Probabilistic Analysis of Kernel Principal Components

S. Kevin Zhou[1], **R. Chellappa**[2] **and B. Moghaddam**[3*]
[1,2]Center for Automation Research and ECE Department
University of Maryland, College Park, MD 20742
[3]Mitsubishi Electric Research Laboratory
201 Broadway, Cambridge, 02139

Abstract

This paper presents a probabilistic analysis of kernel principal components by unifying the theory of probabilistic principal component analysis and kernel principal component analysis. It is shown that, while the kernel component enhances the nonlinear modeling power, the probabilistic structure offers (i) a mixture model for nonlinear data structure containing nonlinear sub-structures, and (ii) an effective classification scheme. It turns out that the original loading matrix is replaced by a newly defined empirical loading matrix. The expectation/maximization algorithm for learning parameters of interest is also presented.

1 Introduction

Principal component analysis (PCA) [3] is one of the most popular statistical data analysis techniques and has been utilized in numerous areas such as data compression, image processing, computer vision, and pattern recognition. However, the PCA has two disadvantages: (i) it lacks a probabilistic model structure which is important in many contexts such as mixture modeling and Bayesian decision theory (also see [12]); and (ii) it restricts itself to a linear setting, where high-order statistical information is often discarded [10].

Probabilistic principal component analysis (PPCA) proposed by Tipping and Bishop [12, 13] overcomes the first disadvantage. By letting the noise component possess an isotropic structure, the PCA is implicitly embedded in a parameter learning stage for this model using the maximum likelihood estimation (MLE) method. An efficient expectation/maximization (EM) algorithm [1] is also developed to iteratively learn the parameters. Similar derivations are found in [6] in the context of decomposing the data space into a principal subspace and a complementary space with isotropic variance.

Kernel principal component analysis (KPCA) proposed by Schölkopf, Smola and Müller [10] overcomes the second disadvantage by using a 'kernel trick'. The essential idea of the KPCA is to avoid the direct evaluation of the required dot product in a high-dimensional feature space that is called as reproducing kernel Hilbert space (RKHS) using the kernel function. Hence, no explicit nonlinear mapping function projecting the data from the original space to the feature space is needed. Since a nonlinear function is used, albeit in an

* {shaohua,rama}@cfar.umd.edu, {baback}@merl.com.

implicit fashion, high-order statistical information is captured. See [7] for a recent survey on RKHS and its applications to discovering pre-image and denoised patterns in the original space.

We propose an approach to analyze kernel principal components in a probabilistic setting. It naturally combines PPCA and KPCA to overcome the disadvantages of PCA noted above. We call it the probabilistic kernel principal component analysis (PKPCA). In Section 2, we motivate the PKPCA approach by treating KPCA as a special case of PCA where the number of samples is smaller than the data dimension. One speciality of KPCA is the data centering issue, which is also taken into account in Section 2.

While the kernel part of PKPCA retains the nonlinear modeling power, which has been widely studied in the literature, the probabilistic structure of PKPCA offers additional advantages that are available from conventional probabilistic analysis. In this paper, we focus on the following two aspects:

1. A mixture modeling capacity of PKPCA. In Section 3, mixture of PKPCA is derived to model a nonlinear structure containing nonlinear substructures in a systematic way. Mixture of PKPCA nontrivially extends to the feature space induced by the kernel function, the theory of mixture of PPCA proposed by Tipping and Bishop [12, 13]. An EM algorithm [1] is developed to iteratively but efficiently learn the parameters of interest. We also show how to compute the Mahalanobis distance and study its limiting behavior.
2. An efficient classification scheme. Our analysis can be easily generalized for a classification task. As shown in Section 4, the performances are similar to those produced by mainstream kernel classifiers, such as support vector machine (SVM) and kernel Fisher discrimination (KFD) classifier.

2 Probabilistic kernel principal component analysis (PKPCA)

Suppose that $\{\mathsf{x}_1, \mathsf{x}_2, \dots, \mathsf{x}_N\}$ are the given training samples in the original data space $\mathcal{R}^d$. KPCA operates in a higher-dimensional feature space $\mathcal{R}^f$ induced by a nonlinear mapping function $\phi : \mathcal{R}^d \rightarrow \mathcal{R}^f$, where $f > d$ and f could even be infinite. The training samples in $\mathcal{R}^f$ are denoted by $\Phi_{f\times N} = [\phi_1, \phi_2, ..., \phi_N]$, where $\phi_n \doteq \phi(\mathsf{x}_n) \in \mathcal{R}^f$. The sample mean $\bar{\phi}_0$ and the covariance matrix $\mathsf{C}_{f\times f}$ in the feature space are given as

$$\bar{\phi}_0 \doteq N^{-1}\sum_{n=1}^{N}\phi(\mathsf{x}_n) = \Phi\mathsf{s}, \;\; \mathsf{C} \doteq N^{-1}\sum_{n=1}^{N}(\phi_n - \bar{\phi}_0)(\phi_n - \bar{\phi}_0)^{\mathsf{T}} = \Phi\mathsf{J}\mathsf{J}^{\mathsf{T}}\Phi^{\mathsf{T}}, \quad (1)$$

where the weight vector $\mathsf{s}_{N\times 1} = N^{-1}\mathsf{1}$ and the centering matrix $\mathsf{J} \doteq N^{-1/2}(\mathsf{I}_N - \mathsf{s}\mathsf{1}^{\mathsf{T}})$ with $\mathsf{1}$ being a vector of ones.

Denote the Gram matrix by K_0 and the centered Gram matrix by K:

$$\mathsf{K}_0 \doteq \Phi^{\mathsf{T}}\Phi, \;\; \mathsf{K} \doteq \mathsf{J}^{\mathsf{T}}\Phi^{\mathsf{T}}\Phi\mathsf{J} = \mathsf{J}^{\mathsf{T}}\mathsf{K}_0\mathsf{J}. \quad (2)$$

The Gram matrix can be evaluated using the 'kernel trick' to avoid the explicit knowledge of the nonlinear mapping function ϕ. Given a kernel function k satisfying

$$k(\mathsf{x}, \mathsf{y}) = \phi(\mathsf{x})^{\mathsf{T}}\phi(\mathsf{y}); \;\; \forall \mathsf{x}, \mathsf{y} \in \mathcal{R}^d, \quad (3)$$

the $(i, j)^{th}$ entry of the Gram matrix K_0 can be calculated as $\phi(\mathsf{x}_i)^{\mathsf{T}}\phi(\mathsf{x}_j) = k(\mathsf{x}_i, \mathsf{x}_j)$. The existence of such kernel functions is guaranteed by the Mercer's Theorem [9]. Examples include the RBF kernel, the polynomial kernel, etc. In this paper, we mainly use the RBF kernel defined as

$$k(\mathsf{x}, \mathsf{y}) = exp\{-||\mathsf{x} - \mathsf{y}||^2/(2\sigma^2)\}. \quad (4)$$

KPCA performs eigen-decomposition of the covariance matrix C in the feature space, typically retaining the top q eigenpairs. An eigenpair consists of an eigenvalues and its corresponding eigenvector. Such a decomposition can be obtained using the standard method reported in [4, 14]. Suppose that the top q eigenpairs for K are $\{(\lambda_n, \mathsf{v}_n)\}_{n=1}^{q}$, where λ_n's are sorted in a non-increasing order. In a matrix form, the eigenvectors of C are encoded in

$$\mathsf{U}_q \doteq [\mathsf{u}_1, ..., \mathsf{u}_q] = \Phi \mathsf{J} \mathsf{V}_q \Lambda_q^{-1/2}, \tag{5}$$

where $\mathsf{V}_q \doteq [\mathsf{v}_1, ..., \mathsf{v}_q]$ and $\Lambda_q \doteq \mathsf{D}[\lambda_1, ..., \lambda_q]$ is a diagonal matrix whose diagonal elements $\{\lambda_1, ..., \lambda_q\}$ are also the eigenvalues of the C matrix.

Theory of PKPCA

Probabilistic analysis of kernel principal components assumes that the data in the feature space follows a special factor analysis model [5] which relates an f-dimensional data $\phi(\mathsf{x})$ to a latent q-dimensional variable z as

$$\phi(\mathsf{x}) = \mu + \mathsf{W}\mathsf{z} + \epsilon, \tag{6}$$

where $\mathsf{z} \sim \mathsf{N}(0, \mathsf{I}_q)$, $\epsilon \sim \mathsf{N}(0, \rho \mathsf{I}_f)$, and W is a $f \times q$ *loading matrix*. Here ρ is a pre-specified small constant, playing a regularizing role. We will also study the limiting case with ρ approaching 0. Therefore, $\phi(\mathsf{x}) \sim \mathsf{N}(\mu, \Sigma)$, where $\Sigma = \mathsf{W}\mathsf{W}^{\mathsf{T}} + \rho \mathsf{I}_f$. Typically, we have $q << N << f$.

As shown in [12, 13], the maximum likelihood estimates (MLE's) for μ and W are given by

$$\hat{\mu} = \bar{\phi}_0 = N^{-1} \sum_{n=1}^{N} \phi(\mathsf{x}_n) = \Phi \mathsf{s}, \;\; \hat{\mathsf{W}} = \mathsf{U}_q (\Lambda_q - \rho \mathsf{I}_q)^{1/2} \mathsf{R}, \tag{7}$$

where R is any $q \times q$ orthogonal matrix and U_q and Λ_q contain top q eigenvectors and eigenvalues of the C matrix. It is in this sense that our probabilistic analysis coincides with the plain KPCA. From now on, we stop using the $(\hat{.})$ notation since we always use MLE and set $\mathsf{R} = \mathsf{I}_q$ without loss of generality.

Substituting (5) into (7), we obtain

$$\mathsf{W} = \Phi \mathsf{J} \mathsf{V}_q \Lambda_q^{-1/2} (\Lambda_q - \rho \mathsf{I}_q)^{1/2} \mathsf{R} = \Phi \mathsf{J} \mathsf{Q}, \tag{8}$$

where the $N \times q$ matrix Q is defined as

$$\mathsf{Q} \doteq \mathsf{V}_q (\mathsf{I}_q - \rho \Lambda_q^{-1})^{1/2} \mathsf{R}. \tag{9}$$

Equation (8) has a very important implication: *W lies in a linear subspace of* Φ. It is this property that makes all subsequent calculations possible by casting them into dot products. We name the Q matrix as the *empirical loading matrix* since this relates the loading matrix to the empirical data. Also since the matrix $(\mathsf{I}_q - \rho \Lambda_q^{-1})$ in (9) is diagonal, additional savings in computing its square root are realized.

Now, the covariance matrix is given by $\Sigma = \Phi \mathsf{J} \mathsf{Q} \mathsf{Q}^{\mathsf{T}} \mathsf{J}^{\mathsf{T}} \Phi^{\mathsf{T}} + \rho \mathsf{I}_f$. This offers a regularized approximation to the covariance matrix $\mathsf{C} = \Phi \mathsf{J} \mathsf{J}^{\mathsf{T}} \Phi^{\mathsf{T}}$. We note that Tipping [11] used a similar technique to approximate the covariance matrix C as $\Sigma = \Phi \mathsf{J} \mathsf{D} \mathsf{J}^{\mathsf{T}} \Phi^{\mathsf{T}} + \rho \mathsf{I}_f$, where D is a diagonal matrix with many diagonal entries being zero. This is not surprising as in our computation $\mathsf{D} = \mathsf{Q}\mathsf{Q}^{\mathsf{T}}$ is rank deficient. However, we do not enforce D to be a diagonal matrix.

It is easy to show that Σ^{-1} can be easily computed using the Woodbury formula [2] as

$$\Sigma^{-1} = \rho^{-1} (\mathsf{I}_f - \mathsf{W} \mathsf{M}^{-1} \mathsf{W}^{\mathsf{T}}) = \rho^{-1} \{ \mathsf{I}_f - \Phi \mathsf{J} \mathsf{Q} \mathsf{M}^{-1} \mathsf{Q}^{\mathsf{T}} \mathsf{J}^{\mathsf{T}} \Phi^{\mathsf{T}} \}. \tag{10}$$

where $\mathsf{M} \doteq \rho \mathsf{I}_q + \mathsf{W}^{\mathsf{T}} \mathsf{W} = \rho \mathsf{I}_q + \mathsf{Q}^{\mathsf{T}} \mathsf{K} \mathsf{Q}$.

Parameter learning using EM

The key for the approach developed above is (8) which relates W to Φ through a linear equation and the empirical loading matrix Q. This motivates us to use the EM learning algorithm to learn the Q matrix instead of the W matrix.

We now present the EM algorithm for learning the parameters Q in PKPCA. Assume that Q is the estimate before the iteration and $\check{\mathsf{Q}}$ is the updated estimate after the iteration.

$$\check{\mathsf{Q}} = \mathsf{K}\mathsf{Q}(\rho \mathsf{I}_q + \mathsf{M}^{-1}\mathsf{Q}^{\mathsf{T}}\mathsf{K}^2\mathsf{Q})^{-1}, \tag{11}$$

The EM algorithm presented above involves only inversions of $q \times q$ matrices and arrives at the same results (up to an orthogonal matrix R) as direct computation. However, in practice one may still use direct computation of complexity $O(N^3)$, since the complexity of computing K^2 is $O(N^3)$. If we pre-compute K^2, the complexity for each iteration reduces to $O(qN^2)$. Clearly, the overall computation complexity depends on the number of iterations needed for the desired accuracy and the ratio of N to q.

Mahalanobis distance

Given a vector $\mathsf{y} \in \mathcal{R}^d$, the Mahalanobis distance is defined as

$$\mathsf{L}(\mathsf{y}) \doteq (\phi(\mathsf{y}) - \bar{\phi}_0)^{\mathsf{T}}\Sigma^{-1}(\phi(\mathsf{y}) - \bar{\phi}_0) = \rho^{-1}\{g_{\mathsf{y}} - \mathsf{h}_{\mathsf{y}}^{\mathsf{T}}\mathsf{J}\mathsf{Q}\mathsf{M}^{-1}\mathsf{Q}^{\mathsf{T}}\mathsf{J}^{\mathsf{T}}\mathsf{h}_{\mathsf{y}}\}. \tag{12}$$

where $g_{\mathsf{y}} \doteq (\phi(\mathsf{y}) - \bar{\phi}_0)^{\mathsf{T}}(\phi(\mathsf{y}) - \bar{\phi}_0)$ is a scalar and $\mathsf{h}_{\mathsf{y}} \doteq \Phi^{\mathsf{T}}(\phi(\mathsf{y}) - \bar{\phi}_0)$ is a vector. Both can be calculated using the 'kernel trick'.

It is interesting to notice that there is a limiting behavior when ρ approaches zero by defining the 'limiting' Mahalanobis distance as

$$\vec{\mathsf{L}}(\mathsf{y}) \doteq \lim_{\rho \to 0} \rho \mathsf{L}(\mathsf{y}) = g_{\mathsf{y}} - \mathsf{h}_{\mathsf{y}}^{\mathsf{T}}\mathsf{J}\vec{\mathsf{Q}}\vec{\mathsf{M}}^{-1}\vec{\mathsf{Q}}^{\mathsf{T}}\mathsf{J}^{\mathsf{T}}\mathsf{h}_{\mathsf{y}}. \tag{13}$$

where $\vec{\mathsf{Q}} \doteq \lim_{\rho \to 0} \mathsf{Q}$ and $\vec{\mathsf{M}} \doteq \lim_{\rho \to 0} \mathsf{M}$.

The 'limiting' distance is very useful in the sense that it free us from specifying the ρ value. However, it is significantly different from the Mahalanobis distance directly computed from the q-dimensional kernel principal subspace that disregards the remaining dimensions. The 'limiting' distance measures the 'growth' speed of the Mahalanobis distance that is defined on the full space when the full space is reduced to the q-dimensional kernel principal subspace, i.e. ρ approaches zero. The only common thing about the 'limiting' distance and the Mahalanobis distance from the q-dimensional kernel principal subspace is that they are both related to the eigenvalues and eigenvectors of the q-dimensional kernel principal subspace.

Recap of PKPCA

So far, we have shown how to perform the probabilistic analysis of kernel principal components. Starting from the Gram matrix K_0 (rather the centered Gram matrix K), we can obtain the empirical loading matrix Q and then the loading matrix W. The remaining computations can be cast into dot products in a straightforward manner. The probabilistic analysis brings us a lot of advantages that are not available for the plain KPCA. We now proceed to illustrate two of the advantages, namely mixture modeling and classification, in Sections 3 and 4.

3 Mixture analysis of probabilistic kernel principal components

Mixture of PKPCA models the data in a high-dimensional feature space using a mixture of density with each mixture component being a PKPCA density associated with an empirical loading matrix Q_i which can be derived from corresponding s_i and J_i (as shown below). Mathematically,

$$\mathsf{p}(\phi(\mathsf{x})) = \sum_{i=1}^{I} m_i \mathsf{p}(\phi(\mathsf{x})|i) = \sum_{i=1}^{I} m_i \mathsf{N}(\phi(\mathsf{x}); \bar{\phi}_i, \Sigma_i), \tag{14}$$

where m_i's are mixing probabilities summing to 1, and $\mathsf{p}(\phi(\mathsf{x})|i) = \mathsf{N}(\bar{\phi}_i, \Sigma_i)$ is the PKPCA density for the i^{th} component defined as

$$\mathsf{N}(\phi(\mathsf{x}); \bar{\phi}_i, \Sigma_i) = \frac{(2\pi)^{-f/2}}{|\Sigma_i|^{1/2}} \exp\{-\frac{\mathsf{L}_i(\mathsf{x})}{2}\} = (2\pi\rho)^{-f/2} \exp\{-\frac{\hat{\mathsf{L}}_i(\mathsf{x})}{2}\} \tag{15}$$

where $\mathsf{L}_i(\mathsf{x})$ is the Mahalanobis distance as in (12) with all the parameters involved coming from the i^{th} component, and

$$\hat{\mathsf{L}}_i(\mathsf{x}) \doteq \mathsf{L}_i(\mathsf{x}) + \log(|\mathsf{M}_i|) + q_i \log(\rho^{-1}). \tag{16}$$

Parameter learning using EM

We invoke the MLE principle [1, 12] to estimate the parameters of interests, i.e., $\{m_i, \mathsf{Q}_i\}$'s from the training data since the log-likelihood involves summations within logarithms.

Assuming the availability of $\{m_i, \mathsf{Q}_i\}$'s as initial conditions, we attempt to obtain the updated parameters $\{\check{m}_i, \check{\mathsf{Q}}_i\}$'s using the EM algorithm. We start from computing the posterior responsibility r_{ni}.

$$r_{ni} \doteq \mathsf{p}(i|\phi_n) = \frac{m_i \mathsf{p}(\phi_n|i)}{\mathsf{p}(\phi_n)} = \frac{m_i \exp\{-\frac{1}{2}\hat{\mathsf{L}}_i(\mathsf{x})\}}{\sum_{j=1}^{I} m_j \exp\{-\frac{1}{2}\hat{\mathsf{L}}_j(\mathsf{x})\}}. \tag{17}$$

We note that there is no need to calculate r_{ni} by exactly following (17). One only needs to evaluate the numerator $m_i \exp\{-\frac{1}{2}\hat{\mathsf{L}}_i(\mathsf{x})\}$ and perform normalization to guarantee that $\sum_{i=1}^{I} r_{ni} = 1$. This results in computational savings.

The EM iterations compute the following quantities:

$$\check{m}_i = \frac{1}{N}\sum_{n=1}^{N} r_{ni}, \quad \bar{\phi}_i = \frac{\sum_{n=1}^{N} r_{ni}\phi_n}{\sum_{n=1}^{N} r_{ni}} = \sum_{n=1}^{N} s_{ni}\phi_n = \Phi \mathsf{s}_i, \tag{18}$$

where $\mathsf{s}_i = [s_{1i}, s_{2i}, \ldots, s_{Ni}]^{\mathsf{T}}$ with $s_{ni} \doteq r_{ni}/(\sum_{n=1}^{N} r_{ni})$.

It is easy to show that the local responsibility-weighted covariance matrix for component i, C_i, is obtained by

$$\mathsf{C}_i \doteq \sum_{n=1}^{N} s_{ni}(\phi_n - \bar{\phi}_i)(\phi_n - \bar{\phi}_i)^{\mathsf{T}} = \Phi \mathsf{J}_i \mathsf{J}_i^{\mathsf{T}} \Phi^{\mathsf{T}}, \ \ \mathsf{J}_i \doteq (\mathsf{I}_N - \mathsf{s}_i \mathsf{1}^{\mathsf{T}})\mathsf{D}[s_{1i}^{1/2}, s_{2i}^{1/2}, \ldots, s_{Ni}^{1/2}]. \tag{19}$$

Using $\mathsf{K}_i \doteq \mathsf{J}_i^{\mathsf{T}} \mathsf{K}_0 \mathsf{J}_i$, the updated $\check{\mathsf{Q}}_i$ can obtained using

$$\check{\mathsf{Q}}_i = \mathsf{V}_{q_i,i}(\mathsf{I}_{q_i} - \rho\Lambda_{q_i,i}^{-1})^{1/2}, \tag{20}$$

where $\Lambda_{q_i,i}$ and $\mathsf{V}_{q_i,i}$ are the top q_i eigenvalues and eigenvectors of K_i. Also, an EM algorithm for learning the Q_i matrix as shown in Section 2 can be used instead of direct computation.

The above derivations indicate that it is not necessary to start our EM iterations from initializing the parameters e.g. $\{m_i, \mathsf{Q}_i\}$'s; instead we can start from assigning the posterior responsibility $\{r_{ni}\}$'s. Once assigned, we follow equations (18) to (20) to compute updated $\{\check{m}_i, \check{\mathsf{Q}}_i\}$'s. The iterations are continued. This way we can easily incorporate any prior knowledge gained from clustering techniques such as the 'kernelized' version of the K-means algorithm [10], or other algorithms [8].

Why mixture of PKPCA?

It is well known [10, 8] that kernel embedding results in clustering capability. This raises the question whether PKPCA is sufficient to model a nonlinear structure with nonlinear substructures. We demonstrate the necessity of mixture of PKPCA using the following examples.

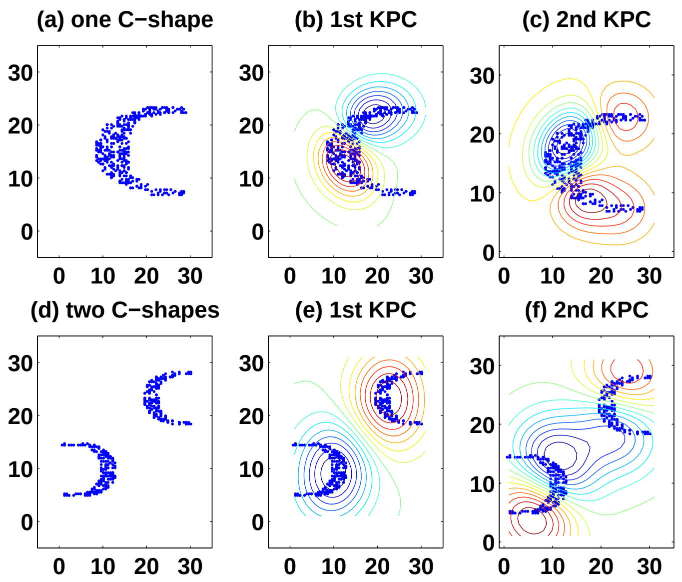
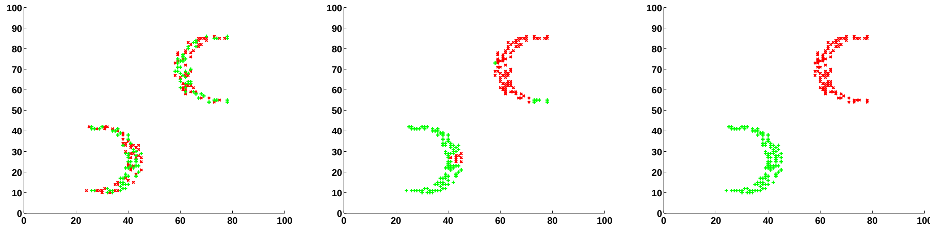


Figure 1: (a) One C-shape and contour plots of its (b) 1st and (c) 2nd KPCA features. (d) Two C-shapes and its contour plots of its (e) 1st and (f) 2nd KPCA features. The RBF kernel is used.

Fig. 1(a) gives a nonlinear structure containing a single C-shape and Figs. 1(b) and 1(c) the contour plots, for the 1st and 2nd kernel principal components, i.e. all points in the contour share the same principal component values. These plots capture the nonlinear shape very precisely. Now in Fig. 1(d), a nonlinear structure containing two C-shapes is presented. Figs. 1(e) and 1(f) displays the contour plots corresponding to 1(d). Clearly, they attempt to capture both C-shapes at the same time. This is not desirable. Ideally, we want to have two KPCAs, each modeling a different C-shape more precisely. This naturally leads us to mixture of PKPCA.

One may also ask: why not use the mixture of PPCA directly? Using mixture of PPCA is legitimate, but its use is not elegant in this scenario since one may need more than two components for Fig. 1(d) to capture the data structure due to the limitation of the linear setting in PCA. But mixture of PKPCA can directly model it using two components.

We now demonstrate how mixture of PKPCA performs using the two C-shapes shown in Fig. 1(d). We set the following parameters: the number of mixture components $I = 2$, the number of eigenpairs to keep for each mixture component $q_1 = q_2 = 2$, the regularizing constant $\rho = 0.01$, and use the RBF kernel with $\sigma = 8$. The algorithm is set to converge if the changes in the $\{r_{ni}\}$'s are small enough. Fig. 2(a) presents the initial configuration for the two C-shapes. We just generate random numbers for $\{r_{ni}\}$'s followed by a normalization step to guarantee $\sum_{i=1}^{I} r_{ni} = 1$. Fig. 2(b) shows the mixture assignment after the first iteration and Fig. 2(c) the final configuration (only after 3 iterations).



(a) (b) (c)

Figure 2: (a) Initial configuration. (b) After first iteration. (c) Final configuration. '+' and 'x' denote two different mixture components.

4 PKPCA Classifier

We now demonstrate the probabilistic interpretation embedded in PKPCA using a pattern classification problem. Suppose we have C classes to be classified. For class c, a PKPCA or mixture of PKPCA density $\mathsf{p}(\phi_c(\mathsf{x})|c)$ is trained; then, the class label for a point x is determined using the MAP principle by

$$\hat{c} = \arg \max_{c=1,\dots,C} \mathsf{p}(c)\mathsf{p}(\phi_c(\mathsf{x})|c), \tag{21}$$

where $\mathsf{p}(c)$ is the prior distribution, $\mathsf{p}(\phi_c(\mathsf{x}|c)$ is the conditional density for class c.

If PKPCA densities are learned for all the classes, (i.e., for class c, we learn $\{\mathsf{Q}_c\}$.) and the prior distribution $\mathsf{p}(c)$ is same for all C classes, the classifier reduces to the minimum distance classifier:

$$\hat{c} = \arg \min_{c=1,\dots,C} \hat{\mathsf{L}}_c(\mathsf{x}) \tag{22}$$

By taking the limit as $\rho \to 0$, it is easy to check that $\lim_{\rho\to 0} \rho\hat{\mathsf{L}}(\mathsf{x}) = \vec{\mathsf{L}}(\mathsf{x})$. Therefore, in practice, we use

$$\hat{c} = \arg \min_{c=1,\dots,C} \vec{\mathsf{L}}_c(\mathsf{x}) \tag{23}$$

in lieu of (22), which frees us from selecting the ρ value.

IDA Benchmark [1]

We tested our classifier on the IDA benchmark repository [7]. We used the cross-validation (the same procedure as in [7]) to choose our parameters such as the RBF kernel width σ, the number of eigenpairs to keep (i.e. the value of q) and invoked the PKPCA density without mixture modeling and the same kernel parameter for different classes. As shown in Table. 1, the PKPCA classifier produces good performances comparative to those of kernel classifiers such as SVM and KFD. Out of thirteen cases, PKPCA tops four of them. The standard deviations of the error rates are similar to those of SVM and KFD. Notice

[1] This is available at http://ida.first.gmd.de/ raetsch/data/benchmarks.htm.

that, unlike SVM and KFD, the PKPCA classifier can arrive at a decision boundary (in the induced RKHS) that is not necessarily a hyperplane. We are now investigating the classification scenarios where this property is advantageous. Another avenue to be explored is to use the mixture of PKPCA classifier.

	PKPCA	SVM	KFD
Banana	**10.5** $\pm$ 0.4	11.5 $\pm$ 0.7	10.8 $\pm$ 0.5
B. Cancer	28.0 $\pm$ 4.7	26.0 $\pm$ 4.7	**25.8** $\pm$ **4.6**
Diabetes	24.8 $\pm$ 1.9	23.5 $\pm$ 1.7	**23.2** $\pm$ **1.6**
German	24.9 $\pm$ 2.2	**23.6** $\pm$ **2.1**	23.7 $\pm$ 2.2
Heart	16.8 $\pm$ 3.4	**16.0** $\pm$ **3.3**	16.1 $\pm$ 3.4
Image	**2.8** $\pm$ **0.6**	3.0 $\pm$ 0.6	3.3 $\pm$ 0.6
Ringnorm	1.6 $\pm$ 0.1	1.7 $\pm$ 0.1	**1.5** $\pm$ **0.1**
F. Solar	34.8 $\pm$ 1.9	**32.4** $\pm$ **1.8**	33.2 $\pm$ 1.7
Splice	12.2$\pm$ 0.8	10.9 $\pm$ 0.7	**10.5** $\pm$ **0.6**
Thyroid	**4.0** $\pm$ **2.0**	4.8 $\pm$ 2.2	4.2 $\pm$ 2.1
Titanic	22.6 $\pm$ 1.3	**22.4** $\pm$ **1.0**	23.2 $\pm$ 2.0
Twonorm	**2.6** $\pm$ **0.2**	3.0 $\pm$ 0.2	**2.6** $\pm$ **0.2**
Waveform	11.4$\pm$ 0.5	**9.9** $\pm$ **0.4**	**9.9** $\pm$ **0.4**

Table 1: Classification error on IDA benchmark repository. The SVM and KFD results are reported in [7].

5 Conclusions

We have presented a new approach to analyze the kernel principal components in a probabilistic manner. It has been shown that the empirical loading matrix takes the place of the original loading matrix in PKPCA. Therefore we are able to convert our computations (such as the Mahalanobis distance) using the dot products, which can be evaluated using the 'kernel trick'. We have also demonstrated that the probabilistic analysis enables a mixture modeling of PKPCAs and an effective classification scheme.

References

[1] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. Roy. Statist. Soc. B*, 1977.

[2] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, 1996.

[3] I. T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, New York, 2002.

[4] M. Kirby and L. Sirovich. Application of karhunen-loéve procedure of the characterization of human faces. *IEEE Trans. PAMI*, PAMI-12(1):103–108, 1990.

[5] K. V. Mardia, J. T. Kent, and J. M. Bibby. *Multivariate Analysis*. Academic Press, 1979.

[6] B. Moghaddam and A. Pentland. Probabilistic visual learning for object detection. *Proc. of ICCV*, 1995.

[7] K. R. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf. An introducation to kernel-based learning algorithms. *IEEE Trans. Neutral Networks*, 12(2):181–202, 2001.

[8] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: analysis and an algorithm. *NIPS*, 2002.

[9] B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

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

[11] M. Tipping. Sparse kernel prinicipal component analysis. *NIPS*, 2001.

[12] M. E. Tipping and C. M. Bishop. Mixtures of probabilistic principal component analysers. *Neural Computation*, 11(2):443–482, 1999.

[13] M. E. Tipping and C. M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society, Series B*, 61(3):611–622, 1999.

[14] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3:72–86, 1991.