Fast Parallel Randomized Algorithm for Nonnegative Matrix Factorization with KL Divergence for Large Sparse Datasets

Duy-Khuong Nguyen and Tu-Bao Ho

***Abstract*—Nonnegative Matrix Factorization (NMF) with Kullback-Leibler Divergence (NMF-KL) is one of the most significant NMF problems and equivalent to Probabilistic Latent Semantic Indexing (PLSI), which has been successfully applied in many applications. For sparse count data, a Poisson distribution and KL divergence provide sparse models and sparse representation, which describe the random variation better than a normal distribution and Frobenius norm. Specially, sparse models provide more concise understanding of the appearance of attributes over latent components, while sparse representation provides concise interpretability of the contribution of latent components over instances. However, minimizing NMF with KL divergence is much more difficult than minimizing NMF with Frobenius norm; and sparse models, sparse representation and fast algorithms for large sparse datasets are still challenges for NMF with KL divergence. In this paper, we propose a fast parallel randomized coordinate descent algorithm having fast convergence for large sparse datasets to archive sparse models and sparse representation. The proposed algorithm's experimental results overperform the current studies' ones in this problem.**

***Index Terms*—Nonnegative matrix factorization, kullback-leibler divergence, sparse models, and sparse representation.**

I. Introduction

The development of technology has been generating big datasets of count sparse data such as documents and social network data, which requires fast effective algorithms to manage this huge amount of information. One of these tools is nonnegative matrix factorization (NMF) with KL divergence, which is proved to be equivalent with Latent Semantic Indexing (PLSI) [1].

NMF is a powerful linear technique to reduce dimension and to extract latent topics, which can be readily interpreted to explain phenomenon in science [2]-[4]. NMF makes post-processing algorithms such as classification and information retrieval faster and more effective. In addition, latent factors extracted by NMF can be more concisely interpreted than other linear methods such as PCA and ICA [4]. In addition, NMF is flexible with numerous divergences to adapt a large number of real applications [5], [6].

For sparse count data, NMF with KL divergence and a Poisson distribution may provide sparse models and sparse representation describing better the random variation rather than NMF with Frobenius norm and a normal distribution [7]. For example, the appearance of words over latent topics and of topics over documents should be sparse. However, achieving sparse models and sparse representation is still a major challenge because minimizing NMF with KL divergence is much more difficult than NMF with Frobenius norm [8].

In the NMF-KL problem, a given nonnegative data matrix $V \in R_+^{n \times m}$ must be factorized into a product of two nonnegative matrices, namely a latent component matrix $W \in R_+^{r \times n}$ and a representation matrix $F \in R_+^{r \times m}$, where n is the dimension of a data instance, m is the number of data instances, and r is the number of latent components or latent factors. The quality of this factorization is controlled by the objective function with KL divergence as follows:

$$D(V \| W^T F) = \sum_{i=1}^{n} \sum_{j=1}^{m} (V_{ij} \log \frac{V_{ij}}{(W^T F)_{ij}} - V_{ij} + (W^T F)_{ij}) \quad (1)$$

In the general form of L_1 and L_2 regularization variants, the objective function is written as follows:

$$D(V \| W^T F) + \frac{\alpha_1}{2} \| W \|_2^2 + \frac{\alpha_2}{2} \| F \|_2^2 + \beta_1 \| W \|_1 + \beta_2 \| F \|_1 \quad (2)$$

NMF with KL divergence has been widely applied in many applications for dense datasets. For example, spatially localized, parts-based subspace representation of visual patterns is learned by local non-negative matrix factorization with a localization constraint (LNMF) [9]. In another study, multiple hidden sound objects from a single channel auditory scene in the magnitude spectrum domain can be extracted by NMF with KL divergence [10]. In addition, two speakers in a single channel recording can be separated by NMF with KL divergence and L_1 regularization on F [11].

However, the existing algorithms for NMF with KL divergence (NMF-KL) are extremely time-consuming for large count sparse datasets. Originally, Lee and Seung, 2001 [12] proposed the first multiple update iterative algorithm based on gradient methods for NMF-KL. Nevertheless, this technique is simple and ineffective because it requires a large number of iterations, and it ignores negative effects of

Manuscript received January 15, 2016; revised March 9, 2016. This work is partially sponsored by Asian Office of Aerospace R&D under agreement number BAA-AFOSR-2014-0001, by funded by Vietnam National University at Ho Chi Minh City under the grant number B2015-42-02m, and by 911 Scholarship from Vietnam Ministry of Education and Training.

D. K. Nguyen is with Japan Advanced Institute of Science and Technology, and University of Engineering and Technology, Vietnam National University, Hanoi, Vietnam (e-mail: khuongnd@gmail.com).

T. B. Ho was with Japan Advanced Institute of Science and Technology, and John von Neumann Institute, Vietnam National University, Ho Chi Minh City, Vietnam (e-mail: bao@jaist.ac.jp).

doi: 10.18178/ijmlc.2016.6.2.583

nonnegative constraints. In addition, gradient methods have slow convergence for complicated logarithmic functions like KL divergence. Subsequently, Cho-Jui & Inderjit, 2011 [13] proposed a cycle coordinate descent algorithm having low complexity of one variable update. However, this method contains several limitations: first, it computes and stores the dense product matrix $W^T F$ although V is sparse; second, the update of $W^T F$ for the change of each cell in W and F is considerably complicated, which leads to practical difficulties in parallel and distributed computation; finally, the sparsity of data is not considered, while large datasets are often highly sparse.

In comparison with NMF with Frobenius norm, NMF with KL divergence is much more complicated because updating one variable will influence derivatives of other variables; this computation is extremely expensive. Hence, it is difficult to employ fast algorithms having multiple variable updates, which limits the number of effective methods.

In this paper, we propose a new advanced version of coordinate descent methods with significant modifications for large sparse datasets. Regarding the contributions of this paper, we:

- Propose a fast sparse randomized coordinate descent algorithm using limited internal memory for nonnegative matrix factorization for huge sparse datasets, the full matrix of which can not stored in the internal memory. In this optimization algorithm, variables are randomly selected with uniform sampling to balance the order priority of variables. Moreover, the proposed algorithm effectively utilizes the sparsity of data, models and representation matrices to improve its performance. Hence, the proposed algorithm can be considered an advanced version of cycle coordinate descent for large sparse datasets proposed in [13].
- Design parallel algorithms for combinational variants of L_1 and L_2 regularizations.
- Indicate that the proposed algorithm using limited memory can fast attain sparse models, sparse representation, and fast convergence by evaluational experiments, which is a significant milestone in this research problem for large sparse datasets.

The rest of the paper is organized as follows. Section II presents the proposed algorithms. The theoretical analysis of convergence and complexity is discussed in Section III. Section IV shows the experimental results, and Section V summarizes the main contributions of this paper and discussion.

II. Proposed Algorithm

In this section, we propose a fast sparse randomized coordinate descent parallel algorithm for nonnegative sparse matrix factorization on Kullback-Leibler divergence. We employ a multiple iterative update algorithm like EM algorithm, see Algorithm 1, because the objective function $D(V \| W^T F)$ is a non-convex function although it is a convex function when fixing one of two matrices W^T 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^T; and the another one is to optimize the objective function by W when fixing F. Furthermore, in this algorithm, we need to minimize Function 3, the decomposed elements of which can be independently optimized in Algorithm 1:

$$D(V \| W^T F) = \sum_{j=1}^{m} D(V_j \| W^T F_j) = \sum_{i=1}^{n} D(V_i^T \| F^T W_i) \quad (3)$$

Specially, a number of optimization problems $D(V_j \| W^T F_j)$ or $D(V_i^T \| F^T W_i)$ in Algorithm 1 with the form $D(v \| Ax)$ can be independently and simultaneously solved by Algorithm 2. In this paper, we concern combinational variants of NMF KL divergence with L_1 and L_2 regularizations in the general formula, Function 4:

$$f(x) = D(x \| Ax) = \sum_{i=1}^{n} (v_i \log \frac{v_i}{A_i^T x} - v_i + A_i^T x) + \frac{\alpha}{2} \| x \|_2^2 + \beta \| x \|_1 \quad (4)$$

where $v \in R_+^n$, $A \in R_+^{n \times r}$, and $x \in R_+^r$.

Algorithm 1: 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 ∈ R_+^{r×n};
    Randomize F ∈ R_+^{r×m};
    while convergence condition is not satisfied do
        ids = a randomized ordered set of values {1, 2, ..., r};
        sumW = W1^n;
        /*Optimizing the objective function by F when fixing W*/;
        for j = 1 to m do
            /*Call Algorithm 2 in parallel*/;
            F_j^{k+1} = Algorithm 2 (V_j, W^T, sumW, F_j^k, α_2, β_2, ids)
        sumF = F1^m;
        /*Optimizing the objective function by W when fixing F*/;
        for i = 1 to n do
            /*Call Algorithm 2 in parallel*/;
            W_i^{k+1} = Algorithm 2 (V_i^T F^T, sumW, W_i^k, α_2, β_2, ids);
    return (W^{k+1})^T, F^{k+1};
```

Because the vector v is given, minimizing Function 4 is equivalent to minimizing Function 5:

$$f(x) = D(x \| Ax) = \sum_{i=1}^{n} (-v_i \log(A_i^T x) + A_i^T x) + \frac{\alpha}{2} \| x \|_2^2 + \beta \| x \|_1 \quad (5)$$

From Equation **5**, the first and second derivative of the variable x_k are computed by Formula 6:

$$\Rightarrow \begin{cases} \nabla f_k = \dfrac{\partial f}{\partial x_k} = -\sum_{i=1}^{n} v_i \dfrac{A_{ik}}{A_i^T x} + \sum_{i=1}^{n} A_{ik} + \alpha x_k + \beta \\ \nabla^2 f_{kk} = \dfrac{\partial^2 f}{\partial x_k^2} = \sum_{i=1}^{n} v_i (\dfrac{A_{ik}}{A_i^T x})^2 + \alpha \end{cases} \quad (6)$$

Based on Formula 6, we have several significant remarks:

- One update of x_k changes all elements of Ax, which are

under the denominators of fractions. Hence, it is difficult to employ fast algorithms having simultaneous updates of multiple variables because it will require heavy computation. Hence, we employ coordinate descent methods to reduce the complexity of each update, and to avoid negative effects of nonnegative constraints.

- One update of x_k has complexity of maintaining ∇f_k and $\nabla^2 f_{kk}$ as $O(k+nnz(v)+nnz(v)) \approx k+nnz(v)$ when $\sum_{i=1}^{n} A_{ik}$ is computed in advance. Exactly, this update employs $O(k+nnz(v))$ of multiple and addition operators; and $O(nnz(v))$ of divide operators; where $nnz(v)$ is the number of is the number of non-zero elements in the vector v. Hence, for sparse datasets, the number of operators can be negligible.
- The used internal memory of Algorithm 1 and Algorithm 2 is $O(nnz(V)+size(W)+size(F)) \approx nnz(V)+(n+m)r$ where $nnz(V)$ is the number of non-zero elements in the given matrix V, which is much smaller than $O(mn+(n+m)r)$ for the existing algorithms [12], [13].

Hence, Algorithm 2 employs a coordinate descent algorithm based on projected Newton methods with quadratic approximation in Algorithm 2 is to optimize Function 5 Specially, because Function 5 is convex, a coordinate descent algorithm based on projected Newton method [14] with quadratic approximation is employed to iteratively update with the nonnegative a lower bound as follows:

$$x_k = \max(0, x_k - \frac{\nabla f_k}{\nabla^2 f_{kk}})$$

Considering the limited internal memory and the sparsity of x, we maintain $W^T F$ via computing Ax instead of storing the dense matrix $W^T F$ for the following reasons:

- The internal memory requirement will significantly decrease, so the proposed algorithm can stably run on limited internal memory machines.
- The complexity of computing Ax is always smaller than the complexity of computing and maintaining ∇f_k and $\nabla^2 f_{kk}$, so it does not cause the computation more complicated.
- The updating $Ax = Ax + \Delta x A_k$ as the adding with a scale of two vectors utilizes the speed of CPU cache because of accessing consecutive memory cells.
- The recomputing helps remove the complexity of maintaining the dense product matrix $W^T F$ as in [12], [13], which is certainly considerable because this maintenance accesses memory cells far together and does not utilize CPU cache.

In summary, in comparison with the original coordinate algorithm [13] for NMK-KL, the proposed algorithm involve significant improvements as follows:

- Randomize the order of variables to optimize the objective function in Algorithm 2. Hence, the proposed algorithm can balance the order priority of variables.
- Remove duplicated computation of maintaining derivatives ∇f_k and $\nabla^2 f_{kk}$ by computing common elements $sumW = W1^n$ and $sumF = F1^n$ in advance, which led to that the complexity of computing ∇f_k and $\nabla^2 f_{kk}$ only depends on the sparsity of data.
- Effectively utilize the sparsity of W and F to reduce the running time of computing Ax.
- Recompute Ax but remove the maintenance of th dense matrix product $W^T F$. Hence, the proposed algorithm stably run on the limited internal memory systems with the required memory size $O(nnz(V)+size(W)+size(F)) = nnz(V)+(m+n)r$, which is much smaller than $O(mn+(n+m)r)$ for the existing algorithms [12], [13].

III. Theoretical Analysis

In this section, we analyze the convergence and complexity of Algorithm 1 and Algorithm 2.

In comparison with the previous algorithm of Hsieh & Dhillon, 2011 [13], the proposed algorithm has significant modifications for large sparse datasets by means of adding the order randomization of indexes and utilizing the sparsity of data V, model W, and instance representation F. These modifications does not affect on the convergence guarantee of algorithm. Hence, based on Theorem 1 in Hsieh & Dhillon, 2011 [13], Algorithm 2 converges to the global minimum of $f(x)$. Furthermore, based on Theorem 3 in Hsieh & Dhillon, 2011 [13], Algorithm 1 using Algorithm 2 will converge to a stationary point. In practice, we set $\varepsilon_x = 0.1$ in Algorithm 2, which is more precise than $\varepsilon_x = 0.5$ in Hsieh & Dhillon, 2011 [13].

Concerning the complexity of Algorithm 2, based on the remarks in Section II, we have Theorem 1. Furthermore, because KL divergence is a convex function over one variable and the nonnegative domain, and project Newton methods with quadratic approximation for convex functions have superlinear rate of convergence [14], [15], the average number of iterations $\bar{t}$ is small.

Theorem 1. *The complexity of Algorithm 2 is* $O(n.nnz(r)+\bar{t}r(r+n+nnz(n)))$, *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 1 is* $O(\bar{t}r(mnr+(m+n)r^2))$.

Proof. Consider the major computation in Algorithm 2, based on Formula 6, we have:

- The complexity of computing Ax in Line 2 is $O(n.nnz(r))$.
- The complexity of computing ∇f_k and $\nabla^2 f_{kk}$ in Line 4 is $O(r+nnz(n))$ because Ax and $sumA$ are computed in advance.

- The complexity of updating ∇f_k and $\nabla^2 f_{kk}$ in Line 12 is $O(r+n+nnz(n))$ because only one dimension of vector x is changed.

Hence, the complexity of Algorithm 2 is $O(n.nnz(r)+\bar{t}r(r+n+nnz(n)))$.

In addition, the complexity of computing $sumW$ and $sumF$ is $O((n+m)r)$. Hence, the complexity of a while iteration in Algorithm 1 is $O((m+n)r+mn.nnz(r)+\bar{t}mr(r+n+nnz(n)) \approx (m+n)r+\bar{t}(mnr+(m+n)r^2) \approx \bar{t}(mnr+(m+n)r^2)$. Therefore, we have Theorem 1.

For large sparse datasets, $m,n \gg r \Rightarrow O(\bar{t}(mnr+(m+n)r^2)) \approx \bar{t}(mnr)$. This complexity is raised by the operators $Ax = Ax + \Delta x A_k$ in Algorithm 2. To reduce the running time of these operators, A_k must be stored in an array to utilize CPU cache memory by accessing continuous memory cells of Ax and A_k.

IV. Experimental Evaluation

In this section, we investigate the effectiveness of the proposed algorithm via convergence and sparsity. Specially, we compare the proposed algorithm Sparse Randomized Coordinate Descent (**SRCD**) with state-of-the-art algorithms as follows:

- Multiplicative Update (**MU**) [12]: This algorithm is the original method for NMF with KL divergence.
- Cycle Coordinate Descent (**CCD**) [13]: This algorithm has the current fastest convergence because it has very low complexity of each update for one variable.

Datasets: To investigate the effectiveness of the algorithms compared, the 4 sparse datasets used are shown in Table I. The dataset Digit is downloaded from http://yann.lecun.com/, and the other tf-idf datasets Reuters21578, TDT2, and RCV1_4Class are downloaded from http://www.cad.zju.edu.cn/.

TABLE I: Summary of Datasets

Dataset (V)	n	m	nnz(V)	Sparsity(%)
Digits	784	60,000	8,994,156	80.880
Reuters21578	8,293	18,933	389,455	99.752
TDT2	9,394	36,771	1,224,135	99.645
RCV1_4Class	9,625	29,992	730,879	99.746

Environment settings: We develop the proposed algorithm SRCD in Matlab with embedded code C++ to compare them with other algorithms. We set system parameters to use only 1 CPU for Matlab and the IO time is excluded in the machine Mac Pro 8-Core Intel Xeon E5 3 GHz 32GB. In addition, the initial matrices W0 and F0 are set to the same values. The source code will be published on our homepage http://khuongnd.appspot.com/.

A. Convergence

In this section, we investigate the convergence of the objective value $D(V \| W^T F)$ versus running time by running all the compared algorithms on the four datasets with two numbers of latent factors $r=10$ and $r=20$. The experimental results are depicted in Fig. 1 and Fig. 2. From these figures, we realize two significant observations as follows:

- The proposed algorithm (SRCD) has much faster convergence than the algorithms CCD and MU.
- The sparser the datasets are, the greater the distinction between the convergence of the algorithm SRCD and the other algorithms CCD and MU is. Specially, for Digits with 81% sparsity, the algorithm SRCD's convergence is lightly faster than the convergence of the algorithms CCD and MU. However, for three more sparse datasets Reuters21578, TDT2, and RCV1 4Class with above 99% sparsity, the distance between these convergence speeds is readily apparent.

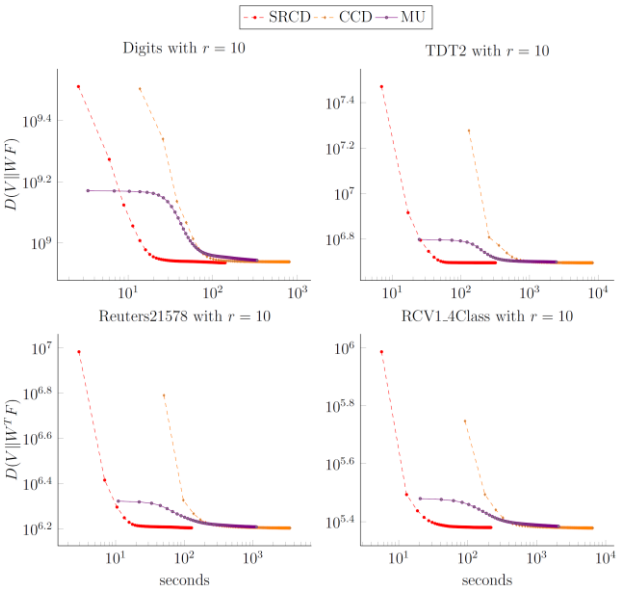

Fig. 1. Objective value $D(V \| W^T F)$ versus running time with $r=10$.

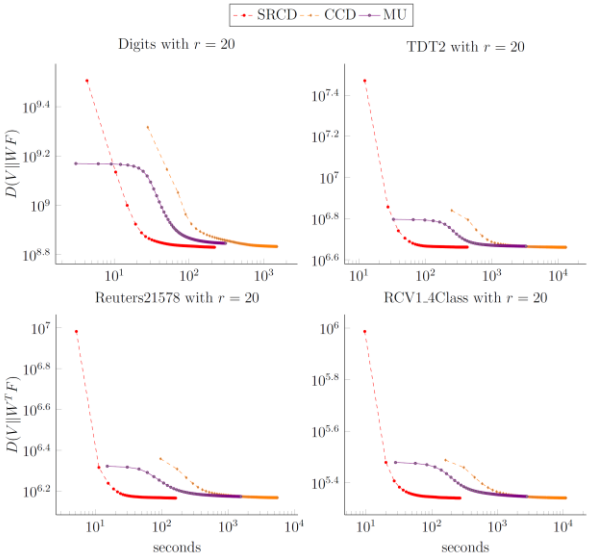

Fig. 2. Objective value $D(V \| W^T F)$ versus running time with $r=10$.

B. Sparsity of Factor Matrices

Concerning the sparsity of factor matrices W and F, the

algorithms CCD and MU does not utilize the sparsity of factor matrices. Hence, these algorithms add a small number into these factor matrices to obtain convenience in processing special numerical cases. Hence, the sparsity of factor matrices W and F for the algorithms CCD and MU both are 0%.

Although this processing may not affect other post-processing tasks such as classification and information retrieval, it will reduce the performance of these algorithms. The sparsity of (W, F) of the proposed algorithm's results is showed in Table II. These results clearly indicate that the sparse model W and the sparse representation F are attained. The results also explain why the proposed algorithm runs very fast on the sparse datasets Reuters21578, TDT2 and RCV1 4Class, when it can obtain highly sparse models and sparse representation in these highly sparse datasets.

TABLE II: 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)

C. Used Internal Memory

Table III shows the internal memory used by algorithms. From the table, we have two significant observations:

- For the dense dataset Digits, the proposed algorithm SRCD uses more internal memory than the the algorithm CCD because a considerable amount of memory is used for the indexing of matrices.
- For the sparse datasets Reuters21578, TDT2, and RCV1 4Class, the internal memory for SRCD is remarkably smaller than the internal one for MU and CCD.

These results indicate that we can conduct the proposed algorithm for huge sparse datasets with a limited internal memory machine is stable.

TABLE II: 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**

D. Running on Large Datasets

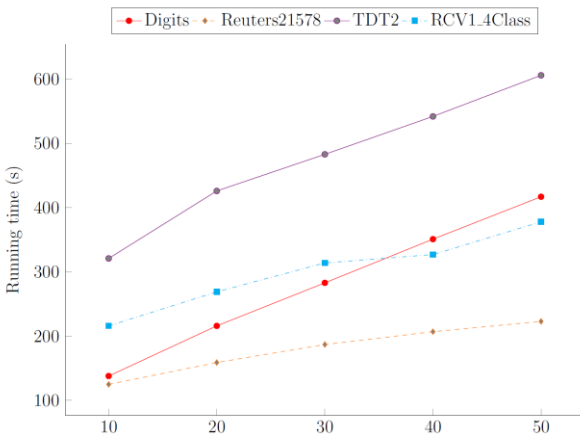

Fig. 3. Running time of 100 iterations with different number of latent component using 1 thread.

This section investigates running the proposed algorithm on large datasets with different settings. Fig. 3 shows the running time of Algorithm SRCD for 100 iterations with different number of latent component using 1 thread. Clearly, the running time linearly increases, which fits the theoretical analyses about fast convergence and linear complexity for large sparse datasets in Section III. Furthermore, concerning the parallel algorithm, the running time of Algorithm SRCD for 100 iterations significantly decreases when the number of used threads increases in Fig. 4. In addition, the running time is acceptable for large applications. Hence, these results indicate that the proposed algorithm SRCD is feasible for large scale applications.

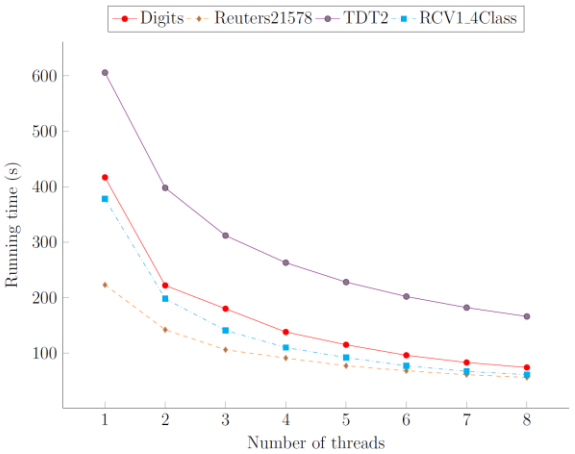

Fig. 4. Running time of 100 iterations with $r = 50$ and using different number of threads.

V. CONCLUSION AND DISCUSSION

In this paper, we propose a fast parallel randomized coordinate descent algorithm for NMF with KL divergence for large sparse datasets. The proposed algorithm attains fast convergence by means of removing duplicated computation, 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. Finally, the experimental results indicate that highly sparse models and sparse representation can be attained for large sparse datasets, a significant milestone in researching this problem. In future research, we will generalize this algorithm for nonnegative tensor factorization.

REFERENCES

[1] C. Ding, T. Li, and W. Peng, "Nonnegative matrix factorization and probabilistic latent semantic indexing: Equivalence chi-square statistic, and a hybrid method," in *Proc. the National Conference on Artificial Intelligence*, 2006, vol. 21, p. 342.

[2] N. Gillis, "The why and how of nonnegative matrix factorization," *Regularization, Optimization, Kernels, and Support Vector Machines*, vol. 12, no. 257, 2014.

[3] D. Lee, H. Seung *et al*., "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.

[4] A. Sotiras, S. M. Resnick, and C. Davatzikos, "Finding imaging patterns of structural covariance via non-negative matrix factorization," *Neuro Image*, vol. 108, pp. 1–16, 2015.

[5] Y.-X. Wang and Y.-J. Zhang, "Nonnegative matrix factorization: A comprehensive review," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 6, pp. 1336–1353, 2013.

[6] Z.-Y. Zhang, *Nonnegative Matrix Factorization: Models, Algorithms and Applications*, vol. 2, 2011.
[7] S. Sra, D. Kim, and B. Sch ̈olkopf, "Non-monotonic poisson likelihood maximization," Technical Report, Tech. Rep. 170, Max Planck Institute for Biological Cybernetics, 2008.
[8] D. Kuang, J. Choo, and H. Park, "Nonnegative matrix factorization for interactive topic modeling and document clustering," *Partitional Clustering Algorithms*, pp. 215–243, 2015.
[9] S. Z. Li, X. W. Hou, H. J. Zhang, and Q. S. Cheng, "Learning spatially localized, parts-based representation," in *Proc. the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2001, vol. 1, pp. 1–207.
[10] P. Smaragdis, "Non-negative matrix factor deconvolution; extraction of multiple sound sources from monophonic inputs," *Independent Component Analysis and Blind Signal Separation*, pp. 494–499, Springer, 2004.
[11] M. N. Schmidt, *Speech Separation Using non-Negative Features and Sparse non-Negative Matrix Factorization*, 2007.
[12] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," *Advances in Neural Information Processing Systems*, pp. 556–562, 2001.
[13] C.-J. Hsieh and I. S. Dhillon, "Fast coordinate descent methods withvariable selection for non-negative matrix factorization," in *Proc. the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2011, pp. 1064–1072.
[14] D. G. Luenberger and Y. Ye, *Linear and Nonlinear Programming*, Springer Science & Business Media, vol. 116, 2008.
[15] D. P. Bertsekas, "Projected newton methods for optimization problems with simple constraints," *SIAM Journal on Control and Optimization*, vol. 20, no. 2, pp. 221–246, 1982.

Duy Khuong Nguyen has received the B.E. and M.S. degrees in computer science from University of Engineering and Technology (UET), Vietnam National University Hanoi in 2008 and 2012, respectively. Currently, he is a Ph.D. student of joint doctoral program between Japan Advanced Institute of Science and Technology (JAIST), and University of Engineering and Technology (UET). His research interests include dimensionality reduction, algorithms, optimization and machine learning.

Tu Bao Ho is currently a professor of School of Knowledge Science, Japan Advanced Institute of Science and Technology. He received a B.T. in applied mathematics from Hanoi University of Science and Technology (1978), M.S. and Ph.D. in computer science from Pierre and Marie Curie University, Paris in 1984 and 1987. His research interests include knowledge-based systems, machine learning, knowledge discovery and data mining.