MULTIPLE KERNEL NONNEGATIVE MATRIX FACTORIZATION

Shounan An [1], *Jeong-Min Yun* [2], *and Seungjin Choi* [2,3]

[1] Machine Intelligence Group, Information and Technology Lab, LG Electronics, Korea
[2] Department of Computer Science, POSTECH, Korea
[3] Division of IT Convergence Engineering, POSTECH, Korea
Email: shounan.an@lge.com, azida@postech.ac.kr, seungjin@postech.ac.kr

ABSTRACT

Kernel nonnegative matrix factorization (KNMF) is a recent kernel extension of NMF, where matrix factorization is carried out in a reproducing kernel Hilbert space (RKHS) with a feature mapping $\phi(\cdot)$. Given a data matrix $\boldsymbol{X} \in \mathbb{R}^{m\times n}$, KNMF seeks a decomposition, $\phi(\boldsymbol{X}) \approx \boldsymbol{U}\boldsymbol{V}^\top$, where the basis matrix takes the form $\boldsymbol{U} = \phi(\boldsymbol{X})\boldsymbol{W}$ and parameters $\boldsymbol{W} \in \mathbb{R}_+^{n\times r}$ and $\boldsymbol{V} \in \mathbb{R}_+^{n\times r}$ are estimated without explicit knowledge of $\phi(\cdot)$. As in most of kernel methods, the performance of KNMF also heavily depends on the choice of kernel. In order to alleviate the kernel selection problem when a single kernel is used, we present *multiple kernel NMF* (MKNMF) where two learning problems are jointly solved in unsupervised manner: (1) learning the best convex combination of kernel matrices; (2) learning parameters $\boldsymbol{W}$ and $\boldsymbol{V}$. We formulate multiple kernel learning in MKNMF as a linear programming and estimate $\boldsymbol{W}$ and $\boldsymbol{V}$ using multiplicative updates as in KNMF. Experiments on benchmark face datasets confirm the high performance of MKNMF over several existing variants of NMF, in the task of feature extraction for face classification.

Index Terms— Face recognition, multiple kernel learning, nonnegative matrix factorization.

1. INTRODUCTION

Nonnegative matrix factorization (NMF) is a method for low-rank approximation of nonnegative multivariate data, the goal of which is to approximate the data matrix (target matrix) $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n] \in \mathbb{R}_+^{m\times n}$ as a product of two nonnegative factor matrices $\boldsymbol{U} \in \mathbb{R}_+^{m\times r}$ and $\boldsymbol{V} \in \mathbb{R}_+^{n\times r}$ (2-factor decomposition), such that $\boldsymbol{X} \approx \boldsymbol{U}\boldsymbol{V}^\top$ [1]. Parameters $\boldsymbol{U}$ and $\boldsymbol{V}$ are estimated by multiplicative updates which iteratively minimize $\|\boldsymbol{X} - \boldsymbol{U}\boldsymbol{V}\|^2$ where $\|\cdot\|$ represents the Frobenious norm of a matrix. In addition to Euclidean distance, various divergence measures were also considered [2].

Various extensions of NMF have been developed. For example, additional constraints were imposed on basis vectors to improve the locality of basis vectors, leading to local NMF (LNMF) [3]. Fisher NMF (FNMF) was proposed in [4] where NMF is regularized by Fisher criterion, incorporating label information into NMF to improve the discriminative power. Recently semi-supervised NMF was presented in [5], where the data matrix and the (partial) class label matrix are jointly decomposed, sharing a factor matrix, in order to exploit both labeled and unlabeled data in the framework of NMF.

Another notable extension of NMF, which is of interest in this paper, is *kernel NMF* (KNMF) where NMF is carried out in a reproducing kernel Hilbert space (RKHS) with a feature mapping $\phi(\cdot) : \mathbb{R}^m \mapsto \mathcal{F}$ where $\mathcal{F}$ is a feature space. [6, 7]. KNMF assumes that basis vectors $\boldsymbol{u}_j$ lie within the column space of $\phi(\boldsymbol{X}) = [\phi(\boldsymbol{x}_1), \ldots, \phi(\boldsymbol{x}_n)]$, i.e., $\boldsymbol{u}_j = \sum_{i=1}^n \phi(\boldsymbol{x}_i)W_{ij}$, leading to $\boldsymbol{U} = \phi(\boldsymbol{X})\boldsymbol{W}$. Each column of $\boldsymbol{W}$ is restricted to satisfy the sum-to-one constraint. Then KNMF seeks a decomposition $\phi(\boldsymbol{X}) \approx \phi(\boldsymbol{X})\boldsymbol{W}\boldsymbol{V}^\top$, estimating parameters $\boldsymbol{W} \in \mathbb{R}_+^{n\times r}$ and $\boldsymbol{V} \in \mathbb{R}_+^{n\times r}$ without explicit knowledge of $\phi(\cdot)$ using *kernel trick*. In fact, KNMF is a special case of convex-NMF [6] and was shown to be useful in extracting spectral features from EEG signals [7].

The performance of KNMF depends on the choice of kernel, when a single kernel is used. Recent advances in kernel methods emphasized the need to consider multiple kernels or parameterizations of kernels, instead of a single fixed kernel [8, 9]. Multiple kernel learning (MKL) considers a convex (or conic) combination of kernels for classifiers. Parameters involving classifiers and the best convex combination of kernels are estimated by a joint optimization.

In this paper we incorporate MKL into KNMF, leading to *multiple kernel NMF* (MKNMF) where we jointly solve two learning problems in unsupervised manner: (1) learning the best convex combination of kernel matrices; (2) learning parameters $\boldsymbol{W}$ and $\boldsymbol{V}$. The useful characteristics of MKNMF is summarized as follows.

- To our best knowledge, this is the first work on incorporating MKL into matrix factorizations.
- MKNMF jointly learns the best convex combination of kernel matrices and parameters $\boldsymbol{W}, \boldsymbol{V}$, in *unsupervised manner*, while most of MKL methods have been developed in supervised learning.
- We develop a simple alternating minimization algorithm for MKNMF, in which the best convex combination of kernels is determined by linear programming and parameters $\boldsymbol{W}$ and $\boldsymbol{V}$ are estimated by multiplicative updates as in KNMF.
- MKNMF can handle both nonnegative and negative data, just like convex-NMF.

2. MULTIPLE KERNEL NMF

We present the objective function and an alternating minimization algorithm for MKNMF to learn the best convex combination of kernels and to estimate factor matrices.

2.1. Objective Function

Suppose that the data matrix $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{m\times n}$ is a collection of m-dimensional vectors. We consider a feature mapping $\phi(\cdot) : \mathbb{R}^m \mapsto \mathcal{F}$ where $\mathcal{F}$ is a feature space. Define $\phi(\boldsymbol{X}) = [\phi(\boldsymbol{x}_1), \ldots, \phi(\boldsymbol{x}_n)]$. Then the kernel matrix $\boldsymbol{K} \in \mathbb{R}_+^{n\times n}$ is given

by $\boldsymbol{K} = \phi^\top(\boldsymbol{X})\phi(\boldsymbol{X})$. A direct application of NMF to the feature matrix $\phi(\boldsymbol{X})$ yields

$$\phi(\boldsymbol{X}) \approx \boldsymbol{U}\boldsymbol{V}^\top, \tag{1}$$

where $\boldsymbol{U} = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_r]$ and $\boldsymbol{V} = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_r]$ are nonnegative basis matrix and encoding matrix, respectively. Without explicit knowledge of $\phi(\cdot)$, parameters $\boldsymbol{U}$ and $\boldsymbol{V}$ cannot be estimated in the factorization (1).

A simple trick to develop KNMF (or MKNMF) is to impose the constraint that basis vectors $\boldsymbol{u}_j$ lie within the column space of $\phi(\boldsymbol{X})$, i.e., $\boldsymbol{u}_j = \phi(\boldsymbol{x}_1)W_{1j} + \cdots + \phi(\boldsymbol{x}_n)W_{nj}$, leading to

$$\boldsymbol{U} = \phi(\boldsymbol{X})\boldsymbol{W}, \tag{2}$$

where W_{ij} is the (i,j)-element of the matrix $\boldsymbol{W} \in \mathbb{R}_+^{n\times r}$. We also restrict ourselves to convex combinations of the columns of $\phi(\boldsymbol{X})$, leading that each column of $\boldsymbol{W}$ satisfies the sum-to-one constraint.

Incorporating the constraint (2) into the least squares objective function for the factorization (1) yields

$$\begin{aligned}
\mathcal{J} &= \frac{1}{2}\|\phi(\boldsymbol{X}) - \boldsymbol{U}\boldsymbol{V}^\top\|^2 \\
&= \frac{1}{2}\|\phi(\boldsymbol{X}) - \phi(\boldsymbol{X})\boldsymbol{W}\boldsymbol{V}^\top\|^2 \\
&= \frac{1}{2}\mathrm{tr}\left\{\boldsymbol{K}(\boldsymbol{I} - \boldsymbol{W}\boldsymbol{V}^\top)(\boldsymbol{I} - \boldsymbol{W}\boldsymbol{V}^\top)^\top\right\},
\end{aligned} \tag{3}$$

where $\mathrm{tr}\{\cdot\}$ is the trace operator and $\boldsymbol{K} = \phi^\top(\boldsymbol{X})\phi(\boldsymbol{X})$ is the kernel matrix that is assumed to be nonnegative matrix in this paper. In fact, Eq. (3) is the objective function for KNMF. Now we consider a convex combination of kernel matrices $\boldsymbol{K}_j$, i.e., $\boldsymbol{K} = \sum_{j=1}^M \beta_j \boldsymbol{K}_j$, where $\beta_j \geq 0$ for $j = 1, \ldots, M$ and $\beta_1 + \cdots \beta_M = 1$. Substitute this relation into (3) to obtain the objective function for MKNMF

$$\mathcal{J} = \frac{1}{2}\mathrm{tr}\left\{\sum_{j=1}^M \beta_j \boldsymbol{K}_j(\boldsymbol{I} - \boldsymbol{W}\boldsymbol{V}^\top)(\boldsymbol{I} - \boldsymbol{W}\boldsymbol{V}^\top)^\top\right\}, \tag{4}$$

subject to $\boldsymbol{W} \in \mathbb{R}_+^{n\times r}$, $\boldsymbol{V} \in \mathbb{R}_+^{n\times r}$, $\mathbf{1}^\top\boldsymbol{\beta} = 1$ ($\mathbf{1} \in \mathbb{R}^M$ is the vector of all ones), and $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_M]^\top \geq 0$.

2.2. Algorithm

The optimization of the objective function (4) involves two learning problems: (1) learning $\boldsymbol{\beta}$ to determine the best convex combination of kernels; (2) estimating two factor matrices $\boldsymbol{W}$ and $\boldsymbol{V}$. We solve this optimization by an alternating minimization. We first update $\boldsymbol{\beta}$ with $\boldsymbol{W}$ and $\boldsymbol{V}$ fixed, which turns out to be a linear programming. Then we estimate $\boldsymbol{W}$ and $\boldsymbol{V}$ with $\boldsymbol{\beta}$ fixed, as in KNMF [7]. The algorithm is summarized in Algorithm 1.

2.2.1. *Optimization of $\boldsymbol{\beta}$*

Fixing $\boldsymbol{W}$ and $\boldsymbol{V}$, the objective function (4) becomes

$$\begin{aligned}
\mathcal{J} &= \frac{1}{2}\mathrm{tr}\left\{\sum_{j=1}^M \beta_j \boldsymbol{K}_j(\boldsymbol{I} - \boldsymbol{W}\boldsymbol{V}^\top)(\boldsymbol{I} - \boldsymbol{W}\boldsymbol{V}^\top)^\top\right\} \\
&= \frac{1}{2}\sum_{j=1}^M \beta_j \,\mathrm{tr}\,\{\boldsymbol{T}_j\} \\
&= \frac{1}{2}\boldsymbol{t}^\top\boldsymbol{\beta},
\end{aligned} \tag{5}$$

Algorithm 1 Algorithm outline for MKNMF.

Input: $\boldsymbol{K}_j$ for $j = 1, \ldots, M$ and r
Output: $\boldsymbol{\beta} \in \mathbb{R}^M, \boldsymbol{W} \in \mathbb{R}^{n\times r}, \boldsymbol{V} \in \mathbb{R}^{n\times r}$

1: Initialize $\boldsymbol{W}$ and $\boldsymbol{V}$
2: **repeat**
3: Apply linear programming to determine $\boldsymbol{\beta}$ which minimizes (5), given $\boldsymbol{W}$ and $\boldsymbol{V}$
4: Given $\boldsymbol{\beta}$, construct $\boldsymbol{K} = \sum_{j=1}^M \beta_j \boldsymbol{K}_j$. Then update $\boldsymbol{W}$ and $\boldsymbol{V}$ using

$$\begin{aligned}
\boldsymbol{W} &\leftarrow \boldsymbol{W} \odot \frac{\boldsymbol{K}\boldsymbol{V}}{\boldsymbol{K}\boldsymbol{W}\boldsymbol{V}^\top\boldsymbol{V}}, \\
\boldsymbol{V} &\leftarrow \boldsymbol{V} \odot \frac{\boldsymbol{K}\boldsymbol{W}}{\boldsymbol{V}\boldsymbol{W}^\top\boldsymbol{K}\boldsymbol{W}}.
\end{aligned}$$

5: **until** convergence

subject to $\mathbf{1}^\top\boldsymbol{\beta} = 1$ and $\boldsymbol{\beta} \geq 0$. The matrices $\boldsymbol{T}_j$ are defined by $\boldsymbol{T}_j = \boldsymbol{K}_j(\boldsymbol{I} - \boldsymbol{W}\boldsymbol{V}^\top)(\boldsymbol{I} - \boldsymbol{W}\boldsymbol{V}^\top)^\top$ for $j = 1, \ldots, M$ and $\boldsymbol{t} \in \mathbb{R}^M$ is the vector whose jth element is given by $[\boldsymbol{t}]_j = \mathrm{tr}\,\{\boldsymbol{T}_j\}$. The optimization of (5) with respect to $\boldsymbol{\beta}$ is the standard linear programming, which is solved using `linprog( )` in our MATLAB implementation. Please read this, which you cannot find in the final paper archived in IEEE Xplore. Our original idea was to find an optimal combination of kernels, determining $\boldsymbol{\beta}$ by LP. However, it turns out that the optimization formulation (5) subject to constraints $\mathbf{1}^\top\boldsymbol{\beta} = 1$ and $\boldsymbol{\beta} \geq 0$, yields a simple solution where $\beta_i = 1$ if $i = \arg\max_j \boldsymbol{t}_j$ and otherwise 0. In other words, $\boldsymbol{\beta}$ is a unit vector and the location of one depends on which entry of $\boldsymbol{t}$ has maximal value. Therefore, instead of running LP to determine $\boldsymbol{\beta}$, it is sufficient to search which entry of $\boldsymbol{t}$ has maximal value. The problem becomes *kernel selection*. In fact, this was first pointed out to me by Mr. Bin Shen in Purdue University. Although it becomes kernel selection, it is still valuable because this simple process finds the best kernel which minimize the cost (5). More careful study is required to analyze this kernel is really the best one for classification, which is not clear yet.

2.2.2. *Optimization of $\boldsymbol{W}$ and $\boldsymbol{V}$*

Given $\boldsymbol{\beta}$, $\boldsymbol{K} = \sum_{j=1}^M \beta_j \boldsymbol{K}_j$ is a fixed kernel matrix. Then the optimization of (4) with respect to $\boldsymbol{W}$ and $\boldsymbol{V}$ follows KNMF, which is derived below.

Suppose that the gradient of an error function has a decomposition that is of the form

$$\nabla\mathcal{J} = [\nabla\mathcal{J}]^+ - [\nabla\mathcal{J}]^-,$$

where $[\nabla\mathcal{J}]^+ > 0$ and $[\nabla\mathcal{J}]^- > 0$. Then the multiplicative updates for the parameters Θ has the form

$$\Theta \leftarrow \Theta \odot \left(\frac{[\nabla\mathcal{J}]^-}{[\nabla\mathcal{J}]^+}\right)^{\cdot\eta}, \tag{6}$$

where $\odot$ denotes the Hadamard product (element-wise product) and $\frac{\boldsymbol{A}}{\boldsymbol{B}}$ represents the element-wise division, i.e. $\left[\frac{\boldsymbol{A}}{\boldsymbol{B}}\right]_{ij} = \frac{A_{ij}}{B_{ij}}$, $(\cdot)^{\cdot\eta}$ denotes the element-wise power and η is a learning rate $(0 < \eta \leq 1)$. The multiplicative update (6) preserves the nonnegativity of the parameter Θ, while $\nabla\mathcal{J} = 0$ when the convergence is achieved [10].

Derivatives of the objective function (4) with respect to $\boldsymbol{W}$ and $\boldsymbol{V}$ are given by

$$\begin{aligned} \nabla_W \mathcal{J} &= [\nabla_W \mathcal{J}]^+ - [\nabla_W \mathcal{J}]^- \\ &= \boldsymbol{K}\boldsymbol{W}\boldsymbol{V}^\top\boldsymbol{V} - \boldsymbol{K}\boldsymbol{V}, \\ \nabla_V \mathcal{J} &= [\nabla_V \mathcal{J}]^+ - [\nabla_V \mathcal{J}]^- \\ &= \boldsymbol{V}\boldsymbol{W}^\top\boldsymbol{K}\boldsymbol{W} - \boldsymbol{K}\boldsymbol{W}. \end{aligned}$$

With these gradient calculations, invoking the relation (6) with $\eta = 1$ yields multiplicative updates for $\boldsymbol{W}$ and $\boldsymbol{V}$ in MKNMF:

$$\boldsymbol{W} \leftarrow \boldsymbol{W} \odot \frac{\boldsymbol{K}\boldsymbol{V}}{\boldsymbol{K}\boldsymbol{W}\boldsymbol{V}^\top\boldsymbol{V}}, \quad (7)$$

$$\boldsymbol{V} \leftarrow \boldsymbol{V} \odot \frac{\boldsymbol{K}\boldsymbol{W}}{\boldsymbol{V}\boldsymbol{W}^\top\boldsymbol{K}\boldsymbol{W}}. \quad (8)$$

3. NUMERICAL EXPERIMENTS

We evaluated the performance of MKNMF in the task of face recognition, compared to existing methods, including PCA (eigenface) [11], kernel PCA (KPCA) [12], NMF [1], local NMF (LNMF) [3], Fisher NMF (FNMF) [4], kernel NMF (KNMF) [7]. Features are extracted by these methods and a nearest neighbor (NN) classifier is used for classification. As the baseline, original face images are used without any feature extraction.

3.1. Datasets

We used two face image datesets: FERET [1] and Yale [2] DB. Near frontal face images were used and resized into 32×32.

- In the case of FERET dataset, we chose a subset of face images which contains 1400 images collected from 200 individuals. For each subject, 7 facial images were collected, reflecting varying facial expressions and illumination conditions.
- Yale dataset provides 11 gray scale face images for each of the 15 individuals. The images demonstrate variations in lighting condition, facial expression and with/without glasses.

3.2. Experiment Settings

We divided face images into training set $\boldsymbol{X}_{train}$ and test set $\boldsymbol{X}_{test}$. For each subject in FERET (or Yale), we assigned randomly-selected 2 (3, or 4) images into the training set and remaining images into the test set, yielding 3 different cases: 'Train-2', 'Train-3', and 'Train-4'.

Feature matrix $\boldsymbol{V}_{test}$ in the case of test set $\boldsymbol{X}_{test}$ is computed by LS projection, i.e., $\boldsymbol{V}_{test} = \boldsymbol{U}^\dagger\boldsymbol{X}_{test}$, where $\boldsymbol{U}^\dagger = (\boldsymbol{U}^\top\boldsymbol{U})^{-1}\boldsymbol{U}^\top$ and $\boldsymbol{U}$ is learned from $\boldsymbol{X}_{train}$. In the case of KNMF or MKNMF, LS projection is easily computed using a kernel trick: $\boldsymbol{V}_{test} = \left(\boldsymbol{W}^\top\boldsymbol{K}\boldsymbol{W}\right)^{-1}\boldsymbol{W}^\top\boldsymbol{K}_{test}$ where $\boldsymbol{K}_{test} = \phi(\boldsymbol{X}_{train})^\top\phi(\boldsymbol{X}_{test})$.

We used Gaussian kernel that is of the form

$$[\boldsymbol{K}]_{ij} = \exp\left\{-\frac{1}{\gamma}\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2\right\},$$

where we used 11 different values of

$$\gamma \in \{\frac{\sigma^2}{32}, \frac{\sigma^2}{16}, \frac{\sigma^2}{8}, \frac{\sigma^2}{4}, \frac{\sigma^2}{2}, \sigma^2, 2\sigma^2, 4\sigma^2, 8\sigma^2, 16\sigma^2, 32\sigma^2\},$$

to consider a convex combination of 11 Gaussian kernel matrices (with different bandwidths). The value of σ was given by the averaged norm of data vectors $\boldsymbol{x}_i$. In the case of KPCA and KNMF, the appropriate value of γ was determined by leave-one-out cross validation. All experiments were carried out on a PC with 3.4GHZ CPU and 2GB RAM.

[1] http://www.itl.nist.gov/iad/humanid/feret/feret-master.html

[2] http://cvc.yale.edu/projects/yalefaces/yalefaces.html

Table 1. Face recognition accuracy on FERET DB.

Algorithm	Train-2	Train-3	Train-4
Baseline	0.5672	0.5938	0.5837
PCA	0.6436	0.6837	0.7028
KPCA	0.6731	0.6928	0.7372
NMF	0.6527	0.6726	0.6892
FNMF	0.6673	0.6847	0.7029
LNMF	0.6738	0.7028	0.7476
KNMF	0.6624	0.7426	0.7562
MKNMF	**0.6928**	**0.7737**	**0.8163**

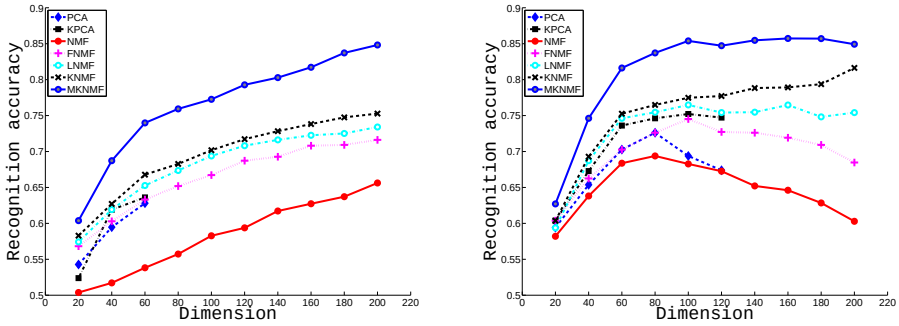
Table 2. Face recognition accuracy on Yale DB.

Algorithm	Train-2	Train-3	Train-4
Baseline	0.4820	0.5137	0.5698
PCA	0.5185	0.5416	0.6285
KPCA	0.5259	0.5616	0.6333
NMF	0.4911	0.4083	0.5295
FNMF	0.5518	0.5937	0.6327
LNMF	0.5284	0.5732	0.6538
KNMF	0.5838	0.6029	0.6650
MKNMF	**0.6259**	**0.6524**	**0.7323**

3.3. Results and Discussions

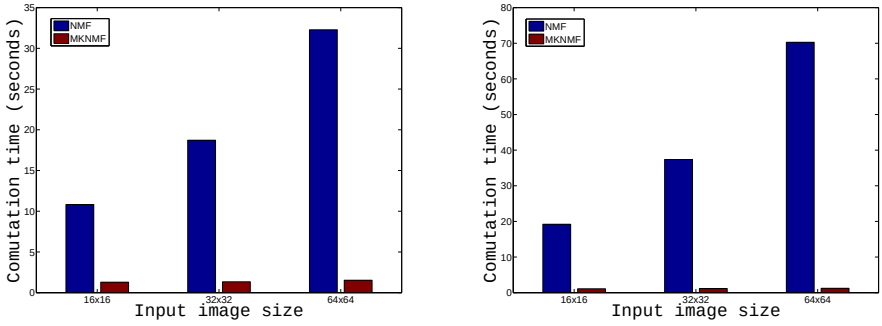
Classification accuracy averaged over 20 independent runs are summarized in Table 1 and 2 for FERET DB and Yale DB, respectively (in these experiments, the intrinsic dimension r was set as 200). The optimal value of γ in KPCA is $8\sigma^2$ and KNMF is $\frac{\sigma^2}{4}$ for FERET DB, while for Yale DB the optimal value of γ is $\frac{\sigma^2}{2}$ in KPCA and $\frac{\sigma^2}{8}$ in KNMF. MKNMF demonstrates the best performance across all cases considered here, emphasizing that multiple kernel learning indeed improves the performance of NMF. Fig. 1 shows the classification accuracy when the value of r varies from 20 to 200. Compared to single kernel-based NMF where the kernel selection is performed via cross-validation, MKNMF jointly optimizes the coefficients β_m in a convex combination of kernel matrices and the factor matrices $\boldsymbol{W}$ and $\boldsymbol{V}$, providing the best performance in these experiments.

MKNMF requires $\mathcal{O}(n^2r)$ in time, while the computational complexity of NMF is $\mathcal{O}(mnr)$, where n is number of data points, m is the dimension of input data and r is the intrinsic dimension. The computational complexity of MKNMF (or KNMF) mainly depends on the number of samples, while NMF depends on both the number of samples and the dimension of the input data. Thus, MKNMF is efficient in cases data dimension is high but the number of training samples is not large. Fig. 2 shows run time vs. different image sizes of both FERET and Yale DB, emphasizing that the run time of MKNMF remains almost constant while the run time of NMF dramatically increases when the size of images increases.



(a) Yale DB (b) FERET DB

Fig. 1. Comparison of classification accuracy when the intrinsic dimension r varies from 20 to 200, in the case of Train-4 for both Yale and FERET.



(a) Yale DB (b) FERET DB

Fig. 2. Comparison of NMF and MKNMF in terms of run time for different sizes of images.

4. CONCLUSIONS

We have presented MKNMF where we incorporated multiple kernel learning into nonnegative matrix factorization in order to alleviate the difficulty in kernel selection when a single fixed kernel matrix was used for NMF. MKNMF involved joint optimization of the coefficients β_j in a convex combination of kernel matrices and the factor matrices $\boldsymbol{W}$ and $\boldsymbol{V}$. We solved this joint optimization by an alternating minimization, where the best convex combination of kernel matrices is determined by linear programming and the factor matrices are estimated by multiplicative updates. We compared MKNMF to various NMF methods in the task of feature extraction for face recognition. Experiments on two face image datasets confirmed the high performance of MKNMF over existing other NMF methods.

Acknowledgments: This work was supported by NIPA ITRC Support Program (NIPA-2010-C1090-1031-0009), NRF Converging Research Center Program (2010K001171), and NRF WCU Program (R31-2010-000-10100-0).

5. REFERENCES

[1] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, pp. 788–791, 1999.

[2] A. Cichocki, H. Lee, Y. -D. Kim, and S. Choi, "Nonnegative matrix factorization with α-divergence," *Pattern Recognition Letters*, vol. 29, no. 9, pp. 1433–1440, 2008.

[3] S. Z. Li, X. W. Hou, H. J. Zhang, and Q. S. Cheng, "Learning spatially localized parts-based representation," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, Kauai, Hawaii, 2001, pp. 207–212.

[4] Y. Wang, Y. Jia, C. Hu, and M. Turk, "Fisher non-negative matrix factorization for learning local features," in *Proceedings of the Asian Conference on Computer Vision (ACCV)*, Jeju Island, Korea, 2004.

[5] H. Lee, J. Yoo, and S. Choi, "Semi-supervised nonnegative matrix factorization," *IEEE Signal Processing Letters*, vol. 17, no. 1, pp. 4–7, 2010.

[6] C. Ding, T. Li, and M. I. Jordan, "Convex and semi-nonnegative matrix factorizations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 1, pp. 45–55, 2010.

[7] H. Lee, A. Cichocki, and S. Choi, "Kernel nonnegative matrix factorization for spectral EEG feature extraction," *Neurocomputing*, vol. 72, pp. 3182–3190, 2009.

[8] G. R. G. Lanchriet, N. Cristianini, P. L. Bartlett, L. E. Ghaoui, and M. I. Jordan, "Learning the kernel matrix with semidefinite programming," *Journal of Machine Learning Research*, vol. 5, pp. 27–72, 2004.

[9] F. R. Bach, G. R. G. Lanchriet, and M. I. Jordan, "Multiple kernel learning, conic duality, and the SMO algorithm," in *Proceedings of the International Conference on Machine Learning (ICML)*, Banff, Canada, 2004.

[10] S. Choi, "Algorithms for orthogonal nonnegative matrix factorization," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, Hong Kong, 2008.

[11] M. Turk and A. Pentland, "Eigenfaces for recognition," *Journal of Cognitive Neuroscience*, vol. 3, no. 1, pp. 71–86, 1991.

[12] B. Schölkopf, A. J. Smola, and K. -R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Computation*, vol. 10, no. 5, pp. 1299–1319, 1998.