$ by F, which calls Algorithm 11; (2) Learning step optimizes $D(V\|W^TF)$ by W, the solution of which can be directly approximated.

Algorithm 11 infers the coefficients of each instance, which is derived from Frank-Wolfe algorithm to achieve guaranteed convergence and to control effectively the sparsity of representation.

We investigate the effectiveness of our approach and the proposed algorithm for simplicial NMF with KL divergence by four criteria: sparsity of solutions, performance in classification tasks and loss information measure. The proposed algorithm for simplicial NMF with KL divergence (sNMF) is is compared to kl-NMF[6], local nonnegative matrix factorization (locNMF)[17], convolutional NMF(conNMF) [18], and

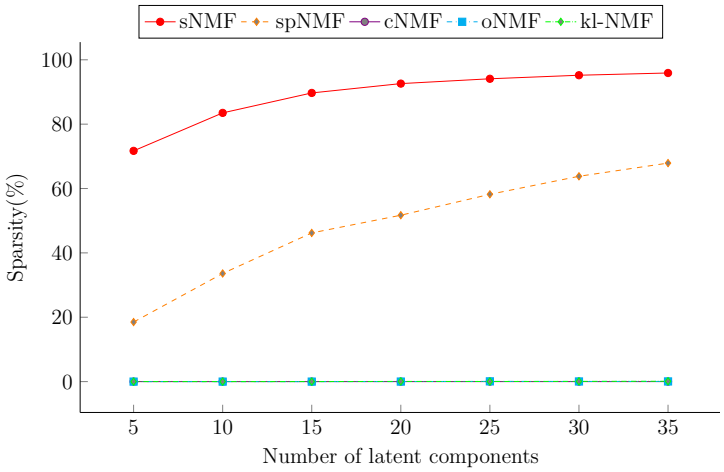

Figure 16: Sparsity of new coefficients for KL divergence with $r = 30$

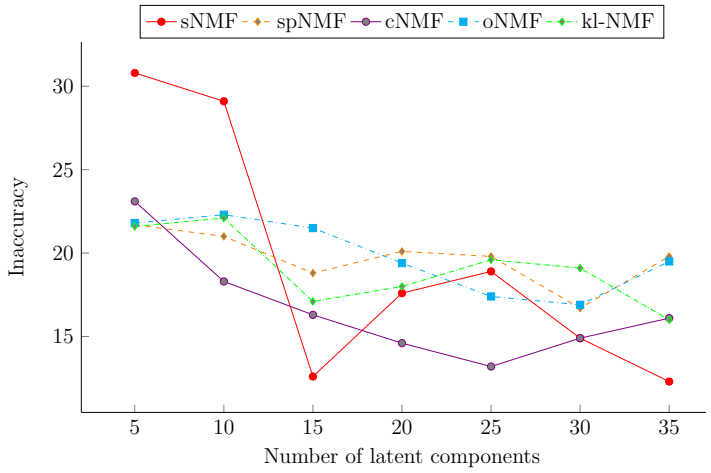

Figure 17: Inaccuracy for Spam Classification

Algorithm 10: Iterative multiplicative update for KL divergence

Input: Data matrix $V = \{V_j\}_{j=1}^{m} \in R_+^{n \times m}$ and r.
Output: Coefficients $F \in R_+^{r \times m}$ and latent components $W \in R_+^{r \times n}$

begin
  Select randomly r components from m data instances;
  repeat
    Inference step: Fix W to find $F \approx \underset{F \in \mathcal{R}^{r \times m}}{\text{argmin}}\, J(V \| W^T F)$;
    Compute $sumW = W\mathbf{1}^n$;
    /*Call Algorithm 11 in parallel*/;
    for $j = 1$ *to* m **do**
      $F_j \approx \underset{x \ in R_+^r, x^T 1^r = 1}{\text{argmin}} D_{KL}(V_j \| W^T x)$ with $sumW$;
    Learning step: Fix F to find $W \approx \underset{W \in \mathcal{R}^{r \times n}}{\text{argmin}}\, J(V^T \| F^T W)$;
    /*Learning new latent components by approximation algorithm*/;
    for $i = 1$ *to* n **do**
      $W_i \approx \underset{x \ in R_+^r}{\text{argmin}}\, D_{KL}(V_i^T \| F^T x) \Rightarrow W_{ki} = \frac{\sum_{j=1}^{m} V_{ij}}{\sum_{j=1}^{m} F_{kj}}$
    if *convergence condition is satisfied* **then**
      break;
  until *False*;

Nonsmooth Nonnegative Matrix Factorization (nsNMF)[19] on 4327 labeled spam emails with 32906 distinct terms.

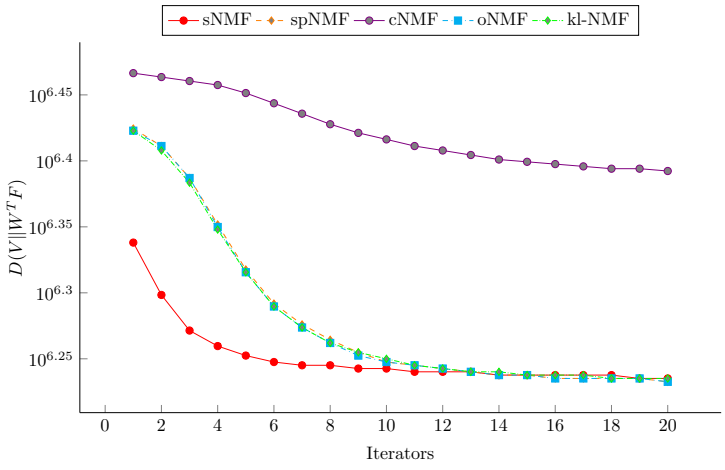

Figure 18: Information Loss for KL divergence with $r = 30$

Algorithm 11: Inference for data instance x

Input: Data instance $v \in \mathcal{R}_+^r$ and latent components $A \in \mathcal{R}_+^{r \times n}$; and $sumA = A\mathbf{1}^n$

, **Output:** New coefficient $x \approx \underset{x \in \mathcal{R}_+^r}{\operatorname{argmin}} \sum_{i=1}^{n} (v_i \log(\frac{v_i}{A_i^T x + \epsilon}) - v_i + A_i^T x$

begin

  Choose component W_i^T closest to x in KL divergence;

  Set $x = \mathbf{0}^r$; $x_k = 1$; and $Wx = W^T x$;

  repeat

    Computing ∇f;

    Select $k = \underset{i \in \{1..r\}}{\operatorname{argmin}} \nabla f_k$;

    Select $\alpha = \underset{\alpha \in [0,1]}{\operatorname{argmin}} f(\alpha e_k + (1-\alpha)x)$;

    $Wx = (1-\alpha)Wx + \alpha W_k^T e_k$;

    Set $x = (1-\alpha)x$ and $x_k = x_k + \alpha$;

  until *Convergence condition satisfied*;

The experimental results indicate that the proposed algorithm can achieve highest sparse representation (see Figure 16), highly competitive accuracy in classification (see Figure 17), and the fastest convergence versus iteration (see Figure 18).

Key references

[1] Christopher M Bishop. Model-based machine learning. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 371(1984):20120222, 2013.

[2] Yu-Xiong Wang and Yu-Jin Zhang. Nonnegative matrix factorization: A comprehensive review. *Knowledge and Data Engineering, IEEE Transactions on*, 25(6):1336–1353, 2013.

[3] Yu Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.

[4] Mark Schmidt and Michael Friedlander. Coordinate descent converges faster with the gauss-southwell rule than random selection. In *NIPS OPT-ML workshop*, 2014.

[5] Naiyang Guan, Dacheng Tao, Zhigang Luo, and Bo Yuan. Nenmf: an optimal gradient method for nonnegative matrix factorization. *Signal Processing, IEEE Transactions on*, 60(6):2882–2898, 2012.

[6] D Seung and L Lee. Algorithms for non-negative matrix factorization. *Advances in neural information processing systems*, 13:556–562, 2001.

[7] Mikkel N Schmidt, Jan Larsen, and Fu-Tien Hsiao. Wind noise reduction using non-negative sparse coding. In *Machine Learning for Signal Processing, 2007 IEEE Workshop on*, pages 431–436. IEEE, 2007.

[8] Seungjin Choi. Algorithms for orthogonal nonnegative matrix factorization. In *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, pages 1828–1832. IEEE, 2008.

[9] C.H.Q. Ding, T. Li, and M.I. Jordan. Convex and semi-nonnegative matrix factorizations. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(1):45–55, 2010.

[10] Andrzej Cichocki, Rafal Zdunek, Anh Huy Phan, and Shun-ichi Amari. *Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation*. Wiley, 2009.

[11] Thomas Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57. ACM, 1999.

[12] Martin Jaggi. Revisiting frank-wolfe: Projection-free sparse convex optimization. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 427–435, 2013.

[13] Simon Lacoste-Julien and Martin Jaggi. An affine invariant linear convergence analysis for frank-wolfe algorithms. *arXiv preprint arXiv:1312.7864*, 2013.

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

[15] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, 29(5):1189–1232, 2001.

[16] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin, 2001.

[17] Stan Z Li, Xin Wen Hou, Hong Jiang Zhang, and Qian Sheng Cheng. Learning spatially localized, parts-based representation. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–207. IEEE, 2001.

[18] Paris Smaragdis. Non-negative matrix factor deconvolution; extraction of multiple sound sources from monophonic inputs. In *Independent Component Analysis and Blind Signal Separation*, pages 494–499. Springer, 2004.

[19] Alberto Pascual-Montano, Jose Maria Carazo, Kieko Kochi, Dietrich Lehmann, and Roberto D Pascual-Marqui. Nonsmooth nonnegative matrix factorization (nsnmf). *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(3):403–415, 2006.

7. List of Publications and Significant Collaborations that resulted from our AOARD supported project

7.1 List of peer-reviewed journal publications

[1] Nguyen, D. K., & Ho, T. B. (2016). Fast Parallel Randomized Algorithm for Nonnegative Matrix Factorization with KL Divergence for Large Sparse Datasets. International Journal of Machine Learning and Computing, 6(2), 111.

[2] Nguyen, D. K., & Ho, T. B. (2016). Accelerated parallel and distributed algorithm using limited internal memory for nonnegative matrix factorization. Journal of Global Optimization, 1-22.

[3] Nguyen, D. K., & Ho, T. B. (2017). Accelerated anti-lopsided algorithm for nonnegative least squares. International Journal of Data Science and Analytics, 1-12.

7.2 List of peer-reviewed conference publications

[4] Nguyen, D. K., Than, K., & Ho, T. B. (2013, November). Simplicial nonnegative matrix factorization. In Computing and Communication Technologies, Research, Innovation, and Vision for the Future (RIVF), 2013 IEEE RIVF International Conference on (pp. 47-52). IEEE.

[5] Nguyen DK., Tran-Dinh Q., Ho TB. (2016) Simplicial Nonnegative Matrix Tri-factorization: Fast Guaranteed Parallel Algorithm. In: Hirose A., Ozawa S., Doya K., Ikeda K., Lee M., Liu D. (eds) Neural Information Processing. ICONIP 2016. Lecture Notes in Computer Science, vol 9948. Springer, Cham.

7.3 Papers published in non-peer-reviewed journals and conference proceedings

7.4 Conference presentations without papers

7.5 Manuscripts submitted but not yet published

7.6 Provide a list any interactions with industry or with Air Force Research Laboratory scientists or significant collaborations that resulted from this work

- Collaboration with Fujitsu Hokuriku on Data mining methods for detection of anormals in IT systems.
- Hokuriku CityHall Office on Analyzing the health care data.

8. Attachments

Publications in sections 7.1 and 7.2 listed above if possible.