Adaptive Kernel Principal Component Analysis With Unsupervised Learning of Kernels

Daoqiang Zhang Zhi-Hua Zhou
National Laboratory for Novel Software Technology
Nanjing University, Nanjing 210093, China
{zhangdq, zhouzh}lamda.nju.edu.cn

Songcan Chen
Department of Computer Science and Engineering
NUAA, Nanjing 210016, China
s.chen@nuaa.edu.cn

Abstract

Choosing an appropriate kernel is one of the key problems in kernel-based methods. Most existing kernel selection methods require that the class labels of the training examples are known. In this paper, we propose an adaptive kernel selection method for kernel principal component analysis, which can effectively learn the kernels when the class labels of the training examples are not available. By iteratively optimizing a novel criterion, the proposed method can achieve nonlinear feature extraction and unsupervised kernel learning simultaneously. Moreover, a non-iterative approximate algorithm is developed. The effectiveness of the proposed algorithms are validated on UCI datasets and the COIL-20 object recognition database.

1 Introduction

As a powerful nonlinear feature extraction method, kernel principal component analysis (KPCA) [10] has been widely used in many applications [7]. The essence of KPCA is to perform principal component analysis (PCA) in the transformed high-dimensional feature space through an implicit nonlinear mapping from the original input space to the feature space. Here it is not necessary to know the explicit form of the nonlinear mapping and only the inner products between two data points in the feature space are needed. Since those inner products in feature space can be equivalently and efficiently computed by a kernel function in the original input space, KPCA elegantly avoids the 'curse of dimensionality' encountered by many classical algorithms working in high-dimensional space. However, like other kernel-based methods such as support vector machines (SVM) [7] and kernel discriminant analysis (KDA) [7], the performance of KPCA is greatly affected by the choice of the kernel and parameters. In other words, the selection of the optimal kernel and parameters is crucial for KPCA to obtain good performance.

Early research mainly focus on the selection of parameters for a certain kernel [2]. There are two widely used approaches for this purpose. The first approach empirically chooses a series of candidate values for the concerned kernel parameter, executes the learning algorithm using every candidate value, and finally assigns the value corresponding to the best performance to the kernel parameter. The second approach is the well-known cross-validation [2], which is also widely used in model selection. Both approaches are time-consuming and can hardly examine a large range of parameters. A recent advances in kernel parameter selection is to use gradient-based methods for adaptive learning the kernel parameters when class labels of the training data are available [14].

Learning the optimal kernel directly from a set of kernels has attracted much attention during the past few years [9, 11]. This kind of methods are based on the observation that in practice no kernel is the best. Thus, seeking the optimal combination of a set of kernels seems more reasonable than only adjusting the parameters of a single kernel. This kind of kernel selection methods include kernel alignment [3], learning the kernel matrix directly with semi-definite programming [6], hyperkernels [9, 11], and idealized kernels [5], etc. However, most of those methods require that the class labels of the training examples are available. Since KPCA works in an unsupervised learning setting where the class labels of the training examples are not known, those

methods can hardly be applied to KPCA directly. Recently, Yang et al. [12] proposed a novel kernel selection method called fisher+kernel criterion for KDA, where the key idea is to generate the feature matrix from multiple kernels. Yet this method is still only applicable when the class labels of training examples are available.

This paper proposes Adaptive Kernel Principal Component Analysis (A-KPCA), which can effectively learn the kernels under the unsupervised learning setting. Inspired by [12], we first transform the original 1D input vectors into 2D feature matrices through a set of nonlinear mappings induced from different kernels, each corresponding to one column of the 2D feature matrix. Then, two coupled sets of projective vectors are extracted from those feature matrices using an iterative procedure. One set of projective vectors corresponds to the column direction of feature matrices and is used for nonlinear feature exaction, while the other corresponds to the row direction of feature matrices and is used for searching the optimal combination of kernels. Moreover, an efficient non-iterative algorithm is proposed to approximate A-KPCA. The effectiveness of the proposed algorithms are validated by extensive experiments.

2 Kernel Principal Component Analysis

Given a training set $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n\}$, where the training examples $\boldsymbol{x}_i \in \Re^n$. Let $\psi : \boldsymbol{x} \in X \rightarrow \psi(\boldsymbol{x}) \in F$ be a nonlinear mapping from the original input space X to a high-dimensional feature space F. The inner product in F is defined as a kernel $\mathcal{K}(\boldsymbol{x}, \boldsymbol{y}) = \psi(\boldsymbol{x})^{\mathrm{T}}\psi(\boldsymbol{y})$ in the original input space. Denote $\boldsymbol{\Psi} = (\psi(\boldsymbol{x}_1), \psi(\boldsymbol{x}_2), ..., \psi(\boldsymbol{x}_n))$, and $\bar{\boldsymbol{\psi}} = 1/n \sum_{i=1}^{n} \psi(\boldsymbol{x}_i)$. Without loss of generality, assume that the data are centered in F, i.e. $\bar{\boldsymbol{\psi}} = 0$, then the total scatter matrix is defined as $\boldsymbol{S}_t = \sum_{i=1}^{n}(\psi(\boldsymbol{x}_i) - \bar{\boldsymbol{\psi}})(\psi(\boldsymbol{x}_i) - \bar{\boldsymbol{\psi}})^{\mathrm{T}} = \boldsymbol{\Psi}\boldsymbol{\Psi}^{\mathrm{T}}$.

KPCA uses the criterion shown in Eq. 1 to compute the optimal projective vector $\boldsymbol{v}$.

$$J(\boldsymbol{v}) \quad = \quad \sum_{k=1}^{n} \|\boldsymbol{v}^{\mathrm{T}}\boldsymbol{\psi}(\boldsymbol{x}_i)\|^2 = \boldsymbol{v}^{\mathrm{T}}\boldsymbol{S}_t\boldsymbol{v} \qquad (1)$$

Maximizing Eq. 1 is equivalent to solving the eigenvalue problem: To find $\lambda \geq 0$ and eigenvectors $\boldsymbol{v} \in F - \{0\}$ satisfying $\lambda\boldsymbol{v} = \boldsymbol{S}_t\boldsymbol{v}$. It is easy to verify that all solutions $\boldsymbol{v}$ with $\lambda \neq 0$ lie within the span of $\{\psi(\boldsymbol{x}_1), \psi(\boldsymbol{x}_2), ..., \psi(\boldsymbol{x}_n)\}$, i.e. $\boldsymbol{v} = \boldsymbol{\Psi}\boldsymbol{q}$. Thus, we can consider the equivalent problem shown in Eq. 2.

$$\lambda\boldsymbol{q} = \boldsymbol{K}\boldsymbol{q} \qquad (2)$$

Here $\boldsymbol{K} = \boldsymbol{\Psi}^{\mathrm{T}}\boldsymbol{\Psi}$ is the $n \times n$ kernel matrix. Suppose $\boldsymbol{q}_1, \boldsymbol{q}_2, ..., \boldsymbol{q}_d$ are the solutions of Eq. 2 corresponding to the largest d eigenvalues, then $\boldsymbol{v}_i = \boldsymbol{\Psi}\boldsymbol{q}_i$ is the solution to Eq. 1. To extract features for a new example $\boldsymbol{x}$ with KPCA, one simply projects the image $\psi(\boldsymbol{x})$ onto $\boldsymbol{v}_i$ as shown in Eq. 3.

$$\boldsymbol{v}_i^{\mathrm{T}}\psi(\boldsymbol{x}) \quad = \quad \boldsymbol{q}_i^{\mathrm{T}}\boldsymbol{\Psi}^{\mathrm{T}}\psi(\boldsymbol{x}) = \sum_{j=1}^{n} \boldsymbol{q}_{ij}\mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}) \qquad (3)$$

Eqs. 2 and 3 show that the computation of KPCA components does not involve the nonlinear mapping ψ, and only the kernel is needed. As mentioned before, the performance of KPCA is greatly affected by the selection of kernels and parameters.

3 A-KPCA

Let $\phi_i : \boldsymbol{x} \in X \rightarrow \phi_i(\boldsymbol{x}) \in \mathcal{H}_i, i \in \{1, 2, ..., f\}$ be a set of nonlinear mappings from the original input space X to a high-dimensional feature space $\mathcal{H}_i$. The inner products in $\mathcal{H}_i$ are defined as the kernels $\mathcal{K}^i(\boldsymbol{x}, \boldsymbol{y}) = \phi_i(\boldsymbol{x})^{\mathrm{T}}\phi_i(\boldsymbol{y})$ respectively. From ϕ_i, we define $\hat{\phi}_i : \boldsymbol{x} \in X \rightarrow \hat{\phi}_i(\boldsymbol{x}) = (\underbrace{\boldsymbol{0}^{\mathrm{T}}...\boldsymbol{0}^{\mathrm{T}}}_{1...i-1}, \phi_i(\boldsymbol{x})^{\mathrm{T}}, \underbrace{\boldsymbol{0}^{\mathrm{T}}...\boldsymbol{0}^{\mathrm{T}}}_{i+1...f})^{\mathrm{T}} \in \mathcal{H}, i \in \{1, 2, ..., f\}$. Here $\mathcal{H}$ is the Hilbert space as the direct sum of $\mathcal{H}_i$ and the j-th $\boldsymbol{0}$ vector lies in $\mathcal{H}_j$. The inner product in $\mathcal{H}$ can be defined as: $\hat{\phi}_i(\boldsymbol{x})^{\mathrm{T}}\hat{\phi}_j(\boldsymbol{y}) = 0$ $(i \neq j)$ and $\hat{\phi}_i(\boldsymbol{x})^{\mathrm{T}}\hat{\phi}_i(\boldsymbol{y}) = \phi_i(\boldsymbol{x})^{\mathrm{T}}\phi_i(\boldsymbol{y}) = \mathcal{K}^i(\boldsymbol{x}, \boldsymbol{y})$.

For an original training example $\boldsymbol{x}_k$, map it into the high-dimensional feature space with all $\hat{\phi}_i$'s and align the mapped vectors column by column to form a 2D feature matrix, denoted by $\boldsymbol{\Phi}_k = (\hat{\phi}_1(\boldsymbol{x}_k), \hat{\phi}_2(\boldsymbol{x}_k), ..., \hat{\phi}_f(\boldsymbol{x}_k))$. Thus, the original vector-based representation of training data turns into a matrix-based representation. Denote $\bar{\boldsymbol{\Phi}} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{\Phi}_i$, and without loss of generality we assume that Φ_k's have zero means, i.e. $\bar{\boldsymbol{\Phi}} = 0$.

For a series of 2D feature matrices $\boldsymbol{\Phi}_1, \boldsymbol{\Phi}_2, ..., \boldsymbol{\Phi}_n$, we want to seek two matrices $\boldsymbol{U}$ and $\boldsymbol{R}$, such that the criterion shown in Eq. 4 is optimized.

$$J(\boldsymbol{U}, \boldsymbol{R}) = \sum_{k=1}^{n} \|\boldsymbol{U}^{\mathrm{T}}\boldsymbol{\Phi}_k\boldsymbol{R}\|_F^2 \qquad (4)$$

Here $\|\cdot\|_F$ is the Frobenius norm of matrix, $\boldsymbol{U} = (\boldsymbol{u}_1, \boldsymbol{u}_2, ..., \boldsymbol{u}_d)$ is the projective vectors corresponding to the column direction of $\boldsymbol{\Phi}_k$ and its purpose is for nonlinear feature extraction, while $\boldsymbol{R} = (\boldsymbol{r}_1, \boldsymbol{r}_2, ..., \boldsymbol{r}_g)$ is the projective vectors corresponding to the row direction of $\boldsymbol{\Phi}_k$ and its purpose is for kernel selection.

It is easy to verify that $\boldsymbol{U} \subseteq span(\hat{\phi}_i(\boldsymbol{x}_k)), 1 \leq i \leq f, 1 \leq k \leq n$. Denote $\boldsymbol{\Phi} = (\boldsymbol{\Phi}_1, \boldsymbol{\Phi}_2, ..., \boldsymbol{\Phi}_n)$, then $\boldsymbol{U} = \boldsymbol{\Phi}\boldsymbol{L}$. The size of the original $\boldsymbol{\Phi}$ is $f \times n$, which is generally very large. To reduce the size of $\boldsymbol{\Phi}$, we replace it with $\tilde{\boldsymbol{\Phi}} = (\sum_{i=1}^{f} \hat{\phi}_i(\boldsymbol{x}_1), \sum_{i=1}^{f} \hat{\phi}_i(\boldsymbol{x}_2), ..., \sum_{i=1}^{f} \hat{\phi}_i(\boldsymbol{x}_n))$ so that $\boldsymbol{U} = \tilde{\boldsymbol{\Phi}}\boldsymbol{L}, \boldsymbol{L} \in \Re^{n \times d}$. Thus, Eq. 4 can be replaced

by the criterion shown in Eq. 5.

$$J(\boldsymbol{L},\boldsymbol{R}) = \sum_{k=1}^{n} \|\boldsymbol{L}^{\mathrm{T}}\tilde{\boldsymbol{\Phi}}^{\mathrm{T}}\boldsymbol{\Phi}_k\boldsymbol{R}\|_F^2 = \sum_{k=1}^{n} \|\boldsymbol{L}^{\mathrm{T}}\boldsymbol{K}_k\boldsymbol{R}\|_F^2 \quad (5)$$

Here we constrain $\boldsymbol{L} \in \Re^{n\times d}, \boldsymbol{L}^{\mathrm{T}}\boldsymbol{L} = \boldsymbol{I}$ and $\boldsymbol{R} \in \Re^{f\times g}, \boldsymbol{R}^{\mathrm{T}}\boldsymbol{R} = \boldsymbol{I}$, and $\boldsymbol{K}_k = \tilde{\boldsymbol{\Phi}}^{\mathrm{T}}\boldsymbol{\Phi}_k$ is the $n \times f$ kernel matrix, whose (i,j)-th element is computed according to Eq. 6.

$$\boldsymbol{K}_k(i,j) = (\sum_{l=1}^{f} \hat{\phi}_l(\boldsymbol{x}_i))^{\mathrm{T}}\hat{\phi}_j(\boldsymbol{x}_k) = \mathcal{K}^j(\boldsymbol{x}_i,\boldsymbol{x}_k) \quad (6)$$

Eq. 6 indicates that the kernel matrices in Eq. 5 can be directly computed from the kernel function bank $\{\mathcal{K}^i\}$. To the best of our knowledge, there is no closed form solution to Eq. 5. Inspired by [13], the following theorem presents an iterative procedure to solve this problem.

Theorem 1 *Let $\boldsymbol{L}$ and $\boldsymbol{R}$ be the optimal solution to Eq. 5, then: (1) For a given $\boldsymbol{R}$, $\boldsymbol{L}$ consists of the d eigenvectors of the matrix $\boldsymbol{M}_L = \sum_{k=1}^{n} \boldsymbol{K}_k\boldsymbol{R}\boldsymbol{R}^{\mathrm{T}}\boldsymbol{K}_k^{\mathrm{T}}$ corresponding to the largest d eigenvalues; (2) For a given $\boldsymbol{L}$, $\boldsymbol{R}$ consists of the g eigenvectors of the matrix $\boldsymbol{M}_R = \sum_{k=1}^{n} \boldsymbol{K}_k^{\mathrm{T}}\boldsymbol{L}\boldsymbol{L}^{\mathrm{T}}\boldsymbol{K}_k$ corresponding to the largest g eigenvalues.*

Proof. Equation 5 can be rewritten as

$$\begin{aligned}\sum_{k=1}^{n} \|\boldsymbol{L}^{\mathrm{T}}\boldsymbol{K}_k\boldsymbol{R}\|_F^2 &= \sum_{k=1}^{n} trace(\boldsymbol{L}^{\mathrm{T}}\boldsymbol{K}_k\boldsymbol{R}\boldsymbol{R}^{\mathrm{T}}\boldsymbol{K}_k^{\mathrm{T}}\boldsymbol{L}) \\ &= trace(\boldsymbol{L}^{\mathrm{T}}\sum_{k=1}^{n}(\boldsymbol{K}_k\boldsymbol{R}\boldsymbol{R}^{\mathrm{T}}\boldsymbol{K}_k^{\mathrm{T}})\boldsymbol{L}) \\ &= trace(\boldsymbol{L}^{\mathrm{T}}\boldsymbol{M}_L\boldsymbol{L}) \quad (7)\end{aligned}$$

Here, for a given $\boldsymbol{R}$, the maximum of Eq. 5 or Eq. 7 is obtained only if $\boldsymbol{L}$ consists of the d eigenvectors of the matrix $\boldsymbol{M}_L$ corresponding to the largest d eigenvalues. Similarly, for a given $\boldsymbol{L}$, the maximum of Eq. 5 or $trace(\boldsymbol{R}^{\mathrm{T}}\boldsymbol{M}_R\boldsymbol{R})$ is obtained only if $\boldsymbol{R}$ consists of the g eigenvectors of the matrix $\boldsymbol{M}_R$ corresponding to the largest g eigenvalues. □

Theorem 1 provides an iterative procedure for computing $\boldsymbol{L}$ and $\boldsymbol{R}$, i.e. the A-KPCA algorithm, as shown in **Algorithm 1**.

After obtaining $\boldsymbol{L}$ and $\boldsymbol{R}$ by A-KPCA, we can use them to extract the nonlinear features for an unseen instance $\boldsymbol{x}$. First, construct the kernel matrix $\boldsymbol{K}_{test}(i,j) = \mathcal{K}^j(\boldsymbol{x}_i,\boldsymbol{x})$. Then, project $\boldsymbol{K}_{test}$ according to $\boldsymbol{C} = \boldsymbol{L}^{\mathrm{T}}\boldsymbol{K}_{test}\boldsymbol{R}$. After that, the nonlinear features are contained in $\boldsymbol{C}$.

Algorithm 1: A-KPCA
Input: Training set $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n\}$.
Output: Matrices $\boldsymbol{L}$ and $\boldsymbol{R}$.

1. Construct the kernel matrix $\boldsymbol{K}_k$ for each $\boldsymbol{x}_k$ using Eq. 6.
2. Obtain initial $\boldsymbol{R}_0$ and set $i \leftarrow 1$.
3. For given $\boldsymbol{R}_{i-1}$, Compute the d eigenvectors $\boldsymbol{L}_i = (\boldsymbol{l}_1, \boldsymbol{l}_2, ..., \boldsymbol{l}_d)$ of $\boldsymbol{M}_L = \sum_{k=1}^{n} \boldsymbol{K}_k\boldsymbol{R}\boldsymbol{R}^{\mathrm{T}}\boldsymbol{K}_k^{\mathrm{T}}$ corresponding to the largest d eigenvalues.
4. For given $\boldsymbol{L}_i$, Compute the g eigenvectors $\boldsymbol{R}_i = (r_1, r_2, ..., r_g)$ of $\boldsymbol{M}_R = \sum_{k=1}^{n} \boldsymbol{K}_k^{\mathrm{T}}\boldsymbol{L}\boldsymbol{L}^{\mathrm{T}}\boldsymbol{K}_k$ corresponding to the largest g eigenvalues.
5. Set $i \leftarrow i+1$, and goto step 3 until convergence.

4 Approximate A-KPCA

In the above section, we have derived an iterative procedure to compute the matrices $\boldsymbol{L}$ and $\boldsymbol{R}$. In this section, we will develop a non-iterative algorithm to approximate A-KPCA in a more efficient way.

Given training set $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n\}$, suppose that the kernel matrices $\boldsymbol{K}_k$'s have been computed using Eq. 6. Then, instead of optimizing Eq. 5 directly, we adopt the alternative criterion shown in Eq. 8.

$$J(\boldsymbol{L},\boldsymbol{R}) = \sum_{k=1}^{n} \|\boldsymbol{L}^{\mathrm{T}}\boldsymbol{K}_k\|_F^2 + \sum_{k=1}^{n} \|\boldsymbol{K}_k\boldsymbol{R}\|_F^2 \quad (8)$$

where $\boldsymbol{L} \in \Re^{n\times d}, \boldsymbol{L}^{\mathrm{T}}\boldsymbol{L} = \boldsymbol{I}$ and $\boldsymbol{R} \in \Re^{f\times g}, \boldsymbol{R}^{\mathrm{T}}\boldsymbol{R} = \boldsymbol{I}$. Comparing Eq. 8 with Eq. 5, we can find that $\boldsymbol{L}$ and $\boldsymbol{R}$ are not tangled together any more, and therefore can be computed analytically and simultaneously.

Theorem 2 presents a closed form solution to Eq. 8, i.e. the approximate A-KPCA algorithm, as shown in **Algorithm 2**. The proof of the theorem is similar to that of PCA [4] and therefore we omit it due to the page limit.

Theorem 2 *The problem in Eq. 8 has closed form solution $\boldsymbol{L}$ and $\boldsymbol{R}$: (1) $\boldsymbol{L}$ consists of the d eigenvectors of the matrix $\boldsymbol{M}_L = \sum_{k=1}^{n} \boldsymbol{K}_k\boldsymbol{K}_k^{\mathrm{T}}$ corresponding to the largest d eigenvalues; (2) $\boldsymbol{R}$ consists of the g eigenvectors of the matrix $\boldsymbol{M}_R = \sum_{k=1}^{n} \boldsymbol{K}_k^{\mathrm{T}}\boldsymbol{K}_k$ corresponding to the largest g eigenvalues.*

5 Experiments

Experiments are conducted to compare the proposed algorithms, A-KPCA and approximate A-KPCA, with traditional PCA and KPCA with single kernel on an toy exam-

Algorithm 2: Approximate A-KPCA
Input: Training set $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n\}$.
Output: Matrices $\boldsymbol{L}$ and $\boldsymbol{R}$.

1. Construct the kernel matrix $\boldsymbol{K}_k$ for each $\boldsymbol{x}_k$ using Eq. 6.
2. Compute the d eigenvectors $\boldsymbol{L} = (\boldsymbol{l}_1, \boldsymbol{l}_2, ..., \boldsymbol{l}_d)$ of $\boldsymbol{M}_L = \sum_{k=1}^{n} \boldsymbol{K}_k \boldsymbol{K}_k^{\mathrm{T}}$ corresponding to the largest d eigenvalues.
3. Compute the g eigenvectors $\boldsymbol{R} = (\boldsymbol{r}_1, \boldsymbol{r}_2, ..., \boldsymbol{r}_g)$ of $\boldsymbol{M}_R = \sum_{k=1}^{n} \boldsymbol{K}_k^{\mathrm{T}} \boldsymbol{K}_k$ corresponding to the largest g eigenvalues.

ple (see Figure 1), four UCI datasets [1] (*Ionosphere*, *Balance*, *Glass* and *Heart*), and the COIL-20 database for object recognition [8]. For each UCI dataset, half of the data are used for training, and the rest for testing. For COIL-20 database the first 12 images per object are used for training and the rest for testing.

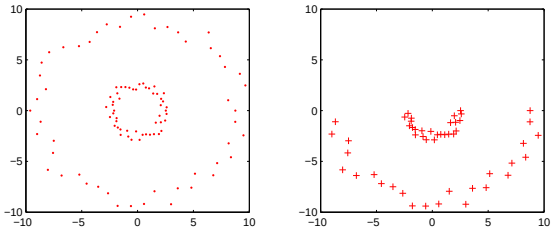

Figure 1. Left: The whole 2D toy dataset, with 2 classes (corresponding to the outer/inner circles respectively). Right: the training set.

In the experiments we adopted the Gaussian kernel function $\mathcal{K}^i(x, y) = exp(\|\boldsymbol{x}-\boldsymbol{y}\|^2/2\sigma_i^2)$ with width $\sigma_i = \sigma_0 \times i$, $i = 1, 2, ..., 5$, where σ_0 is the standard variation of training data. We denote KPCA with the above 5 different kernels as KPCA-1 to KPCA-5, respectively. In A-KPCA and approximate A-KPCA, the value of g (number of columns in R) is set to 1. After feature extraction, the well-known nearest neighbor (1-NN) classifier is used for classification.

Figure 2 compares the classification accuracies of PCA, KPCA (with different kernels) and A-KPCA under different dimensions. It can be seen from Figure 2 that in most cases, A-KPCA outperforms PCA and KPCA-1 to KPCA-5 greatly. It is impressive that on almost all data sets the proposed A-KPCA consistently outperforms PCA and KPCAs no matter which dimension is considered. A-KPCA achieves the best performance on all the data sets when a relatively large dimension is used.

Figure 2 also indicates that among KPCAs with single kernel, i.e. KPCA-1 to KPCA-5, none could always achieve the highest accuracy. For example, when large dimension is used, KPCA-1 outperforms KPCA-2 to KPCA-5 on *Ionosphere*, while on *Glass* KPCA-1 is with the lowest accuracy among KPCA-1 to KPCA-5. Thus, it leaves space for ensembling KPCA with different kernels for better performance.

Table 1 compares the average accuracies under different dimensions. From Table 1 we can find that A-KPCA outperforms PCA and all the five KPCAs on the averaged accuracies (see the bold). It is interesting to note, from Table 1, that no single KPCA always achieves the best performance (see underlined), which verifies our former claim again.

6 Conclusion

This paper proposes an algorithm for adaptive kernel principal component analysis, which is based on a new criterion enabling nonlinear feature extraction and unsupervised kernel selection be performed simultaneously. An iterative algorithm is proposed to optimize the criterion. Besides, a non-iterative approximate algorithm is presented to improve the efficiency. Experiments validate the superiority of the proposed algorithms.

In the reported experiments we only use Gaussian kernel as the base kernel for Eq. 6. We will investigate the performance of the proposed algorithms using different kinds of base kernels simultaneously in future work. Also, comparison with other kernel tuning algorithms will be carried out in the future. Moreover, it is possible to extend the proposed unsupervised kernel learning algorithms to other kernel-based methods, such as kernel k-means clustering.

Acknowledgments: Supported by the National Science Foundation of China (60505004, 60473035), the Jiangsu Science Foundation (BK2004001, BK2006521), and the National Science Fund for Distinguished Young Scholars of China (60325207).

References

[1] C. Blake, E. Keogh, and C. J. Merz. UCI repository of machine learning databases. [http://www.ics.uci.edu/∼mlearn/MLRepository.html], Department of Information and Computer Science, University of California, Irvine, CA, 1998.

[2] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 46(1):131–159, 2002.

[3] N. Cristianini, J. Shawe-Taylor, A. Elisseeff, and J. Kandola. On kernel-target alignment. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 367–373. MIT Press, Cambridge, MA, 2002.

[4] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. Wiley and Sons, New York, 1973.

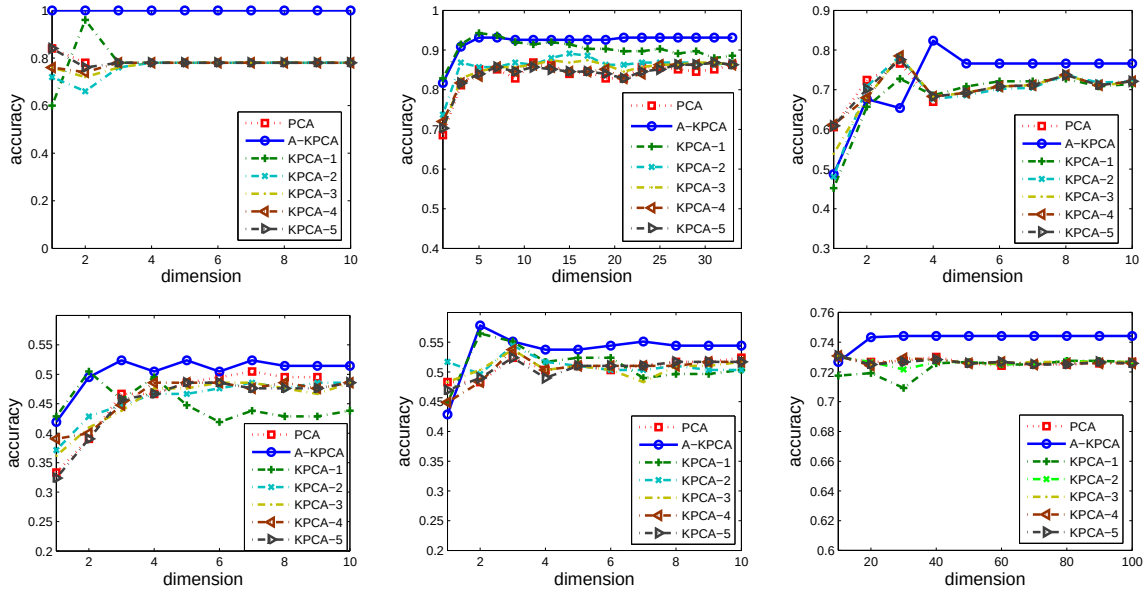

Figure 2. Comparisons on *Toy* (top left), *Ionosphere* (top middle), *Balance* (top right), *Glass* (bottom left), *Heart* (bottom middle) and *COIL-20* (bottom right).

Table 1. Comparisons of averaged accuracies (%) under different dimensions

Methods	Ionosphere	Balance	Glass	Heart	COIL-20
PCA	83.70	69.15	49.53	50.71	72.67
KPCA-1	90.29	68.24	44.86	50.76	72.00
KPCA-2	86.12	69.20	45.91	50.97	72.63
KPCA-3	85.04	69.65	45.62	51.18	72.70
KPCA-4	84.27	70.42	46.19	50.92	72.67
KPCA-5	84.03	70.51	45.24	50.86	72.66
A-KPCA	92.13	**72.37**	**50.38**	53.53	**74.24**
Approximate A-KPCA	**92.20**	72.27	49.24	**53.58**	74.22

[5] J. T. Kwok and I. W. Tsang. Learning with idealized kernels. In *Proceedings of the 20th International Conference on Machine Learning*, pages 400–407, Washington, DC, 2003.

[6] C. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine learning Research*, 5:27–72, 2004.

[7] K.-R. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf. An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks*, 12(2):181–201, 2001.

[8] S. A. Nene, S. K. Nayar, and H. Murase. Columbia object image library (COIL-20) [http://www1.cs.columbia.edu/cave/research/softlib/coil-20.html]. Technical Report CUCS-005-96, Columbia University, New York, NY, 1996.

[9] C. S. Ong, A. J. Smola, and R. C. Williamson. Hyperkernels. In S. T. S. Becker and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 478–485. MIT Press, Cambridge, MA, 2003.

[10] B. Schölkopf, A. J. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigen-value problem. *Neural Computation*, 10(5):1299–1319, 1998.

[11] I. W. Tsang and J. T. Kwok. Efficient hyperkernel learning using second-order cone programming. In *Proceedings of the 15th European Conference on Machine Learning*, pages 453–464, Pisa, Italy, 2004.

[12] S. Yang, S. Yan, D. Xu, X. Tang, and C. Zhang. Fisher + Kernel criterion for discriminant analysis. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Patter Recognition*, pages 197–202, San Diego, CA, 2005.

[13] J. Ye. Generalized low rank approximations of matrices. In *Proceedings of the 21st International Conference on Machine Learning*, pages 887–894, Banff, Alberta, Canada, 2004.

[14] D. Q. Zhang, S. C. Chen, and Z.-H. Zhou. Learning the kernel parameters in kernel minimum distance classifier. *Pattern Recognition*, 39(1):133–135, 2006.