Sparse Non-negative Matrix Factorization with Generalized Kullback-Leibler Divergence

Jingwei Chen[1], Yong Feng[1], Yang Liu[2(✉)], Bing Tang[3], and Wenyuan Wu[1]

[1] Chongqing Key Laboratory of Automated Reasoning and Cognition, Chongqing Institute of Green and Intelligent Technology, CAS, Chongqing 400714, China
{chenjingwei,yongfeng,wuwenyuan}@cigit.ac.cn
[2] College of Information Science and Engineering, Chongqing Jiaotong University, Chongqing 400074, China
ly1246@qq.com
[3] School of Computer Science and Engineering, Hunan University of Science and Technology, Xiangtan 411201, China
btang@hnust.edu.cn

Abstract. Non-negative Matrix Factorization (NMF), especially with sparseness constraints, plays a critically important role in data engineering and machine learning. Hoyer (2004) presented an algorithm to compute NMF with exact sparseness constraints. The exact sparseness constraints depends on a projection operator. In the present work, we first give a very simple counterexample, for which the projection operator of the Hoyer (2004) algorithm fails. After analysing the reason geometrically, we fix this bug by adding some random terms and show that the fixed one works correctly. Based on the fixed projection operator, we propose another sparse NMF algorithm aiming at optimizing the generalized Kullback-Leibler divergence, hence named SNMF-GKLD. Experimental results show that SNMF-GKLD not only has similar effects with Hoyer (2004) on the same data sets, but is also efficient.

Keywords: Non-negative Matrix Factorization · Projection operator · Generalized Kullback-Leibler divergence

1 Introduction

Since Lee and Seung's *Nature* paper [11], Non-negative Matrix Factorization (NMF) has been extensively studied and has a great deal of applications in science and engineering. In contrast to Principal Component Analysis [9] and Independent Component Analysis [8], NMF is strictly required that the entries of both resulting matrices are non-negative. Such a constraint is very meaningful in many applications, in which the data representation is purely additive, for instance, the parts-based representation of face image data from CBCL database.

Given a non-negative matrix $V \in \mathbb{R}^{m\times n}$ and a positive integer $r < \min\{m, n\}$, the goal of NMF is to find non-negative matrices $W \in \mathbb{R}^{m\times r}$ and $H \in \mathbb{R}^{r\times n}$

H. Yin et al. (Eds.): IDEAL 2016, LNCS 9937, pp. 353–360, 2016.
DOI: 10.1007/978-3-319-46257-8_38

minimizing the function $f(W,H) = \|V - WH\|_F^2$, or the function $d(W,H) = \sum_{i,j}(V_{i,j}\log(V_{i,j}/(WH)_{i,j}) - V_{i,j} + (WH)_{i,j})$, where $\|\cdot\|_F$ is the Frobenius norm of a matrix. We call $f(W,H)$ the *Square of Euclidean Distance* (SED) and $d(W,H)$ the *Generalized Kullback-Leibler Divergence* (GKLD). The product WH is called an *NMF* for V. Note that, in most cases, WH is not equal to V. The parameter r is problem-dependent, and is set by users. Usually, r is chosen to satisfy $r \ll \min\{m,n\}$ such that WH can be thought of as a compressed form of the original data. Lee and Seung [12] presented two NMF algorithms based on multiplicative formulae whose objective functions are SED and GKLD, respectively. The two NMF algorithms can be seen as the basic ones, on which many other NMF algorithms are based. In 2004, Hoyer [6] introduced the sparseness definition for any non-zero n-dimensional vector $\boldsymbol{x}$, i.e., $\text{Sparseness}(\boldsymbol{x}) = (\sqrt{n} - \|\boldsymbol{x}\|_1/\|\boldsymbol{x}\|_2)/(\sqrt{n}-1)$, and proposed an algorithm to perform NMF with sparseness constraints (NMFSC). NMFSC adopts Lee and Seung's multiplicative formulae to optimize SED and uses a non-linear projection operator to control the sparseness of W and H.

In the present paper, we first revisit the Hoyer's projection operator [6, p. 1463]. In fact, for a kind of examples, the projection operation may fail. Geometrically, the failure case corresponds to that a direction vector of the projection operator is $\mathbf{0}$, so that the operator can not decide how to process further. We modify the operator a little to fix this bug and prove its correctness in Sect. 2. In Sect. 3, we propose a sparse NMF algorithm based on the fixed Hoyer's projection operator, named SNMF-GKLD, whose objective function is GKLD. We show experimentally that SNMF-GKLD is efficient and has similar effects with NMFSC in Sect. 4.

1.1 Related Work

Here, we only focus on algorithms to compute sparse NMF. We refer to [1,4,7] and references therein for general NMF discussion.

Li et al. proposed a local NMF algorithm [13], in which the key idea is to limit the columns of W orthogonal to each other, which makes W sparse, but H may be far from sparse. Hoyer [5] combined sparse coding and NMF. Liu et al. [15] gave a similar algorithm, but with SED replaced by GKLD. Liu and Zheng [14] presented an ℓ_p-NMF algorithms, which uses GKLD as its objective function and limits $\|W_i\|_p = 1$. For larger p, ℓ_p-NMF gives sparser representations. Hoyer [6] adopted SED as the objective function to propose an NMF algorithm (NMFSC) with exact sparseness constraints. The exact constrain on sparseness depends on a nonlinear projection that may fail for some cases. We fix the little bug in this paper. Stadlthanner et al. extended NMFSC [17] with exact sparseness constraints on W_i and H_i and disscussed the uniqueness of Hoyer's projection operator. Cichocki et al. [3] presented an NMF algorithm which uses $\|V - WH\|_F^2 + \alpha J_1(W) + \beta J_2(H)$ as its objective function. Their algorithms are modified from the basic ones and easy to implement, however, they may diverge. Pascual-Montano et al. [16] proposed the nsNMF algorithm and showed that it balances the sparseness and the capability of representing the

original data. Kim and Park [10] gave SNMF/L and SNMF/R. Tong et al. [18] proposed an NMF algorithm which combines SVD initialization technique from [2] and the extended NMFSC from [17]. Although a large number of sparse NMF algorithms have been proposed, there seems to be no sparse NMF algorithm in literature combining Hoyer's projection operator and the multiplicative formula for GKLD. In this paper, we explore this combination.

2 The Hoyer's Projection Operator Revisited

As indicated in [6] by the author, Hoyer's NMFSC algorithm is essentially the multiplication iteration for the gradient descent algorithm with a projection operator which enforces the desired degree of sparseness. Actually, the projection operator solves the following problem: given any vector $\boldsymbol{x}$, find the closest (in the euclidean sense) non-negative vector $\boldsymbol{s}$ with a given ℓ_1-norm L_1 and a given ℓ_2-norm L_2. This operator is naturally used to control the sparseness exactly. We recall this operator as in Algorithm 1.

Algorithm 1. (The projection operator in [6]).

Input: A vector $\boldsymbol{x} \in \mathbb{R}^n$, norm conditions L_1 and L_2.
Output: A closest non-negative $\boldsymbol{s}$ to $\boldsymbol{x}$ with $\|\boldsymbol{s}\|_i = L_i$, $i = 1, 2$.
1: Set $\boldsymbol{s} := \boldsymbol{x} + (L_1 - \|\boldsymbol{x}\|_1)/n\boldsymbol{e}$ with $\boldsymbol{e} = (1, 1, \cdots, 1)^T \in \mathbb{R}^n$. Set $\boldsymbol{m} := (L_1/n)\boldsymbol{e}$.
2: Set $\boldsymbol{s} := \boldsymbol{m} + \alpha(\boldsymbol{s} - \boldsymbol{m})$ with $\alpha > 0$ such that $\|\boldsymbol{s}\|_2 = L_2$.
3: **if** there exists j with $s_j < 0$ **then**
4: Set $s_j := 0$. Remove j-th coordinate of $\boldsymbol{x}$.
5: Decrease dimension $n := n - 1$.
6: **goto** 1.
7: **end if**

The projection algorithm starts from orthogonally projecting the given vector $\boldsymbol{x}$ onto the hyperplane $\sum_{i=1}^n s_i = L_1$. Next, within this hyperplane, it projects to the closest point on the joint constraint hypersphere. Namely, computing a point $\boldsymbol{s}$ satisfying $\sum_{i=1}^n s_i = L_1$ and $\|\boldsymbol{s}\|_2 = L_2$ simultaneously. This is done by, step 2, moving radially outward from the center of the sphere (the center is given by the point where all components have equal values). If the result is completely non-negative, we have arrived at our destination. If not, then we have $\|\boldsymbol{s}\|_1 > L_1$, and hence those components that attained negative values must be fixed to zero (step 4) and the new point must be projected onto the hyperplane $\sum_{i=1}^n s_i = L_1$ again (step 5 and 6), until the algorithm converges. The above iteration terminates after at most n iterations since at each iteration, the algorithm either terminates, or at least one component is set to zero and removed.

2.1 A Counterexample

It is ingenious to design the projection algorithm. Further, Stadlthanner et al. [17] proved the uniqueness of the projection operator. However, there exists a

case, for which the projection algorithm may fail. The case happens when $\boldsymbol{s} = \boldsymbol{m}$ before step 2, i.e., $\forall i \notin Z$, all components s_i's are equal. Geometrically, in this case, we can not moving from $\boldsymbol{m}$, the center of the joint constraint hypersphere, to the closest point. Instead, the algorithm will return $\boldsymbol{m}$, however, $\|\boldsymbol{m}\|_2$ is not be L_2 in general. For this case, the projection algorithm fails. Here is a simple counterexample for which Hoyer's Matlab implementation does not work:

```
>> s =[-1,-1]'
>> projfunc(s, 3, 5.598076212, 1)
```

In the code, the parameter 3 is the ℓ_1-norm and 5.598076212 is the square of the ℓ_2-norm. For this example, `projfunc` falls into an endless loop.

Note that this kind of examples does not contradict with the uniqueness in [17, Theorem 1], because it has been already indicated that the exception set for uniqueness has Lebesgue measure 0. In fact, the set of counterexamples pointed out here has exact Lebesgue measure 0.

2.2 Bug Fixing

Here is a modification to fix the above bug. The basic idea is the following: if $\boldsymbol{s} = \boldsymbol{m}$ before step 3, we re-choose $\boldsymbol{s}$ such that $\sum_{i=1}^{n} s_i = L_1$ and that $\boldsymbol{s} - \boldsymbol{m} \neq \boldsymbol{0}$. In particular, one can insert "If $\boldsymbol{s} = \boldsymbol{m}$, then randomly choose s_i such that $\sum_i s_i = L_1$. Repeat this step until $\boldsymbol{s} \neq \boldsymbol{m}$" to the location between step 1 and 2.

Proposition 1. *Algorithm 1 with the above modification correctly computes a closest non-negative $\boldsymbol{s}$ to $\boldsymbol{x}$ with $\|\boldsymbol{s}\|_i = L_i$, $i = 1, 2$.*

Proof. As indicated in Sect. 2.1, $\boldsymbol{s} = \boldsymbol{m}$ means that all components s_i's are equal. Then the orthogonal projection of $\boldsymbol{s}$ onto $\sum_{i=1}^{n} s_i = L_1$ is exactly $\boldsymbol{m}$, the center of the joint constraint hypersphere. This means that the distances between $\boldsymbol{m}(= \boldsymbol{s})$ and each intersection point of the sum and the ℓ_2-norm constraints are equal, so do the distances between $\boldsymbol{x}$ and each intersection point. Thus, in the constraint hypersphere we can move from $\boldsymbol{m}$ along any direction (i.e., either $\alpha \geq 0$ or $\alpha < 0$) to the closest point. The correctness follows.

3 Sparse NMF with GKLD

We now present a sparse NMF algorithm with the generated Kullback-Leibler divergence $d(W, H)$ as its objective function. We call the algorithm SNMF-GKLD, which can be seen as a result of combing the corresponding multiplicative iterations from [12] and the modified Hoyer's projection operator in Sect. 2.

SNMF-GKLD has a similar structure with Hoyer's NMFSC algorithm [6], however, besides the modified projection operator, it is different from NMFSC in at least the following two aspects. Firstly, we use GKLD as our objective function, while NMFSC uses SED. The two objective functions are well-known as basic objective functions for NMF. It is natural and necessary to investigate the performance of GKLD plus Hoyer's projection operator. Secondly, NMFSC

uses the additive version of gradient descent algorithm with automatically chosen stepsizes to make the objective function decrease. SNMF-GKLD gives up this step, since according to a large number of experimental observations, it seems that the objective function $d(W, H)$ always decreases after each iteration, if only one sparseness of W and H is constrained. Unfortunately, this observation does not hold when both sparseness of W and H are constrained. As a consequence, SNMF-GKLD only allows to constrain the sparseness of one factor, i.e., either W or H. An advantage is that SNMF-GKLD has a more practical efficiency than NMFSC for large size data, as the experiments will show in the next section.

Algorithm 2. (SNMF-GKLD).

Input: A non-negative matrix V of size $m \times n$; $r \in \mathbb{Z}$; $0 \leq \gamma_W \leq 1$ or $0 \leq \gamma_H \leq 1$.
Output: A matrix W of size $m \times r$ and a matrix H of size $r \times n$ minimizing $d(W, H)$.

1: Initialize W and H as random non-negative matrices.
2: If sparseness constraints on W apply, then projection each column W to be non-negative, have unchanged ℓ_2-norm, but ℓ_1-norm set to achieve desired sparseness.
3: If sparseness constraints on H apply, then projection each row H to be non-negative, have unit ℓ_2-norm and ℓ_1-norm set to achieve desired sparseness.
4: **while** converge or stop **do**
5: $\quad W_{i,a} := W_{i,a} \frac{\sum_\mu H_{a,\mu} V_{i,\mu}/(WH)_{i,\mu}}{\sum_\nu H_{a,\nu}}$.
6: $\quad$ **if** sparseness constraints on W apply **then**
7: $\quad\quad$ Project each column W to be non-negative, having unchanged ℓ_2-norm, but set ℓ_1-norm to achieve desired sparseness.
8: $\quad$ **end if**
9: $\quad H_{a,\mu} := H_{a,\mu} \frac{\sum_i W_{i,a} V_{i,\mu}/(WH)_{i,\mu}}{\sum_k W_{k,a}}$.
10: $\quad$ **if** sparseness constraints on H apply **then**
11: $\quad\quad$ Project each row H to be non-negative, having unit ℓ_2-norm and set ℓ_1-norm to achieve desired sparseness.
12: $\quad$ **end if**
13: **end while**

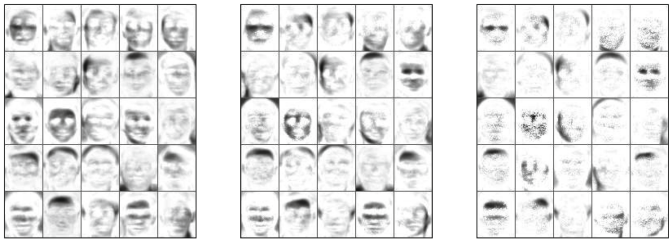
4 Experiments

We now report some experimental results, which show that SNMF-GKLD has almost the same capability as NMFSC, but is more practical for "big data". The data sets we use are CBCL, ORL and ON/OFF filtered natural image database that is included in Hoyer's NMF software package. These data sets were also used in [6] to test NMFSC. We run all experiments in Matlab® R2015b on a Win 10 PC with Intel® Core™ i5-4300U CPU and 8 GB memory. In addition, we fix the number of iterations at 300 for all experiments.

For the CBCL data, some resulting bases are shown in Fig. 1, in which the parameters are taken from [6, Fig. 3]. i.e., for (a) $\gamma_W = 0.8$, for (b) $\gamma_H = 0.8$ and for (c), $\gamma_W = 0.2$. As NMFSC, setting a high sparseness value for W results in a local representation, and global features can be learned by setting a low sparseness value for W or a high sparseness for H. Figure 2 shows bases learned by

(a) (b) (c)

Fig. 1. Features learned from the CBCL database by SNMF-GKLD



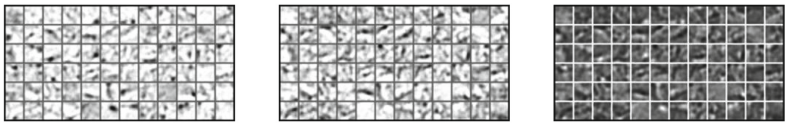
(a) (b) (c)

Fig. 2. Features learned from the ORL database by SNMF-GKLD

SNMF-GKLD for various sparseness settings, where sparseness levels were set to (a) $\gamma_W = 0.4$, (b) $\gamma_W = 0.5$ and (c) $\gamma_W = 0.6$. The representation switches from global to local with sparseness increasing. Figure 3 shows that SNMF-GKLD is also able to learn oriented features. According to these experiments, SNMF-GKLD has similar effects as NMFSC (comparing with [6, Figs. 3, 4, 5]).

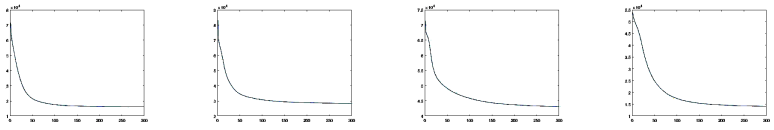
According to our experiments, for CBCL and ORL data, SNMF-GKLD has a similar efficiency with NMFSC, however, for ON/OFF-filtered natural images, SNMF-GKLD costs only about a half time of NMFSC. More specifically, for learning the features in Fig. 3, SNMF-GKLD uses 87.5 s while NMFSC uses 155.9 s. For another example, SNMF-GKLD uses 85.9 s when the sparseness of H was fixed at 0.8, while NMFSC uses 155 s. There may be two reasons for this phenomenon. Firstly, ON/OFF-filtered natural images have larger size than that of CBCL and ORL data. Secondly, NMFSC has to re-choose the stepsize to make the objective function decrease, but SNMF-GKLD omits this step thanks to the observation obtained by Fig. 4.

Figure 4 shows the evolution of the objective function $d(W, H)$ for all experiments above. It shows that the objective function decreases after each iteration in SNMF-GKLD. This can be seen as an experimental evidence for convergence. However, we lack a mathematical convergence proof for the moment.

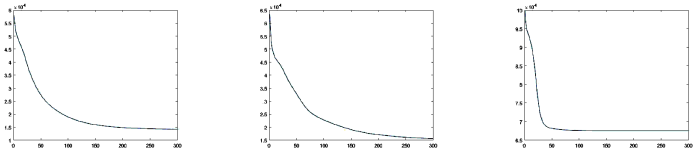


(a) (b) (c)

Fig. 3. Basis vectors learned from ON/OFF-filtered images by SNMF-GKLD



(a) (b) (c) (d)



(e) (f) (g)

Fig. 4. The evolution of $d(W, H)$ for all experiments in Figs. 1, 2 and 3

5 Conclusion and Discussion

In the present paper, we analyse and fix a bug of the key part of Hoyer's NMFSC algorithm, the projection operator, and prove the fixed version is correct. Combining the fixed projector operation with the generalized Kullback-Leibler divergence objective function, we propose a sparse NMF algorithm, SNMF-GKLD. Experiments shows that SNMF-GKLD has almost the same capability as NMFSC and that it is efficient.

SNMF-GKLD can control the sparseness exactly, but, unfortunately, the sparseness can be constrained only on one factor, i.e., W or H. How to extend SNMF-GKLD such that it can be used to control the sparseness of W and H simultaneously is an open problem. In addition, it would be very interesting to explore the theoretical convergence of SNMF-GKLD.

Acknowledgments. This work was partially supported by NSFC (11471307, 11501540, 61572024), CAS "Light of West China" Program (2014), NSF of Hunan Province (2015JJ3071) and Chongqing Research Program (cstc2015jcyjys40001).

References

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

2. Boutsidis, C., Gallopoulos, E.: SVD based initialization: a head start for nonnegative matrix factorization. Pattern Recogn. **41**(4), 1350–1362 (2008)
3. Cichocki, A., Amari, S.I., Zdunek, R., Kompass, R., Hori, G., He, Z.: Extended SMART algorithms for non-negative matrix factorization. In: Rutkowski, L., Tadeusiewicz, R., Zadeh, L.A., Żurada, J.M. (eds.) ICAISC 2006. Lecture Notes in Artificial Intelligence (LNAI), vol. 4029, pp. 548–562. Springer, Heidelberg (2006). doi:10.1007/11785231_58
4. Cichocki, A., Zdunek, R., Phan, A.H., Amari, S.I.: Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-way Data Analysis and Blind Source Separation. John Wiley & Sons Ltd., Chichester (2009)
5. Hoyer, P.O.: Non-negative sparse coding. In: Bourlard, H., Adali, T., Bengio, S., Larsen, J., Douglas, S. (eds.) NNSP 2012, pp. 557–565. IEEE, New York (2002)
6. Hoyer, P.O.: Non-negative matrix factorization with sparseness constraints. J. Mach. Learn. Res. **5**, 1457–1469 (2004)
7. Huang, Z., Zhou, A., Zhang, G.: Non-negative matrix factorization: a short survey on methods and applications. In: Li, K., Li, J., Liu, Y., Castiglione, A. (eds.) ISICA 2015. CCIS, vol. 575, pp. 331–340. Springer, Heidelberg (2012). doi:10.1007/978-3-642-34289-9_37
8. Hyvärinen, A., Karhunen, J., Oja, E.: Independent Component Analysis. John Wiley & Sons, New York (2001)
9. Jolliffe, I.T.: Principal Component Analysis, 2nd edn. Springer, New York (2002)
10. Kim, H., Park, H.: Sparse non-negative matrix factorizations via alternating non-negativity-constrained least squares for microarray data analysis. Bioinformatics **23**(12), 1495–1502 (2007)
11. Lee, D.D., Seung, H.S.: Learning the parts of objects by non-negative matrix factorization. Nature **401**(6755), 788–791 (1999)
12. Lee, D.D., Seung, H.S.: Algorithms for non-negative matrix factorization. In: Leen, T.K., Dietterich, T.G., Tresp, V. (eds.) NIPS*2000, pp. 556–562. MIT Press, Cambridge (2001)
13. Li, S.Z., Hou, X., Zhang, H., Cheng, Q.: Learning spatially localized, parts-based representation. In: CVPR 2001, vol. 1, pp. 207–212. IEEE, Los Alamitos (2001)
14. Liu, W., Zheng, N.: Learning sparse features for classification by mixture models. Pattern Recogn. Lett. **25**(2), 155–161 (2004)
15. Liu, W., Zheng, N., Lu, X.: Non-negative matrix factorization for visual coding. In: ICASSP 2003, vol. 3, pp. 293–296. IEEE, Piscataway (2003)
16. Pascual-Montano, A., Carazo, J.M., Kochi, K., Lehmann, D., Pascual-Marqui, R.D.: Nonsmooth nonnegative matrix factorization (nsNMF). IEEE Trans. Pattern Anal. Mach. Intell. **28**(3), 403–415 (2006)
17. Stadlthanner, K., Theis, F.J., Puntonet, C.G., Lang, E.W.: Extended sparse non-negative matrix factorization. In: Cabestany, J., Prieto, A., Sandoval, F. (eds.) IWANN 2005. LNCS, vol. 3512, pp. 249–256. Springer, Heidelberg (2005). doi:10.1007/11494669_31
18. Tong, M., Guo, J., Tao, S., Wu, Y.: Independent detection and self-recovery video authentication mechanism using extended NMF with different sparseness constraints. Multimedia Tools Appl. **75**(13), 8045–8069 (2016)