An Efficient Nonnegative Matrix Factorization Approach in Flexible Kernel Space *

Daoqiang Zhang[1] **Wanquan Liu**[2]
[1]Dept. of CSE, Nanjing University of Aeronautics & Astronautics, China
[2]Dept. of Computing, Curtin University of Technology, Australia
dqzhang@nuaa.edu.cn w.liu@curtin.edu.au

Abstract

In this paper, we propose a general formulation for kernel nonnegative matrix factorization with flexible kernels. Specifically, we propose the Gaussian nonnegative matrix factorization (GNMF) algorithm by using the Gaussian kernel in the framework. Different from a recently developed polynomial NMF (PNMF), GNMF finds basis vectors in the kernel-induced feature space and the computational cost is independent of input dimensions. Furthermore, we prove the convergence and nonnegativity of decomposition of our method. Extensive experiments compared with PNMF and other NMF algorithms on several face databases, validate the effectiveness of the proposed method.

1 Introduction

Nonnegative matrix factorization (NMF) is a recent linear method for finding low-dimensional representation of nonnegative high-dimensional data such as images and texts. It imposes the nonnegativity constraints in both its basis vectors (bases) and coefficients. Due to its part-based representation property [Lee and Seung, 1999], NMF and its variations have been applied to a variety of applications, such as image classification, face expression recognition, face and object recognition, document clustering, *etc* [Berry *et al.*, 2007].

Over the last decade, many variants on NMF have been proposed to improve original NMF from different perspectives. To our knowledge, most works focus on one or several of the following aspects: 1) enhancing the sparseness of representation [Li *et al.*, 2001][Hoyer, 2004][Pascual-Montano *et al.*, 2006]; 2) investigating alternative computational solutions [Berry *et al.*, 2007][Lin, 2007]; 3) introducing discriminative information to improve classification power [Zafeiriou *et al.*, 2006][Yang *et al.*, 2008]. For example, to enhance the sparseness, Li *et al.* [2001] and Hoyer [2004] imposed different extra constraints. As for alternative computational solutions, besides the well-known multiplicative update algorithms, Lin [2007] recently proposed the projected gradient methods for NMF based on bound-constrained optimization. At last, Zafeiriou *et al.* [2006] and Yang *et al.* [2008] both introduced discriminant information into NMF for better classification power.

On the other hand, NMF and its many variants are linear models, i.e. data are decomposed as a linear mixture of basis vectors. Recently, kernel methods [Shawe-Taylor and Cristianini, 2004] have been used in NMF to deal with nonlinear correlation in data. Buciu *et al.* [2008] proposed the polynomial nonnegative matrix factorization (PNMF) method, where the original data as well as the unknown basis vectors are first transformed by a nonlinear polynomial kernel mapping into a higher feature space and then a nonnegative decomposition is accomplished in the feature space. Although PNMF shows improved classification power over conventional NMF algorithms, there remain several problems unresolved yet. Firstly, only the polynomial kernel function can be used in PNMF to keep the nonnegativity constraint. Other kernel functions such as the well-known Gaussian kernel may not be adopted because of the negative solution resulting from the derivative associated from the Gaussian kernel. Secondly, although the decomposition is performed in feature space, PNMF still seeks basis vectors in the original input space and then transform them into feature space. It remains unknown how to find basis vectors directly in the feature space. Finally, at each iteration step of PNMF, the kernel matrices have to be recomputed, and thus a great deal of computational cost are required. In our previous work, we ever proposed performing NMF directly on kernel matrices, but rigorous derivation and analysis in theory was not given [Zhang *et al.*, 2006].

In this paper, we propose an alternative way for using kernel method in NMF. A general framework for kernel based NMF is presented which can efficiently use flexible kernel functions. Besides, unlike in PNMF where basis vectors are still found in original input space, our method directly seeks bases in transformed feature space, which can be further changed into a much easier kernel decomposition problem by using kernel functions. Furthermore, there is no need to repeatedly compute the kernel matrices in each iteration, and the computational cost is low. Algorithmic convergence and nonnegativity property are guaranteed by theoretical proof. Specifically, we use the Gaussian kernel in our framework and present the Gaussian nonnegative matrix factorization (GNMF) algorithm. The effectiveness of the proposed method is validated by extensive experiments on sev-

*This work is partially supported by NSFC (60875030), Doctoral Fund of MOE (200802870003) and Australia ARC Linkage grant.

eral face databases compared with PNMF and conventional NMF algorithms.

The rest of this paper is organized as follows. Section 2 reviews the standard NMF algorithm and the recently proposed PNMF algorithm. Then in Section 3, we present the flexible kernel based NMF framework and give the proposed GNMF algorithm in detail. Experimental results on several benchmark face databases are reported in Section 4. And finally, we conclude this paper and indicate some issues for future research in Section 5.

Notations: Throughout this paper, we use lowercase bold letters to denote vectors and uppercase bold letters to denote matrices, if not stated specially. The operator $\langle\cdot\rangle$ means the inner product, and $\|\cdot\|$ denotes the Frobenius norm. $\boldsymbol{A}^T$ denotes the transpose of a matrix $\boldsymbol{A}$, $\boldsymbol{A}^+$ indicates the Moore-Penrose pseudo-inverse of matrix $\boldsymbol{A}$, and $tr(\boldsymbol{A})$ means the trace operator of the corresponding matrix $\boldsymbol{A}$. The symbol $\boldsymbol{A}_{i\cdot}$ denotes the ith row vector of matrix $\boldsymbol{A}$, and $\boldsymbol{A}_{\cdot i}$ means the ith column vector of matrix $\boldsymbol{A}$. $X \geq 0$ represents the matrix is nonnegative.

2 NMF and PNMF

2.1 NMF

The key ingredient of NMF is the non-negativity constraints imposed on the two matrix factors. Assume that the observed data of the objects are represented as an $n \times m$ matrix $\boldsymbol{X}$, each column of which contains n non-negative attribute values in one of the m objects. In order to represent data or reduce the dimensionality, NMF finds two non-negative matrix factors $\boldsymbol{W}$ and $\boldsymbol{H}$ such that $\boldsymbol{X} \approx \boldsymbol{W}\boldsymbol{H}$. In general, the standard NMF problem can be formally expressed as follows [Lee and Seung, 2001]:

Problem 1 *(The NMF problem) Given an $n \times m$ nonnegative matrix $\boldsymbol{X}$ and a positive integer $r < \min\{n, m\}$, find nonnegative matrices $\boldsymbol{W}$ and $\boldsymbol{H}$ to minimize the following objective function*

$$\begin{aligned} J_1(\boldsymbol{W}, \boldsymbol{H}) &= \min_{\boldsymbol{W},\boldsymbol{H}} \frac{1}{2}\|\boldsymbol{X} - \boldsymbol{W}\boldsymbol{H}\|^2 \\ s.t. &\quad \boldsymbol{W} \geq 0, \boldsymbol{H} \geq 0. \end{aligned} \tag{1}$$

In order to obtain $\boldsymbol{W}$ and $\boldsymbol{H}$, a multiplicative update rule is given in [Lee and Seung, 2001].

2.2 PNMF

The standard NMF is a linear model, and thus it only allows linear correlation. To handle the nonlinear correlation, the polynomial NMF (PNMF) algorithm was recently proposed. The main idea of PNMF is to first transform data into higher dimensional feature space by using a polynomial kernel-induced nonlinear mapping and then perform decomposition in that feature space. Let ϕ denote the nonlinear mapping corresponding to the polynomial kernel, i.e. $k(\boldsymbol{x}, \boldsymbol{z}) = \langle \boldsymbol{x}, \boldsymbol{z}\rangle^{\{d\}} = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{z})\rangle$, then the PNMF problem can be formally expressed as follows [Buciu *et al.*, 2008]:

Problem 2 *(The PNMF problem) Given the nonnegative input data $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_m]$ and the corresponding transformed input data in polynomial feature space $\Phi(\boldsymbol{X}) = [\phi(\boldsymbol{x}_1), \phi(\boldsymbol{x}_2), ..., \phi(\boldsymbol{x}_m)]$, and a positive integer r, find nonnegative matrices $\boldsymbol{W} = [\boldsymbol{w}_1, \boldsymbol{w}_2, ..., \boldsymbol{w}_r]$ and $\boldsymbol{H}$ to minimize the following objective function*

$$\begin{aligned} J_2(\boldsymbol{W}, \boldsymbol{H}) &= \min_{\boldsymbol{W},\boldsymbol{H}} \frac{1}{2}\|\Phi(\boldsymbol{X}) - \boldsymbol{Y}\boldsymbol{H}\|^2 \\ s.t. &\quad \boldsymbol{W} \geq 0, \boldsymbol{H} \geq 0, \end{aligned} \tag{2}$$

where $\boldsymbol{Y} = [\phi(\boldsymbol{w}_1), \phi(\boldsymbol{w}_2), ..., \phi(\boldsymbol{w}_r)]$.

It is easy to see that if we expand the objective function in Eq. 2, the PNMF problem can be solved by invoking only the kernel function. In order to obtain $\boldsymbol{W}$ and $\boldsymbol{H}$, a multiplicative update rule is given as follows [Buciu *et al.*, 2008]:

$$\boldsymbol{H}_{a\mu} = \boldsymbol{H}_{a\mu} \frac{(\boldsymbol{K}_{wx})_{a\mu}}{(\boldsymbol{K}_{ww}\boldsymbol{H})_{a\mu}} \tag{3}$$

$$\boldsymbol{W}_{ia} = \boldsymbol{W}_{ia} \frac{(\boldsymbol{X}\boldsymbol{K}'_{xw})_{ia}}{(\boldsymbol{W}\boldsymbol{\Lambda}\boldsymbol{K}'_{ww})_{ia}} \tag{4}$$

where $(\boldsymbol{K}_{wx})_{a\mu} = k(\boldsymbol{w}_a, \boldsymbol{x}_\mu)$, $(\boldsymbol{K}_{ww})_{ab} = k(\boldsymbol{w}_a, \boldsymbol{w}_b)$ are kernel matrices of dimensions $r \times m$ and $r \times r$, respectively. $(\boldsymbol{K}'_{xw})_{ia} = k'(\boldsymbol{x}_i, \boldsymbol{w}_a)$, $(\boldsymbol{K}'_{ww})_{ab} = k'(\boldsymbol{w}_a, \boldsymbol{w}_b)$ are kernel matrices of dimensions $m \times r$ and $r \times r$ respectively, wherek' is the derivative of the polynomial kernel k, i.e. $k'(\boldsymbol{x}, \boldsymbol{z}) = d\langle \boldsymbol{x}, \boldsymbol{z}\rangle^{\{d-1\}}$. Λ is a diagonal matrix whose diagonal elements are $\lambda_{aa} = \sum_{j=1}^{m} \boldsymbol{H}_{aj}$.

It is noteworthy that although PNMF has used the kernel method to handle nonlinear correlations, it is restricted with the polynomialhkn kernel functions. That is because the iteration updating rule (Eq. 4) needs to compute the derivative of a kernel, while most kernel functions such as the Gaussian kernel may have negative derivatives and thus cannot remain the nonnegativity property in the decomposition. This motivates us to find alternative ways to allow more flexible kernel functions in NMF.

3 The Proposed Method

To overcome the limitations of PNMF, in this section, we propose an alternative kernel NMF framework with flexible kernels. In the following, we first give the new problem formulation, and then derive the iterative update rules and prove the convergence. Specifically, we use the Gaussian kernel in the framework and give the Gaussian NMF (GNMF) algorithm in detail at the end of this section.

3.1 Problem Formulation

Assume that the observed data of the objects are represented as an $n \times m$ matrix $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_m]$. Let ϕ be an implicit nonlinear mapping from the original input space to a high-dimensional feature space, where the inner product is defined as a kernel $k(\boldsymbol{x}, \boldsymbol{y}) = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{y})\rangle$ in the original input space. Denote $\Phi(\boldsymbol{X}) = [\phi(\boldsymbol{x}_1), \phi(\boldsymbol{x}_2), ..., \phi(\boldsymbol{x}_m)]$. Like in NMF, we want to find two non-negative matrix factors $\boldsymbol{W}$ and $\boldsymbol{H}$ such that $\Phi(\boldsymbol{X}) \approx \boldsymbol{W}\boldsymbol{H}$. However, because the explicit form of ϕ is unknown and $\phi(\boldsymbol{x}_i)$ may lie in very high or even infinite dimensional space, it is unpractical to directly decompose $\Phi(\boldsymbol{X})$ in the feature space.

Fortunately, we can solve that problem by representing the basis vectors $\boldsymbol{w}_i$ with linear combinations of transformed

data $\phi(\boldsymbol{x}_1), \phi(\boldsymbol{x}_2), ..., \phi(\boldsymbol{x}_m)$, i.e. $\boldsymbol{w}_i = \sum_{j=1}^{m} \boldsymbol{A}_{ji}\phi(\boldsymbol{x}_j) = \Phi(\boldsymbol{X})\boldsymbol{A}_{\cdot i}, i = 1, ..., r$. Denote $\boldsymbol{W} = \Phi(\boldsymbol{X})\boldsymbol{A}$, we have

$$\frac{1}{2}\|\Phi(\boldsymbol{X}) - \boldsymbol{W}\boldsymbol{H}\|^2 = \frac{1}{2}\|\Phi(\boldsymbol{X}) - \Phi(\boldsymbol{X})\boldsymbol{A}\boldsymbol{H}\|^2$$
$$= \frac{1}{2}tr(\boldsymbol{K} - 2\boldsymbol{K}\boldsymbol{A}\boldsymbol{H} + \boldsymbol{H}^T\boldsymbol{A}^T\boldsymbol{K}\boldsymbol{A}\boldsymbol{H}) \quad (5)$$

where $\boldsymbol{K} = \Phi^T(\boldsymbol{X})\Phi(\boldsymbol{X})$ is the kernel matrix. Note that each column vector of $\boldsymbol{W}$ lies in the kernel-induced feature space, and thus we cannot constrain it explicitly. Instead, we approximately constrain $\boldsymbol{A}^T\boldsymbol{K}\boldsymbol{A} \geq 0$ due to $\boldsymbol{W}^T\boldsymbol{W} = \boldsymbol{A}^T\Phi^T(\boldsymbol{X})\Phi(\boldsymbol{X})\boldsymbol{A} = \boldsymbol{A}^T\boldsymbol{K}\boldsymbol{A} \geq 0$. From Eq. 5, the flexible-kernel NMF problem can be expressed as follows:

Problem 3 *(The Flexible-Kernel NMF problem) Given the nonnegative input data $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_m]$ and the corresponding kernel matrix $\boldsymbol{K} = \Phi^T(\boldsymbol{X})\Phi(\boldsymbol{X})$, and a positive integer r, find $\boldsymbol{A}$ and nonnegative matrix $\boldsymbol{H}$ to minimize the following objective function*

$$J_3(\boldsymbol{A}, \boldsymbol{H}) = \min_{\boldsymbol{A},\boldsymbol{H}} \frac{1}{2}tr(\boldsymbol{K} - 2\boldsymbol{K}\boldsymbol{A}\boldsymbol{H} + \boldsymbol{H}^T\boldsymbol{A}^T\boldsymbol{K}\boldsymbol{A}\boldsymbol{H})$$
$$s.t. \quad \boldsymbol{A}^T\boldsymbol{K}\boldsymbol{A} \geq 0, \boldsymbol{H} \geq 0. \quad (6)$$

It is easy to see that in the flexible-kernel NMF problem, basis vectors are sought in the transformed feature space, which is apparently different from the PNMF problem. On the other hand, the objective function in Eq. 6 is biquadratic, and generally there is no closed-form solution for it. In the next subsection, we will present an alternately iterative procedure for computing the nonnegative solution.

3.2 Iterative Update Procedure

Before formally describing the derivations of the iterative update rule, we first introduce some preliminary concepts and lemmas which will be used later.

Definition 1 *(Auxiliary function) Function $G(\boldsymbol{A}, \boldsymbol{A}')$ is an auxiliary function for function $F(\boldsymbol{A})$ if the conditions*

$$G(\boldsymbol{A}, \boldsymbol{A}') \geq F(\boldsymbol{A}), G(\boldsymbol{A}, \boldsymbol{A}) = F(\boldsymbol{A}) \quad (7)$$

are satisfied.

Lemma 1 *[Lee and Seung, 2001] If G is an auxiliary function, then F is nonincreasing under the update*

$$\boldsymbol{A}^{t+1} = \arg\min_{\boldsymbol{A}} G(\boldsymbol{A}, \boldsymbol{A}^t), \quad (8)$$

where t denotes the t-th iteration.

Solution of $\boldsymbol{H}$ for given A

When $\boldsymbol{A}$ is fixed, the objective function in Eq. 6 with respect to the coefficient matrix $\boldsymbol{H} = [\boldsymbol{H}_{\cdot 1}, \boldsymbol{H}_{\cdot 2}, ..., \boldsymbol{H}_{\cdot m}]$ can be rewritten as

$$F(\boldsymbol{H}) = \frac{1}{2}tr(\boldsymbol{K} - 2\boldsymbol{K}\boldsymbol{A}\boldsymbol{H} + \boldsymbol{H}^T\boldsymbol{A}^T\boldsymbol{K}\boldsymbol{A}\boldsymbol{H})$$
$$= \frac{1}{2}tr(\boldsymbol{K}) - \sum_{i=1}^{m}\boldsymbol{K}_{i\cdot}\boldsymbol{A}\boldsymbol{H}_{\cdot i} + \frac{1}{2}\sum_{i=1}^{m}\boldsymbol{H}_{\cdot i}^T\boldsymbol{A}^T\boldsymbol{K}\boldsymbol{A}\boldsymbol{H}_{\cdot i} \quad (9)$$

From Eq. 9, it is easy to notice that different column vectors of $\boldsymbol{H}$ are independent to each other for optimization, and thus the objective function can be further simplified into column-wise form as

$$F(\boldsymbol{H}_{\cdot i}) = \frac{1}{2}tr(\boldsymbol{K}) - \boldsymbol{K}_{i\cdot}\boldsymbol{A}\boldsymbol{H}_{\cdot i} + \frac{1}{2}\boldsymbol{H}_{\cdot i}^T\boldsymbol{A}^T\boldsymbol{K}\boldsymbol{A}\boldsymbol{H}_{\cdot i} \quad (10)$$

Following [Lee and Seung, 2001], we can construct an auxiliary function of $F(\boldsymbol{H}_{\cdot i})$ in Eq. 10 as below.

Lemma 2 *If $\boldsymbol{L}(\boldsymbol{H}_{\cdot i}^t)$ is the diagonal matrix*

$$\boldsymbol{L}_{ab}(\boldsymbol{H}_{\cdot i}^t) = \delta_{ab}(\boldsymbol{A}^T\boldsymbol{K}\boldsymbol{A}\boldsymbol{H}_{\cdot i}^t)_a / \boldsymbol{H}_{ai}^t, \quad (11)$$

where δ_{ab} is the indicator function, then

$$G(\boldsymbol{H}_{\cdot i}, \boldsymbol{H}_{\cdot i}^t) = F(\boldsymbol{H}_{\cdot i}^t) + \nabla F(\boldsymbol{H}_{\cdot i}^t)(\boldsymbol{H}_{\cdot i} - \boldsymbol{H}_{\cdot i}^t)$$
$$+\frac{1}{2}(\boldsymbol{H}_{\cdot i} - \boldsymbol{H}_{\cdot i}^t)^T\boldsymbol{L}(\boldsymbol{H}_{\cdot i}^t)(\boldsymbol{H}_{\cdot i} - \boldsymbol{H}_{\cdot i}^t) \quad (12)$$

is an auxiliary function of $F(\boldsymbol{H}_{\cdot i})$ in Eq. 10.

The proof for Lemma 2 is similar as that in [Lee and Seung, 2001] and we omit it due to space limit. Then, according to Lemma 1, $\boldsymbol{H}_{\cdot i}^{t+1}$ can be computed by minimizing $G(\boldsymbol{H}_{\cdot i}, \boldsymbol{H}_{\cdot i}^t)$.

By setting $\frac{\partial G(\boldsymbol{H}_{\cdot i}, \boldsymbol{H}_{\cdot i}^t)}{\partial \boldsymbol{H}_{\cdot i}} = 0$, we have

$$\boldsymbol{H}_{\cdot i}^{t+1} = \boldsymbol{H}_{\cdot i}^t - [\boldsymbol{L}(\boldsymbol{H}_{\cdot i}^t)]^{-1}\nabla F(\boldsymbol{H}_{\cdot i}^t) \quad (13)$$

From Eqs. 10 and 11, and after some algebra operations, we obtain the update rule for $\boldsymbol{H}_{ai}$ as

$$\boldsymbol{H}_{ai}^{t+1} = \frac{\boldsymbol{H}_{ai}^t(\boldsymbol{A}^T\boldsymbol{K})_{ai}}{(\boldsymbol{A}^T\boldsymbol{K}\boldsymbol{A}\boldsymbol{H}^t)_{ai}} \quad (14)$$

Solution of $\boldsymbol{A}$ for given $\boldsymbol{H}$

When $\boldsymbol{H}$ is fixed, we want to optimize $\boldsymbol{A}$ according to the objective function in Eq. 6. For that purpose, we introduce an auxiliary matrix $\boldsymbol{B} = \boldsymbol{K}^{\frac{1}{2}}\boldsymbol{A}$, where $\boldsymbol{K}$ is the kernel matrix. However, there may be a few negative components in matrix $\boldsymbol{K}^{\frac{1}{2}}$. To keep the nonnegativity property, in this paper we project those negative values to the nearest nonnegative value, i.e. 0, and obtain both symmetric and nonnegative matrix $\boldsymbol{K}^{\frac{1}{2}}$. In our experiments, we found that only very few components of $\boldsymbol{K}^{\frac{1}{2}}$ are of negative values and the projection method works very well in practice.

From $\boldsymbol{B} = \boldsymbol{K}^{\frac{1}{2}}\boldsymbol{A}$, we have $\boldsymbol{A} = (K^{\frac{1}{2}})^{-1}\boldsymbol{B}$, then the objective function in Eq. 6 with with respect to the matrix $\boldsymbol{B} = [\boldsymbol{B}_{1\cdot}^T, \boldsymbol{B}_{2\cdot}^T, ..., \boldsymbol{B}_{m\cdot}^T]^T$ can be rewritten as

$$F(\boldsymbol{B}) = \frac{1}{2}tr(\boldsymbol{K} - 2\boldsymbol{K}^{\frac{1}{2}}\boldsymbol{B}\boldsymbol{H} + \boldsymbol{H}^T\boldsymbol{B}^T\boldsymbol{B}\boldsymbol{H})$$
$$= \frac{1}{2}tr(\boldsymbol{K}) - \sum_{i=1}^{m}\boldsymbol{B}_{i\cdot}\boldsymbol{H}\boldsymbol{K}_{\cdot i}^{\frac{1}{2}} + \frac{1}{2}\sum_{i=1}^{m}\boldsymbol{B}_{i\cdot}\boldsymbol{H}\boldsymbol{H}^T\boldsymbol{B}_{i\cdot}^T \quad (15)$$

From Eq. 15, it is obvious that different row vectors of $\boldsymbol{B}$ are independent to each other for optimization, and thus

the objective function can be further simplified into row-wise form as

$$F(\boldsymbol{B}_{i\cdot}) = \frac{1}{2}tr(\boldsymbol{K}) - \boldsymbol{B}_{i\cdot}\boldsymbol{H}\boldsymbol{K}^{\frac{1}{2}}_{\cdot i} + \frac{1}{2}\boldsymbol{B}_{i\cdot}\boldsymbol{H}\boldsymbol{H}^T\boldsymbol{B}^T_{i\cdot} \quad (16)$$

Similarly, we can construct an auxiliary function of $F(\boldsymbol{B}_{i\cdot})$ in Eq. 16 as below.

Lemma 3 *If $\boldsymbol{L}(\boldsymbol{B}^t_{i\cdot})$ is the diagonal matrix*

$$\boldsymbol{L}_{ab}(\boldsymbol{B}^t_{i\cdot}) = \delta_{ab}(\boldsymbol{B}^T_{i\cdot}\boldsymbol{H}\boldsymbol{H}^T)_a/\boldsymbol{B}^t_{ia}, \quad (17)$$

where δ_{ab} is the indicator function, then

$$G(\boldsymbol{B}_{i\cdot}, \boldsymbol{B}^t_{i\cdot}) = F(\boldsymbol{B}^t_{i\cdot}) + \nabla F(\boldsymbol{B}^t_{i\cdot})(\boldsymbol{B}_{i\cdot} - \boldsymbol{B}^t_{i\cdot}) + \frac{1}{2}(\boldsymbol{B}_{i\cdot} - \boldsymbol{B}^t_{i\cdot})^T\boldsymbol{L}(\boldsymbol{B}^t_{i\cdot})(\boldsymbol{B}_{i\cdot} - \boldsymbol{B}^t_{i\cdot}) \quad (18)$$

is an auxiliary function of $F(\boldsymbol{B}_{i\cdot})$ in Eq. 16.

Also, it is easy to prove Lemma 3 following [Lee and Seung, 2001]. Similarly, according to Lemma 1, $\boldsymbol{B}^{t+1}_{i\cdot}$ can be computed by minimizing $G(\boldsymbol{B}_{i\cdot}, \boldsymbol{B}^t_{i\cdot})$.

By setting $\frac{\partial G(\boldsymbol{B}_{i\cdot}, \boldsymbol{B}^t_{i\cdot})}{\partial \boldsymbol{B}_{i\cdot}} = 0$, we have

$$\boldsymbol{B}^{t+1}_{i\cdot} = \boldsymbol{B}^t_{i\cdot} - [\boldsymbol{L}(\boldsymbol{B}^t_{i\cdot})]^{-1}\nabla F(\boldsymbol{B}^t_{i\cdot}) \quad (19)$$

From Eq. 16 and Eq. 17, and after some algebra operations, we obtain the update rule for $\boldsymbol{B}_{ia}$ as

$$\boldsymbol{B}^{t+1}_{ia} = \frac{\boldsymbol{B}^t_{ia}(\boldsymbol{K}^{\frac{1}{2}}\boldsymbol{H}^T)_{ia}}{(\boldsymbol{B}^T\boldsymbol{H}\boldsymbol{H}^T)_{ia}} \quad (20)$$

It is obvious that $\boldsymbol{B}^{t+1}$ is nonnegative if the matrices $\boldsymbol{H}$ and $\boldsymbol{B}^t$ are nonnegative. After $\boldsymbol{B}$ is obtained, we update the matrix $\boldsymbol{A}$ as

$$\boldsymbol{A} = (K^{\frac{1}{2}})^{-1}\boldsymbol{B} \quad (21)$$

Equations 20, 21 and 14 constitute the iterative update procedure, which optimizes the matrices $\boldsymbol{H}$ and $\boldsymbol{A}$ alternatively. In the next subsection, we will prove the iterative update procedure can converge to a local optimum.

3.3 Convergence Proof

In this section, we prove the convergence of the iterative update procedure proposed in last subsection.

The iterative update procedure between $\boldsymbol{H}$ and $\boldsymbol{A}$ can be further transformed the iterative update between $\boldsymbol{H}$ and $\boldsymbol{B}$. Substituting Eq. 21 into Eq. 14, we have

$$\boldsymbol{H}^{t+1}_{ai} = \frac{\boldsymbol{H}^t_{ai}(\boldsymbol{B}^T\boldsymbol{K}^{\frac{1}{2}})_{ai}}{(\boldsymbol{B}^T\boldsymbol{B}\boldsymbol{H}^t)_{ai}} \quad (22)$$

Now the iterative update procedure consist of Eqs. 22 and 20. From Eq. 22, the updated matrix $\boldsymbol{H}^{t+1}$ is still nonnegative if the matrices $\boldsymbol{B}$ and $\boldsymbol{H}^t$ are nonnegative.

Theorem 1 *The alternative iterative update procedure*

$$\boldsymbol{H}^{t+1}_{ai} = \frac{\boldsymbol{H}^t_{ai}(\boldsymbol{B}^T\boldsymbol{K}^{\frac{1}{2}})_{ai}}{(\boldsymbol{B}^T\boldsymbol{B}\boldsymbol{H}^t)_{ai}}, \boldsymbol{B}^{t+1}_{ia} = \frac{\boldsymbol{B}^t_{ia}(\boldsymbol{K}^{\frac{1}{2}}\boldsymbol{H}^T)_{ia}}{(\boldsymbol{B}^T\boldsymbol{H}\boldsymbol{H}^T)_{ia}}$$

converges to a local optimum.

Proof. Following [Lee and Seung, 2001] and [Yang *et al.*, 2008], we define

$$F(\boldsymbol{B}, \boldsymbol{H}) = \frac{1}{2}tr(\boldsymbol{K} - 2\boldsymbol{K}^{\frac{1}{2}}\boldsymbol{B}\boldsymbol{H} + \boldsymbol{H}^T\boldsymbol{B}^T\boldsymbol{B}\boldsymbol{H})$$

From the update rule for $\boldsymbol{B}$, we have

$$F(\boldsymbol{B}^{t+1}, \boldsymbol{H}^t) \leq G(\boldsymbol{B}^{t+1}, \boldsymbol{B}^t) \leq F(\boldsymbol{B}^t, \boldsymbol{H}^t)$$

Similarly, from the update rule for $\boldsymbol{H}$, we have

$$F(\boldsymbol{B}^{t+1}, \boldsymbol{H}^{t+1}) \leq G(\boldsymbol{H}^{t+1}, \boldsymbol{H}^t) \leq F(\boldsymbol{B}^{t+1}, \boldsymbol{H}^t)$$

So $F(\boldsymbol{B}^{t+1}, \boldsymbol{H}^{t+1}) \leq F(\boldsymbol{B}^t, \boldsymbol{H}^t)$.

On the other hand, from Eq. 5, it is easy to notice that $F(\boldsymbol{B}^t, \boldsymbol{H}^t) \geq 0$. Then, $F(\boldsymbol{B}^t, \boldsymbol{H}^t)$ decreases monotonically and has lower bound, and hence $F(\boldsymbol{B}^t, \boldsymbol{H}^t)$ will converge to a local optimum. □

3.4 The GNMF Algorithm

In this section, we summarize the above analysis by presenting a specific Gaussian NMF (GNMF) algorithm by using the Gaussian kernel in the flexible-kernel NMF framework. However, it is noteworthy that our method is not confined to the Gaussian kernel, and any kind of kernels can be used. Algorithm 1 lists the GNMF algorithm in detail.

Algorithm 1: The GNMF algorithm

Input:

Kernel matrix $\{\boldsymbol{K}_{ij} = \exp(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2})\}^m_{i,j=1}$
A positive integer $r < m$
A small threshold $\varepsilon > 0$.

Initialize:

Perform SVD decomposition $\boldsymbol{K} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{U}^T$
Compute $\boldsymbol{K}^{\frac{1}{2}} = \max(\boldsymbol{U}\boldsymbol{S}^{\frac{1}{2}}\boldsymbol{U}^T, 0)$
Generate initial nonnegative matrices $\boldsymbol{B}^0$ and $\boldsymbol{H}^0$ with dimensions $m \times r$ and $r \times m$ respectively.

For $t = 1, ..., t_{max}$

1. For given $\boldsymbol{H} = \boldsymbol{H}^t$, update the matrix $\boldsymbol{B}$ as
$$\boldsymbol{B}^{t+1}_{ia} = \boldsymbol{B}^t_{ia}\frac{(\boldsymbol{K}^{\frac{1}{2}}\boldsymbol{H}^T)_{ia}}{(\boldsymbol{B}^T\boldsymbol{H}\boldsymbol{H}^T)_{ia}}$$
2. For given $\boldsymbol{B} = \boldsymbol{B}^t$, update the matrix $\boldsymbol{H}$ as
$$\boldsymbol{H}^{t+1}_{ai} = \boldsymbol{H}^t_{ai}\frac{(\boldsymbol{B}^T\boldsymbol{K}^{\frac{1}{2}})_{ai}}{(\boldsymbol{B}^T\boldsymbol{B}\boldsymbol{H}^t)_{ai}}$$
3. If $\frac{\|\boldsymbol{B}^{t+1} - \boldsymbol{B}^t\|}{\sqrt{mr}} < \varepsilon$ and $\frac{\|\boldsymbol{H}^{t+1} - \boldsymbol{H}^t\|}{\sqrt{mr}} < \varepsilon$, then break.

Output:

$\boldsymbol{A} = (\boldsymbol{K}^{\frac{1}{2}})^{-1}\boldsymbol{B}^t$ and $\boldsymbol{H} = \boldsymbol{H}^t$.

4 Experiments

In this section, we test the performance of the proposed flexible-kernel NMF method. We first compare the GNMF algorithm with standard NMF, Localized NMF (LNMF) and PNMF. Also, we replace the Gaussian kernel in our GNMF with polynomial kernel (pKNMF) and compare its performance. All the NMF algorithms use the same stopping condition (see Step 3 in Algorithm 1) and ε is set to 10^{-4}, and the maximum iteration steps t_{max} is set to 500 in all experiments. For completeness, we also report results of kernel principal component analysis with both the Gaussian kernel (gKPCA) and polynomial kernel (pKPCA).

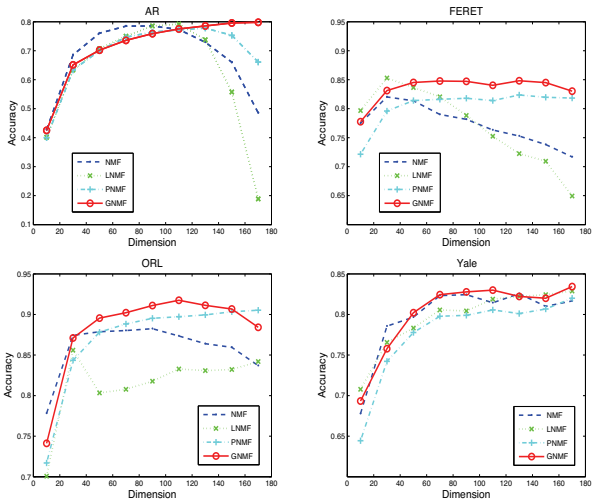

Figure 1: Classification accuracies (%) vs. different number of dimensions on AR-16x12, FERET-16x16, ORL-16x16 and Yale-16x16 databases.

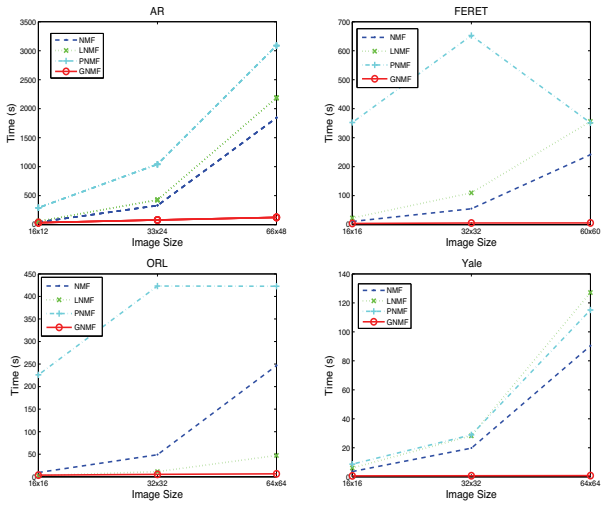

Figure 2: Running time (second) of the four algorithms under different image sizes.

4.1 Data Sets and Experimental Config

In our experiments, we use 4 benchmark face databases, i.e. AR, FERET, ORL and Yale. The AR database contains 1400 frontal facial images from 100 persons, each of which has 14 images at 2 different stages. The FERET database used in our experiments contains 200 persons, each with 2 images. The ORL database consists of 40 persons, each with 10 images. The Yale database contains 165 images from 11 persons. For each database, we resize images in 3 different scales, i.e. 66x48, 33x24 and 16x12 for AR, 60x60, 32x32 and 16x16 for FERET, 64x64, 32x32 and 16x16 for both ORL and Yale. So, there are totally 12 databases for experiments.

We evaluate performances of different algorithms using recognition accuracy. For each database, the first half of the images from each person are used for training and and the rest for testing. The Nearest Neighborhood classifier is adopted for classification after dimensionality reduction, where the number of reduced dimensions is set as $mn/(m+n)$, if without extra explanations. For PNMF, pKNMF and GNMF as well as pKPCA and gKPCA, cross-validation is used for selecting the kernel parameters d and σ respectively. For pKNMF and GNMF, features of a test image $\boldsymbol{x}_{te}$ are extracted as $(\Phi(\boldsymbol{X})\boldsymbol{A})^{+}\phi(\boldsymbol{x}_{te}) \approx \boldsymbol{A}^{+}(\Phi(\boldsymbol{X}))^{+}(\Phi^{T}(\boldsymbol{X}))^{+}\Phi^{T}(\boldsymbol{X})\phi(\boldsymbol{x}_{te}) = \boldsymbol{A}^{+}\boldsymbol{K}^{-1}\boldsymbol{K}_{te}$, where $\boldsymbol{K}_{te} = \Phi^{T}(\boldsymbol{X})\phi(\boldsymbol{x}_{te})$. All experiments are carried out on a PC with 2.7GHz CPU and 1GB RAM.

4.2 Experimental Results

We first compare GNMF with NMF, LNMF, PNMF, and Table 1 gives the classification accuracies of under fixed dimensions ($r = mn/(m+n)$) on the 12 databases. It can be seen from Table 1 that GNMF outperforms the other three algorithms in most cases and is consistently superior to PNMF. Table 1 also indicates that in most cases (except on AR) the four algorithms achieve better performances on small image size than large ones. Furthermore, Fig. 1 gives the classification accuracies of the four algorithms when different number of dimensions are used. We can see from Fig. 1 that GNMF outperforms other algorithms in most cases and is more robust to variations on dimensions.

Table 1: Classification accuracies (%) of NMF, LNMF, PNMF and GNMF on the 12 databases.

Data sets	NMF	LNMF	PNMF	GNMF
AR-16x12	67.16	60.29	75.46	**79.26**
AR-33x24	79.51	**85.69**	82.51	84.67
AR-66x48	77.69	**87.73**	80.31	81.17
FERET-16x16	76.7	75.2	82.6	**84.95**
FERET-32x32	72.55	73.6	80.4	**83.25**
FERET-60x60	65.65	73.5	79.65	**82.5**
ORL-16x16	88.25	82.75	90.65	**91.7**
ORL-32x32	86.2	81.45	87.75	**89.15**
ORL-64x64	81.7	80.3	83.75	**85.25**
Yale-16x16	82.44	82.44	81.22	**83.11**
Yale-32x32	80.78	81.78	81.78	**83.0**
Yale-64x64	77.89	81.89	80.67	**82.56**
Average	78.04	78.89	82.23	**84.21**

We also investigate the computational costs of four algorithms. It is easy to derive that the computational complexity for the iterative procedure of GNMF is $O(m^2rt)$, where m is the data size, r is the reduced dimensions and t is the iteration numbers. In comparison, the complexities of NMF and PNMF are $O(nmrt)$ and $O(nmrdt)$, where n is data dimensions and d is the order of polynomial kernel. Figure 2 plots the curves of running time vs. different image sizes for the four algorithms. As we expected, the curves of GNMF is nearly horizontal on all databases because its computational complexity is not dependent on the image size, i.e. n. Figure 2 shows that GNMF is much efficient than the other algorithms, especially for high-dimensional cases.

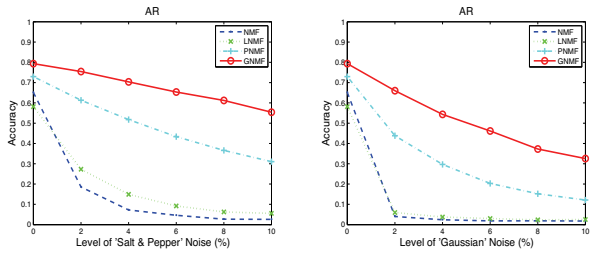

Figure 3: Classification accuracies (%) vs. different levels of noises on AR-16x12.

On the other hand, due to the use of kernel functions both PNMF and GNMF can deal with nonlinear correlations between basis vectors and thus are potentially more robust to image noises than their linear competitors. We carry out experiments on AR database when the test images are corrupted by different levels of 'Salt & Pepper' and 'Gaussian' noises respectively. Figure 3 gives the classification accuracies of four algorithms when test images are corrupted by different levels of noises. Figure 3 validate that nonlinear methods PNMF and GNMF are more advantageous for enhancing robustness to image noises than both NMF and LNMF, and GNMF consistently outperforms PNMF in both cases.

Finally, we make comparisons between kernel NMF (including pKNMF and GNMF) and kernel PCA (including pKPCA and gKPCA), and the results are given in Table 2. Table 2 indicates that pKNMF and GNMF achieve better averaged accuracies than pKPCA and gKPCA respectively across 12 databases, which validates the usefulness of kernel NMF. Furthermore, contrasting Table 2 with Table 1, it can be seen that our pKNMF outperforms PNMF in most cases and achieves better averaged accuracy.

5 Conclusion

In this paper, we proposed a general flexible-kernel based framework for nonnegative matrix factorization. We derived an alternative iteration update procedure and proved its convergence. Specifically, we proposed the Gaussian NMF (GNMF) algorithm with the Gaussian kernel and evaluated its performances on several face databases. One extra advantage of our method is that its computational complexity is independent on data dimensions and thus is potential for high-dimensional data decomposition. Moreover, GNMF can be used for negative data decomposition due to the Gaussian kernel transform and we will investigate that issue in future. Another future work is exploiting supervision information in GNMF to further enhance the discriminant power.

Acknowledgments We thank the the anonymous reviewers for their helpful comments and suggestions.

Table 2: Classification accuracies (%) of pKPCA, gKPCA, pKNMF and GNMF on the 12 databases.

Data sets	pKPCA	gKPCA	pKNMF	GNMF
AR-16x12	68.57	68.14	**79.77**	79.26
AR-33x24	74.29	74.14	**86.26**	84.67
AR-66x48	78.0	77.86	**83.57**	81.17
FERET-16x16	**85.5**	**85.5**	82.9	84.95
FERET-32x32	**85.0**	84.0	77.8	83.25
FERET-60x60	**84.5**	84.0	77.1	82.5
ORL-16x16	87.5	87.0	91.4	**91.7**
ORL-32x32	88.5	88.5	87.8	**89.15**
ORL-64x64	**87.5**	**87.5**	84.7	85.25
Yale-16x16	81.11	**85.56**	82.67	83.11
Yale-32x32	81.11	**84.44**	83.11	83.0
Yale-64x64	81.11	**84.44**	82.22	82.56
Average	81.89	82.59	83.26	**84.21**

References

[Berry *et al.*, 2007] M. Berry, M. Browne, A. Langville, V. Pauca, and R. Plemmons. Algorithms and applications for approximate nonnegative matrix factorization. *Computational Statistics & Data Analysis*, 52:155–173, 2007.

[Buciu *et al.*, 2008] I. Buciu, N. Nikolaidis, and I. Pitas. Nonnegative matrix factorization in polynomial feature space. *IEEE Trans. on NN*, 19(6):1090–695, 2008.

[Hoyer, 2004] P.O. Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of Machine Learning Research*, 5:1457–1469, 2004.

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

[Lee and Seung, 2001] D.D. Lee and H.S. Seung. Algorithms for non-negative matrix factorization. In *NIPS*, volume 13, pages 629–634, 2001.

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

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

[Pascual-Montano *et al.*, 2006] A. Pascual-Montano, J.M. Carazo, K. Kochi, D. Lehmann, and R.D. Pascual-Marqui. Non-smooth non-negative matrix factorization (nsnmf). *IEEE Trans. on PAMI*, 28(3):403–415, 2006.

[Shawe-Taylor and Cristianini, 2004] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[Yang *et al.*, 2008] J. Yang, S. Yan, Y. Fu, X. Li, and T. Huang. Non-negative graph embedding. In *CVPR*, pages 1–8, 2008.

[Zafeiriou *et al.*, 2006] S. Zafeiriou, A. Tefas, I. Buciu, and I. Pitas. Exploiting discriminant information in nonnegative matrix factorization with application to frontal face verification. *IEEE Trans. on NN*, 17(3):683–695, 2006.

[Zhang *et al.*, 2006] D. Zhang, Z.H. Zhou, and S. Chen. Non-negative matrix factorization on kernels. In *PRICAI*, pages 404–412, 2006.