Probabilistic analysis of kernel principal components: mixture modeling, and classification

Shaohua Zhou
Center for Automation Research
Department of Electrical and Computer Engineering
University of Maryland, College Park, MD 20742

Abstract

This paper presents a probabilistic approach to analyze kernel principal components by naturally combining in one treatment the theory of probabilistic principal component analysis and that of kernel principal component analysis. In this formulation, the kernel component enhances the nonlinear modeling power, while the probabilistic structure offers (i) a mixture model for nonlinear data structure containing nonlinear sub-structures, and (ii) an effective classification scheme. It also turns out that the original loading matrix is replaced by the newly defined empirical loading matrix. The expectation/maximization algorithm for learning parameters of interest is then developed. Computation of reconstruction error and Mahalanobis distance is also discussed. Finally, we apply this to a real application of face recognition.

Index Terms

Principal component analysis, kernel methods, probabilistic analysis, mixture model, pattern classification.

I. Introduction

Principal component analysis (PCA) [1] 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, to name a few. However, the PCA has two disadvantages: (i) it lacks a probabilistic

Supported partially by the DARPA/ONR grant N00014-00-1-0908.

model structure which is important in many contexts such as mixture modeling and bayesian decision (also see [2]); and (ii) it restricts itself to a linear setting, where high-order statistical information is discarded [3].

Probabilistic principal component analysis (PPCA) proposed by Tipping and Bishop [2], [4] 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 [6] is also developed to iteratively learn the parameters.

Kernel principal component analysis (KPCA) proposed by Schölkopf, Smola and Müller [3] overcomes the second disadvantage by using a 'kernel trick'. As shown in Sec. II, the essential idea of the KPCA is to avoid the direct evaluation of the required dot product in a high-dimensional feature space 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 implicit fashion, high-order statistical information is captured. See [5] for a recent survey on the kernel space and application on discovering pre-image and denoised pattern in the original space.

We propose an approach to analyze kernel principal components in a probabilistic manner. It naturally combines PPCA and KPCA in one treatment to overcome the both disadvantages of PCA. We call it the probabilistic kernel principal component analysis (PKPCA). In Sec. III, we present our development of the PKPCA approach by treating the 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 Sec. III.

While the kernel part retains the nonlinear modeling power, resulting in a smaller reconstruction error, the additional probabilistic structure offers us (i) a mixture modeling capacity of PKPCA, and (ii) an efficient classification scheme.

In Sec. IV, mixture of PKPCA is derived to model 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 [2], [4]. An EM algorithm [6] is also developed to iteratively but efficiently learn the parameters of interest. We also show how to compute two important quantities, namely the reconstruction error and the Mahalanobis distance.

Our analysis can be easily generalized for a classification task. As shown in Sec. V, our performances are similar to those produced by mainstream kernel classifiers, such as support vector machine (SVM) and kernel Fisher discrimination (KFD) classifier, but our analysis provides more regularized approximation to the data structure. We apply PKPCA to face recognition in Sec. VI and summarize our work in Sec. VII.

A. Two examples

Fig. 1 shows two examples of nonlinear data structures to be modeled. In Fig. 1(a) presents the first example: a C-shaped structure in the foreground. In context of data modeling, we consider only the foreground and assume a uniform distribution within the C-shaped region and zeros outside. Fig. 1(b) displays 200 sample points drawn from this density. In context of pattern classification, we consider both the foreground and the background and further assume that the background class possess a uniform outside the C-shaped region and zeros inside. Fig. 1(c) shows the samples for the background class.

Fig. 1(d) shows the second example where the foreground nonlinear data structure consists of two C-shaped substructures. Figs. 1(e) and 1(f) present the drawn samples for the foreground and background classes, respectively. We mainly use this example for mixture modeling.

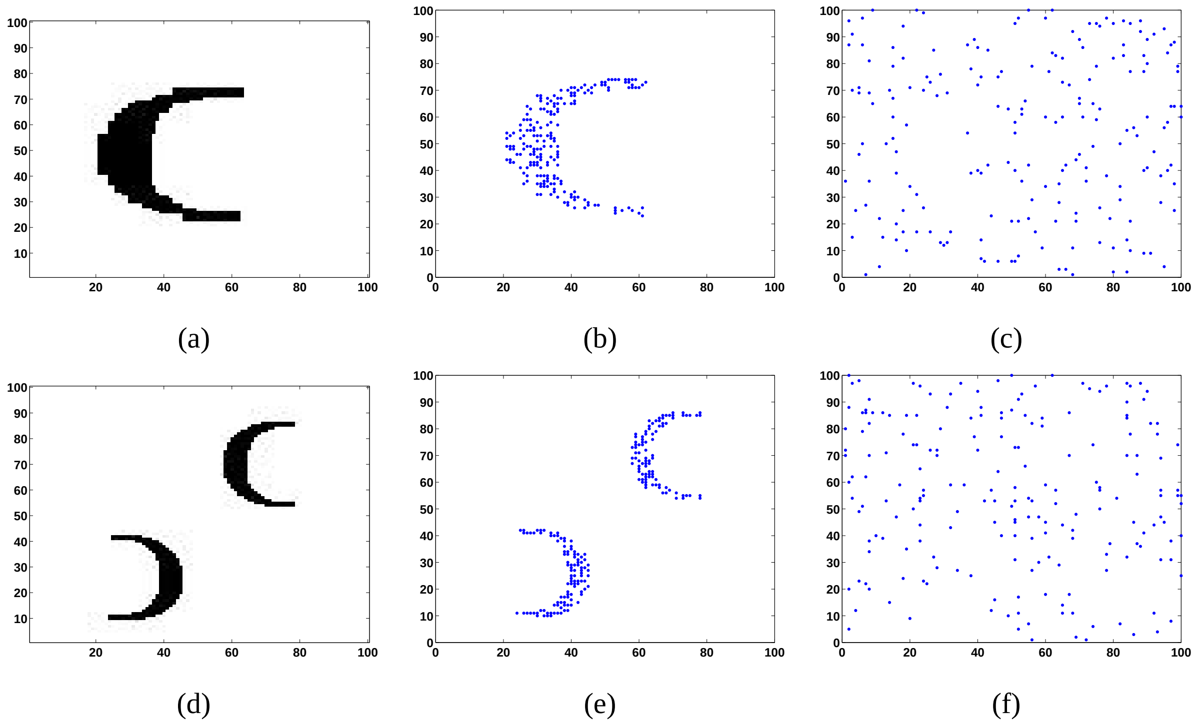


Fig. 1. Two nonlinear data structures (a)(d) and their drawn samples (of size 200) for the foreground class (b)(e) and the background (c)(f).

B. Notations

x is a scalar, x a vector, and X a matrix. X^T represents the matrix transpose, $\mathtt{tr}(\mathsf{X})$ the matrix trace, and $|\mathsf{X}|$ the matrix determinant. I_m denotes an $m \times m$ identity matrix. $\mathsf{1}$ denotes a vector or matrix of ones. $\mathsf{D}[a_1, a_2, \ldots, a_m]$ means an $m \times m$ diagonal matrix with diagonal elements $a_1, a_2, \ldots, a_m$. $\mathsf{p}(.)$ is a general probability density function. $\mathsf{N}(\mu, \Sigma)$ means normal density with mean μ and covariance matrix Σ.

II. Kernel principal component analysis

A. PCA in the feature space

Suppose that $\{\mathsf{x}_1, \mathsf{x}_2, \ldots, \mathsf{x}_N\}$ are 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 $\phi_n \doteq \phi(\mathsf{x}_n) \in \mathcal{R}^f$. Denote the sample mean in the feature space as

$$\bar{\phi}_0 \doteq \frac{1}{N} \sum_{n=1}^{N} \phi(\mathsf{x}_n) = \Phi\mathsf{s}, \tag{1}$$

where $\mathsf{s}_{N\times 1} = N^{-1}\mathsf{1}$.

The $f \times f$ covariance matrix in the feature space denoted by C is given as

$$\mathsf{C} \doteq \frac{1}{N} \sum_{n=1}^{N} (\phi_n - \bar{\phi}_0)(\phi_n - \bar{\phi}_0)^\mathsf{T} = \Phi\mathsf{J}\mathsf{J}^\mathsf{T}\Phi^\mathsf{T} = \Psi\Psi^\mathsf{T}, \tag{2}$$

where

$$\mathsf{J} \doteq N^{-1/2}(\mathsf{I}_N - \mathsf{s}\mathsf{1}^\mathsf{T}), \quad \Psi \doteq \Phi\mathsf{J}. \tag{3}$$

KPCA performs eigen-decomposition of the covariance matrix C in the feature space. Due to the high dimensionality of the feature space, we commonly possess insufficient number of samples, i.e., the rank of the C matrix is maximally N instead of f. However, computing eigensystem is still possible using the method presented in [7], [8]. Before that, we first show how to avoid the explicit knowledge of the nonlinear feature mapping.

B. Kernel trick

Define

$$\mathsf{K} \doteq \Psi^\mathsf{T}\Psi = \mathsf{J}^\mathsf{T}\Phi^\mathsf{T}\Phi\mathsf{J} = \mathsf{J}^\mathsf{T}\mathsf{K}_0\mathsf{J}, \tag{4}$$

where $\mathsf{K}_0 \doteq \Phi^{\mathsf{T}}\Phi$ is the grand matrix or the dot product matrix and can be evaluated using the 'kernel trick'; thus the explicit knowledge of the mapping function ϕ is avoided. Given a kernel function k satisfying

$$k(\mathsf{x},\mathsf{y}) = \phi(\mathsf{x})^{\mathsf{T}}\phi(\mathsf{y}); \quad \forall \mathsf{x},\mathsf{y} \in \mathcal{R}^d, \tag{5}$$

the $(i,j)^{th}$ entry of the grand matrix K_0 can be calculated as follows:

$$\mathsf{K}_0^{ij} = \phi(\mathsf{x}_i)^{\mathsf{T}}\phi(\mathsf{x}_j) = k(\mathsf{x}_i,\mathsf{x}_j). \tag{6}$$

The existence of such kernel functions is guaranteed by the Mercer's Theorem [9]. One example is the Gaussian kernel (or the RBF kernel) which is widely studied in the literature and the focus of this paper. It is defined as

$$k(\mathsf{x},\mathsf{y}) = \exp(-\frac{\|\mathsf{x}-\mathsf{y}\|^2}{2\sigma^2}) \quad \forall \mathsf{x},\mathsf{y} \in \mathcal{R}^d, \tag{7}$$

where σ controls the kernel width. In this case we have $f = \infty$. How to select the kernel width is rather ad hoc. We address this issue in the appendix I.

The use of the 'kernel trick' (or kernel embedding) captures high-order statistical information since the ϕ function coming from the nonlinear kernel function is nonlinear. We also note that, as long as the computations of interest can be casted in terms of dot products, we can safely use the 'kernel trick' to embed our operations into the feature space. This is the essence of all kernel methods, such as support vector machine (SVM) [10], kernel Fisher discriminant analysis (KFDA) [11], [12], and kernel independent component analysis (KICA) [13], including this work.

C. Computing eigensystem for the C *matrix*

As shown in [7], [8], the eigensystem for C can be derived from K. Suppose that the eigenpairs for K are $\{(\lambda_n, \mathsf{v}_n)\}_{n=1}^N$, where λ_n's are sorted in a non-increasing order. We now have

$$\mathsf{K}\mathsf{v}_n = \Psi^{\mathsf{T}}\Psi\mathsf{v}_n = \lambda_n\mathsf{v}_n; \quad n = 1,...,N. \tag{8}$$

Pre-multiplying (8) by Ψ gives rises to

$$\Psi\Psi^{\mathsf{T}}(\Psi\mathsf{v}_n) = \mathsf{C}(\Psi\mathsf{v}_n) = \lambda_n(\Psi\mathsf{v}_n); \quad n = 1,...,N. \tag{9}$$

Hence λ_n is the desired eigenvalue of C, with its corresponding eigenvector $\Psi\mathsf{v}_n$. To get the normalized eigenvector u_n for C, we only need to normalize $\Psi\mathsf{v}_n$.

$$(\Psi\mathsf{v}_n)^{\mathsf{T}}(\Psi\mathsf{v}_n) = \mathsf{v}_n^{\mathsf{T}}\Psi^{\mathsf{T}}\Psi\mathsf{v}_n = \mathsf{v}_n^{\mathsf{T}}\lambda_n\mathsf{v}_n = \lambda_n. \tag{10}$$

So,

$$\mathsf{u}_n = (\lambda_n)^{-1/2}\Psi\mathsf{v}_n, \quad n = 1, ..., N, \tag{11}$$

In a matrix form (if only top q eigenvectors are retained),

$$\mathsf{U}_q \doteq [\mathsf{u}_1, ..., \mathsf{u}_q] = \Psi\mathsf{V}_q\Lambda_q^{-1/2}, \tag{12}$$

where $\mathsf{V}_q \doteq [\mathsf{v}_1, ..., \mathsf{v}_q]$ and $\Lambda_q \doteq \mathsf{D}[\lambda_1, ..., \lambda_q]$.

It is clear that we are not operating in the full feature space, but in a low-dimensional subspace of it, which is spanned by the training samples. It seems that the modeling capacity is limited by the subspace dimensionality, or by the number of the samples. In reality, it however turns out that even in this subspace the smallest eigenvalues are very close to zero, which means that the full feature space can be further captured by a subspace with an even-lower dimensionality. This motivates us the use of the latent model.

III. Probabilistic analysis of kernel principal components

A. Theory of PKPCA

Probabilistic analysis assumes that the data in the feature space follows a special factor analysis model [14] which relates an f-dimensional data $\phi(\mathsf{x})$ to a latent q-dimensional variable z as

$$\phi(\mathsf{x}) = \mu + \mathsf{W}\mathsf{z} + \epsilon, \tag{13}$$

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*. Therefore, $\phi(\mathsf{x}) \sim \mathsf{N}(\mu, \Sigma)$, where

$$\Sigma = \mathsf{W}\mathsf{W}^{\mathsf{T}} + \rho\mathsf{I}_f. \tag{14}$$

Typically, we have $q << N << f$.

As shown in [2], [4], the maximum likelihood estimates (MLE's) for μ and W are given by

$$\mu = \bar{\phi}_0 = \frac{1}{N}\sum_{n=1}^{N}\phi(\mathsf{x}_n) = \Phi\mathsf{s}, \tag{15}$$

$$\mathsf{W} = \mathsf{U}_q(\Lambda_q - \rho\mathsf{I}_q)^{1/2}\mathsf{R}, \tag{16}$$

where R is any $q \times q$ orthogonal matrix, and U_q and Λ_q contains top q eigenvectors and eigenvalues of the C matrix. It is in this sense that our probabilistic analysis coincides with the plain KPCA.

Substituting (12) into (16), we obtain the following:

$$\mathsf{W} = \Psi\mathsf{V}_q\Lambda_q^{-1/2}(\Lambda_q - \rho\mathsf{I}_q)^{1/2}\mathsf{R} = \Psi\mathsf{Q} = \Phi\mathsf{J}\mathsf{Q}, \tag{17}$$

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

$$\mathsf{Q} \doteq \mathsf{V}_q(\mathsf{I}_q - \rho\Lambda_q^{-1})^{1/2}\mathsf{R}. \tag{18}$$

Equation (17) has a very important implication: W lies in a linear subspace of Φ. We name the Q matrix as *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 (18) is diagonal, additional savings in computing its square root are realized.

The MLE for ρ [2], [4] is given as

$$\rho = \frac{1}{f-q}\{\mathtt{tr}(\mathsf{C}) - \mathtt{tr}(\Lambda_q)\}. \tag{19}$$

Assuming that the remaining eigenvalues are zero, (this is a reasonable assumption supported by empirical evidences when f is finite), it is approximated as

$$\rho \simeq \frac{1}{f-q}\{\mathtt{tr}(\mathsf{K}) - \mathtt{tr}(\Lambda_q)\}. \tag{20}$$

But when f is infinite, this is doubtful since this always gives $\rho = 0$. In such a case, there is no automatic way of learning this. One way is to set a manual choice as in [15]. Even a fixed ρ is used, the optimal estimate for W is still same as in (17). It is interesting to note that Moghaddam and Pentland [16] derived (19) in a different context by minimizing the Kullback-Leibler divergence distance [17].

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 $\mathsf{C} = \Phi\mathsf{J}\mathsf{J}^\mathsf{T}\Phi^\mathsf{T}$. It is interesting to note that Tipping [15] 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.

A useful matrix denoted by M, which can be thought as a 'reciprocal' matrix for Σ is defined as

$$\mathsf{M} \doteq \rho\mathsf{I}_q + \mathsf{W}^\mathsf{T}\mathsf{W} = \rho\mathsf{I}_q + \mathsf{Q}^\mathsf{T}\mathsf{K}\mathsf{Q}. \tag{21}$$

If (18) is substituted into (21), we have $\mathsf{M} = \mathsf{R}^\mathsf{T}\Lambda_q\mathsf{R}$ (see the appendix for details).

B. Parameter learning using EM

The key for the approach developed in Sec. III-A is (17) which relates W to Φ using 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 and ρ in PKPCA. Assume that Q and ρ are the estimates before iteration, and $\tilde{\mathsf{Q}}$ and $\tilde{\rho}$ are the updated estimates after iteration.

$$\tilde{\mathsf{Q}} = \mathsf{K}\mathsf{Q}(\rho\mathsf{I}_q + \mathsf{M}^{-1}\mathsf{Q}^\mathsf{T}\mathsf{K}^2\mathsf{Q})^{-1}, \tag{22}$$

$$\tilde{\rho} = \frac{1}{f}\mathsf{tr}(\mathsf{K} - \mathsf{K}\mathsf{Q}\mathsf{M}^{-1}\tilde{\mathsf{Q}}^{\mathsf{T}}\mathsf{K}). \tag{23}$$

As mentioned earlier, when f is infinite, using (23) is not appropriate and hence a manual choice of ρ is used instead. With ρ fixed, Q is nothing but the solution to (22) and one can check that Q given in (18) is the solution. The above EM algorithm 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 desired accuracy and the ratio of N to q.

C. *Reconstruction error and Mahalanobis distance*

Given a vector $\mathsf{y} \in \mathcal{R}^d$, we are often interested in computing the following two quantities: (i) the reconstruction error $\epsilon_\phi(\mathsf{y}) \doteq (\phi(\mathsf{y}) - \hat{\phi}(\mathsf{y}))^{\mathsf{T}}(\phi(\mathsf{y}) - \hat{\phi}(\mathsf{y}))$ where $\hat{\phi}(\mathsf{y})$ is the reconstructed version of $\phi(\mathsf{y})$; and (ii) the Mahalanobis distance $\mathsf{L}(\mathsf{y}) \doteq (\phi(\mathsf{y}) - \bar{\phi}_0)^{\mathsf{T}}\Sigma^{-1}(\phi(\mathsf{y}) - \bar{\phi}_0)$.

As shown in [2], the best predictor for $\phi(\mathsf{y})$ is

$$\hat{\phi}(\mathsf{y}) = \mathsf{W}(\mathsf{W}^{\mathsf{T}}\mathsf{W})^{-1}\mathsf{W}^{\mathsf{T}}(\phi(\mathsf{y}) - \bar{\phi}_0) + \bar{\phi}_0, \tag{24}$$

and $\phi(\mathsf{y}) - \hat{\phi}(\mathsf{y})$ is given by

$$\phi(\mathsf{y}) - \hat{\phi}(\mathsf{y}) = (\mathsf{I}_f - \mathsf{W}(\mathsf{W}^{\mathsf{T}}\mathsf{W})^{-1}\mathsf{W}^{\mathsf{T}})(\phi(\mathsf{y}) - \bar{\phi}_0) = \Pi(\phi(\mathsf{y}) - \phi_0), \tag{25}$$

where the $f \times f$ matrix $\Pi \doteq \mathsf{I}_f - \mathsf{W}(\mathsf{W}^{\mathsf{T}}\mathsf{W})^{-1}\mathsf{W}^{\mathsf{T}}$ is symmetric and *idempotent* as $\Pi^2 = \Pi$. So, $\epsilon_\phi(\mathsf{y})$ is computed as follows:

$$\epsilon_\phi(\mathsf{y}) = (\phi(\mathsf{y}) - \bar{\phi}_0)^{\mathsf{T}}\Pi(\phi(\mathsf{y}) - \bar{\phi}_0) = g_{\mathsf{y}} - \mathsf{h}_{\mathsf{y}}^{\mathsf{T}}\mathsf{J}\mathsf{Q}(\mathsf{Q}^{\mathsf{T}}\mathsf{K}\mathsf{Q})^{-1}\mathsf{Q}^{\mathsf{T}}\mathsf{J}^{\mathsf{T}}\mathsf{h}_{\mathsf{y}} \tag{26}$$

where g_{y} and h_{y} are defined by:

$$g_{\mathsf{y}} \doteq (\phi(\mathsf{y}) - \bar{\phi}_0)^{\mathsf{T}}(\phi(\mathsf{y}) - \bar{\phi}_0) = k(\mathsf{y}, \mathsf{y}) - 2k_{\mathsf{y}}^{\mathsf{T}}\mathsf{s} + \mathsf{s}^{\mathsf{T}}\mathsf{K}_0\mathsf{s}, \tag{27}$$

$$\mathsf{h}_{\mathsf{y}} \doteq \Phi^{\mathsf{T}}(\phi(\mathsf{y}) - \bar{\phi}_0) = k_{\mathsf{y}} - \mathsf{K}_0\mathsf{s}, \tag{28}$$

$$k_{\mathsf{y}} \doteq \Phi^{\mathsf{T}}\phi(\mathsf{y}) = [k(\mathsf{x}_1, \mathsf{y}), ..., k(\mathsf{x}_N, \mathsf{y})]^{\mathsf{T}}. \tag{29}$$

The Mahalanobis distance is calculated (detailed in the appendix) as follows:

$$\mathsf{L}(\mathsf{y}) = (\phi(\mathsf{y}) - \bar{\phi}_0)^{\mathsf{T}}\Sigma^{-1}(\phi(\mathsf{y}) - \bar{\phi}_0) = \rho^{-1}\{g_{\mathsf{y}} - \mathsf{h}_{\mathsf{y}}^{\mathsf{T}}\mathsf{J}\mathsf{Q}\mathsf{M}^{-1}\mathsf{Q}^{\mathsf{T}}\mathsf{J}^{\mathsf{T}}\mathsf{h}_{\mathsf{y}}\}. \tag{30}$$

Finally, an important observation is that as long as we can express $\bar{\phi}$ and C as in (1) and (2), i.e. there exist s and J that relate $\bar{\phi}_0$ and C to Φ, we can safely use the derivations in (4-30). This lays a solid foundation for the development of the mixture of PKPCA theory.

D. Experiments on kernel modeling

This section addresses the power of kernel modeling part in PKPCA in terms of the reconstruction error. The probabilistic nature of PKPCA will be illustrated in the next sections.

We compare PPCA and PKPCA since the only difference between them is the kernel modeling part. We define the reconstruction error percentage η as follows:

$$\eta(\mathsf{y}) = \frac{\epsilon(\mathsf{y})}{\mathsf{y}^{\mathsf{T}}\mathsf{y}}, \quad \eta_\phi(\mathsf{y}) = \frac{\epsilon_\phi(\mathsf{y})}{k(\mathsf{y},\mathsf{y})}, \tag{31}$$

where $\eta(\mathsf{y})$ is for PPCA and $\eta_\phi(\mathsf{y})$ for PKPCA.

Algorithm	PPCA $q=2$	PPCA $q=3$	PKPCA $q=9$	PKPCA $q=15$
Mean	8.23%	1.42%	3.88%	1.39%
Std. dev.	13.12%	4.52%	3.86%	1.39%

TABLE I

RECONSTRUCTION ERROR PERCENTAGE

Fig. 2 shows the histogram of η for the famous iris data [1]. This dataset consists of 150 samples and is used in pattern classification tasks. We, however, just treat it as a whole regardless of its class labels. Since it is just 4-d data, PPCA keeps at most 3 principal component, i.e. $q \leq 3$, while PKPCA has no such limit and can have $q \leq 149$. Fig. 2 and Table I show that PKPCA with $q = 9$, i.e. using 6% percent principal components produces a small η than PPCA with $q = 2$ that uses 50% components. In addition, PKPCA with $q = 15$ that uses 10% percent principal components produces a small η than PPCA with $q = 3$, using 75% components. A larger q produces even smaller η. This improvement benefits from the kernel modeling, which is able to capture the nonlinear structure of the data. However, PKPCA involves much more computation than PPCA.

IV. MIXTURE MODELING OF PROBABILISTIC KERNEL PRINCIPAL COMPONENTS

A. Theory of mixture of PKPCA

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

[1] This is available at the UCI Machine Learning Repository. The URL is http://www.ics.uci.edu/ mlearn/MLRepository.html.

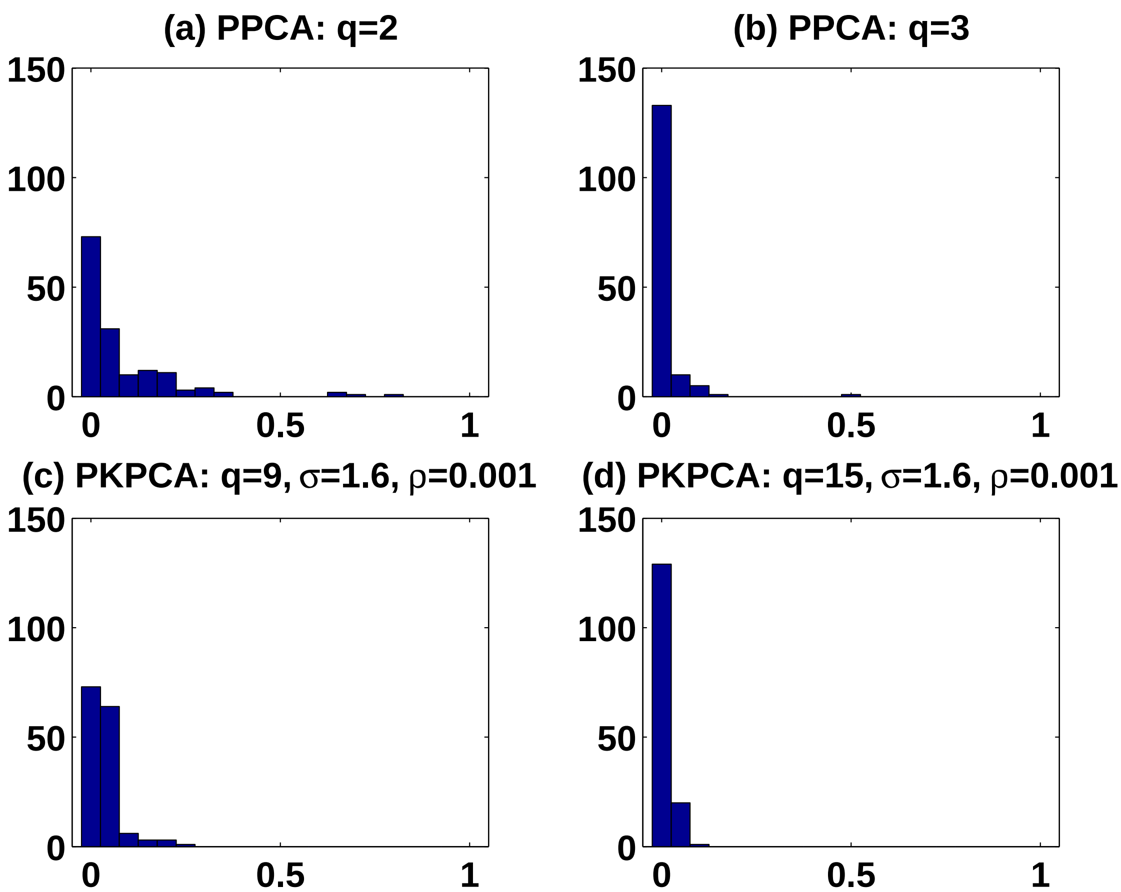


Fig. 2. Histogram of η for iris data obtained by (a) PPCA with $q = 2$, (b) PPCA with $q = 3$, (c) PKPCA with Gaussian kernel with $q = 9$, $\sigma = 2$ and $\rho = 0.001$, and (d) PKPCA with Gaussian kernel with $q = 15$, $\sigma = 2$ and $\rho = 0.001$.

and a ρ_i which can be derived from corresponding s_i and J_i (as shown below). Mathematically,

$$\mathsf{p}(\phi(\mathsf{x})) = \sum_{i=1}^{I} m_i \mathsf{p}(\phi(\mathsf{x})|i) = \sum_{i=1}^{I} m_i \mathsf{N}(\bar{\phi}_i, \Sigma_i), \tag{32}$$

where m_i's are mixing probabilities summing up 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

$$\mathsf{N}(\bar{\phi}_i, \Sigma_i) = \frac{(2\pi)^{-f/2}}{|\Sigma_i|^{1/2}} \exp\{-\frac{1}{2}\mathsf{L}_i(\mathsf{x})\} = \frac{(2\pi)^{-f/2}}{\rho_i^{(f-q_i)/2}|\mathsf{M}_i|^{1/2}} \exp\{-\frac{1}{2}\mathsf{L}_i(\mathsf{x})\} = (2\pi\rho_i)^{-f/2} \exp\{-\frac{1}{2}\hat{\mathsf{L}}_i(\mathsf{x})\} \tag{33}$$

where $\mathsf{L}_i(\mathsf{x})$ is the Mahalanobis distance as in (30) with all parameters involved coming from the i^{th} component, and

$$\hat{\mathsf{L}}_i(\mathsf{x}) \doteq \mathsf{L}_i(\mathsf{x}) + \log(|\mathsf{M}_i|) + q_i \log(\rho_i^{-1}). \tag{34}$$

B. Parameter learning using EM

We invoke the ML principle to estimate parameters of interests, i.e., $\{m_i, \mathsf{Q}_i, \rho_i\}$'s from the training data. It turns out that direct maximization is very difficult since the log-likelihood involves summations within logarithms. The EM algorithm [6], [2] is used instead.

Now, assuming the availability of $\{m_i, \mathsf{Q}_i, \rho_i\}$'s as initial conditions, we attempt to obtain updated parameters $\{\tilde{m}_i, \tilde{\mathsf{Q}}_i, \tilde{\rho}_i\}$'s using the EM algorithm.

We start from computing the posterior responsibility r_{ni}.

$$r_{ni} \doteq \mathsf{p}(i|\phi_n) = \frac{m_i \mathsf{p}(\phi_n|i)}{\mathsf{p}(\phi_n)} = \frac{m_i \rho_i^{-f/2} \exp\{-\frac{1}{2}\hat{\mathsf{L}}_i(\mathsf{x})\}}{\sum_{j=1}^{I} m_j \rho_j^{-f/2} \exp\{-\frac{1}{2}\hat{\mathsf{L}}_j(\mathsf{x})\}}. \tag{35}$$

The above equation can not be evaluated if we have no knowledge about f or even f is infinite. To avoid this, we manually set $\rho_i \equiv \rho$ so that we can cancel those terms involving f. From now on, we will assume this. Further cancellation can be reached by assuming $q_i \equiv q$, i.e., the third term in the right-hand side of (34) can be ignored. However, the use of this assumption largely depends on the nonlinear data structure. It is desirable, though difficult, that the number q_i is determined by the nonlinear complexity of the i^{th} sub-structure.

So,

$$r_{ni} = \frac{m_i \exp\{-\frac{1}{2}\hat{\mathsf{L}}_i(\mathsf{x})\}}{\sum_{j=1}^{I} m_j \exp\{-\frac{1}{2}\hat{\mathsf{L}}_j(\mathsf{x})\}}. \tag{36}$$

For computational saving, there is no need to calculate r_{ni} by exactly following (36). 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 $\sum_{i=1}^{I} r_{ni} = 1$.

The EM iterations compute the following quantities:

$$\tilde{m}_i = \frac{1}{N}\sum_{n=1}^{N} r_{ni}, \quad \tilde{\rho}_i = \rho_i, \tag{37}$$

$$\bar{\phi}_i = \frac{\sum_{n=1}^{N} r_{ni}\phi_n}{\sum_{n=1}^{N} r_{ni}} = \sum_{n=1}^{N} s_{ni}\phi_n = \Phi \mathsf{s}_i, \tag{38}$$

where $\mathsf{s}_i = [s_{1i}, s_{2i}, \ldots, s_{Ni}]^{\mathsf{T}}$ with

$$s_{ni} \doteq \frac{r_{ni}}{\sum_{n=1}^{N} r_{ni}}. \tag{39}$$

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

$$\mathsf{C}_i \doteq \sum_{n=1}^{N} s_{ni}(\phi_n - \bar{\phi}_i)(\phi_n - \bar{\phi}_i)^{\mathsf{T}} = \Phi \mathsf{J}_i \mathsf{J}_i^{\mathsf{T}} \Phi^{\mathsf{T}} = \Psi_i \Psi_i^{\mathsf{T}}, \tag{40}$$

where $\Psi_i = \Phi \mathsf{J}_i$, and

$$\mathsf{J}_i \doteq (\mathsf{I}_N - \mathsf{s}_i \mathsf{1}^{\mathsf{T}})\mathsf{D}[s_{1i}^{1/2}, s_{2i}^{1/2}, \ldots, s_{Ni}^{1/2}]. \tag{41}$$

Using

$$\mathsf{K}_i = \mathsf{J}_i^{\mathsf{T}} \mathsf{K}_0 \mathsf{J}_i, \tag{42}$$

and with ρ_i fixed as ρ, the updated $\tilde{\mathsf{Q}}_i$ can obtained using

$$\tilde{\mathsf{Q}}_i = \mathsf{V}_{q_i,i}(\mathsf{I}_{q_i} - \rho \Lambda_{q_i,i}^{-1})^{1/2}, \tag{43}$$

where $\Lambda_{q_i,i}$ and $\mathsf{V}_{q_i,i}$ are top q_i eigenvalues and eigenvectors of K_i. Also, an EM algorithm for learning the Q_i matrix as shown in Sec. III-B 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. $\{m_i, \mathsf{Q}_i, \rho_i\}$'s; instead we can start from assigning the posterior responsibility $\{r_{ni}\}$'s. Once assigned, we follow equations (37) to (43) to compute updated $\{\tilde{m}_i, \tilde{\mathsf{Q}}_i, \tilde{\rho}_i\}$'s. The iterations then move on. This way we can easily incorporate any prior knowledge gained from clustering techniques such as the 'kernelized' version of the K-means algorithm [3], or other algorithms [18].

C. *Why mixture of PKPCA?*

It is well known [3], [18] that kernel embedding results in clustering capability. This raises the doubt whether PKPCA is sufficient to model a nonlinear structure with nonlinear substructures. We demonstrate the necessity of mixture of PKPCA with the following examples.

Fig. 3(a) gives a nonlinear structure containing a single C-shape and Figs. 3(b) and 3(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 captures the nonlinear shape very precisely. Now in Fig. 3(d), a nonlinear structure containing two C-shapes is presented. Figs. 3(e) and 3(f) displays the contour plots corresponding to 3(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. However, plain KPCA has no such capability but PKPCA does. This naturally leads us to mixture of PKPCA. Sec. V also demonstrates this using classification results.

The successful kernel clustering algorithm [18] shows that after kernel embedding, the clusters become more separable. This further sheds light on the necessity of 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 2 components for Fig. 3(d) to capture the data structure due to the limitation of the linear setting in PCA. But mixture of PKPCA can neatly model it using two components.

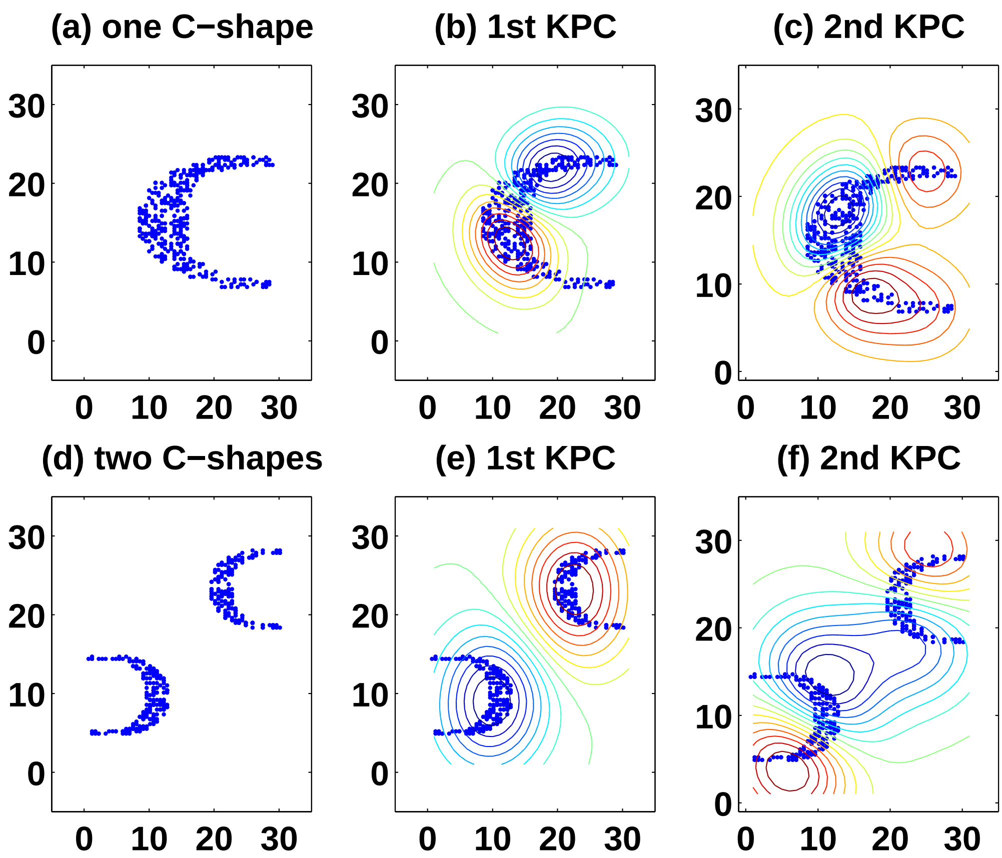
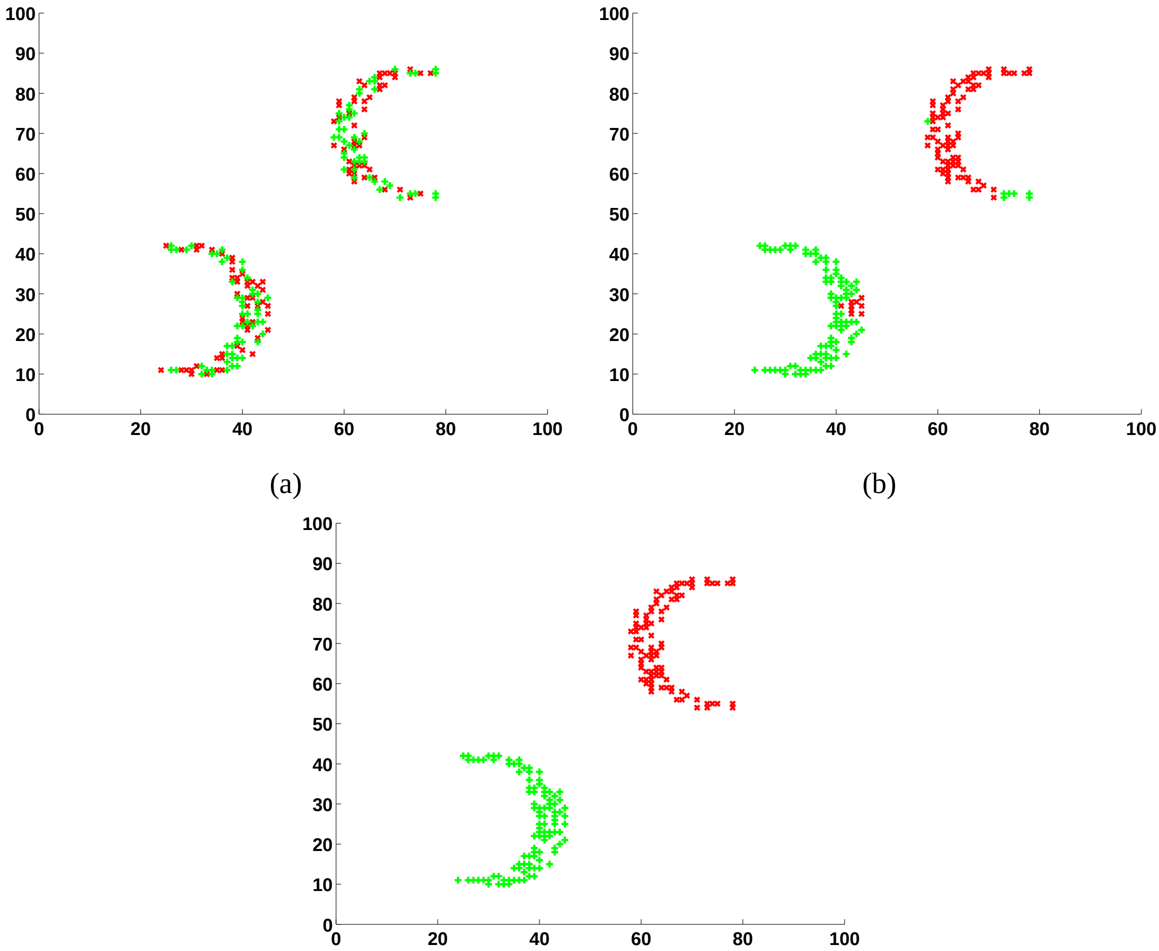


Fig. 3. (a) One C-shape and contour plots of its (b) 1st and (c) 2nd KPCA features. (d) Two C-shapes and its contour plots of its (e) 1st and (f) 2nd KPCA features.

D. Experiments

We now demonstrate how mixture of PKPCA performs by fitting it to the two C-shapes shown in Fig. 3(d). We set the following parameters: $I = 2$, $q = 2$, $\rho = 1e-2$, and $\sigma = 8$. The algorithm is set to converge if the changes in the $\{r_{ni}\}$'s are small enough.

Fig. 4(a) presents the initial configuration for the two C-shapes. We just generate random numbers for $\{r_{ni}\}$'s then normalization is done to guarantee $\sum_{i=1}^{I} r_{ni} = 1$. Fig. 4(b) shows the mixture assignment after the first iteration and Fig. 4(c) the final configuration (only after 3 iterations). A final note is that the EM algorithm can still converge to local minimum. In this case, the clustering method [18] is very helpful for initialization.



(a) (b)

(c)

Fig. 4. (a) Initial configuration. (b) After first iteration. (c) Final configuration. '+' and 'x' denote two different mixture components.

V. CLASSIFICATION

A. PKPCA or mixture of PKPCA classifier

We now demonstrate the probabilistic interpretation embedded in PKPCA using a patter 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 bayesian decision principle by

$$\hat{c} = \arg \max_{c=1,\dots,C} \mathsf{p}(c)\mathsf{p}(\mathsf{x}|c) = \arg \max_{c=1,\dots,C} \mathsf{p}(c)\mathsf{p}(\phi_c(\mathsf{x})|c)|\mathsf{J}_c(\mathsf{x})|, \tag{44}$$

where $\mathsf{p}(c)$ is the prior distribution, $\mathsf{p}(\mathsf{x}|c)$ is the conditional density for class c in the original space, and $\mathsf{J}_c(\mathsf{x})$ is the Jacobi matrix for class c.

To use (44), we are confronted by two dilemmas: (i) the Jacobi matrices, $\mathsf{J}_c(\mathsf{x})$'s, are unknown since we have no knowledge about $\phi_c(\mathsf{x})$; and (ii) the densities, $\mathsf{p}(\phi_c(\mathsf{x})|c)$'s, involves infinite f.

One trick to attack the first dilemma is to use same kernel function for all classes with the same kernel width σ, i.e. $\sigma_c = \sigma$. However, it might not be favorable since different classes possess different data structures. An alternative approach is that we still use different kernel functions for different classes but we approximate the Jacobi matrices. We use the following approximation:

$$|\mathsf{J}_c(\mathsf{x})| \simeq const, \quad \forall \mathsf{x}. \tag{45}$$

Fig. 5 demonstrates our approximation rationale. In Fig. 5(a) presents the contour plots for the true density to be modeled, which is uniform inside the black C-shaped region (Fig. 1(a)). All contours plots are located on the boundary. We fit PKPCA density ($\sigma = 15$, $q = 20$, and $\rho = 1e-6$) based on the samples shown in Fig. 1(b) and visualize the density using Fig. 5(b), which displays the map of $\log(\mathsf{L}(\mathsf{x}))$. To verify that the values in the C-shaped region are uniform, we show in Fig. 5(c) the contour plots for $\hat{\mathsf{L}}(\mathsf{x})$ inside the C-shaped region. Most contours are close the boundary, which indicates the uniformity of the density $\mathsf{p}(\phi(x))$ inside the C-shaped region and thus the Jacobi approximation which relates $\mathsf{p}(\phi(x))$ and $\mathsf{p}(\mathsf{x})$ is reasonable.

The above approximation leads to a linear decision rule. For example, in a two-class problem, the decision rule is, for some $\alpha > 0$,

$$\text{If} \quad \mathsf{p}(\phi_1(x)|1) \geq \alpha \ \mathsf{p}(\phi_2(x)|2) \quad \text{then class 1;} \quad \text{Else} \quad \text{class 2} \tag{46}$$

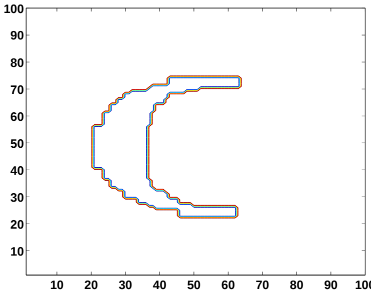
In the sequel, we simply take $\alpha = 1$.

To deal with the second dilemma, we have to cancel those untractable terms in $\mathsf{p}(\phi_c(\mathsf{x})|c)$ with f involved. Since $\mathsf{p}(\phi_c(\mathsf{x})|c)$ can be expressed as in (33) (except this is for the c^{th} class, not the i^{th} component), this cancellation can be done via the similar method presented in Sec. IV-B, i.e., assuming $\rho_c \equiv \rho$ for all classes.

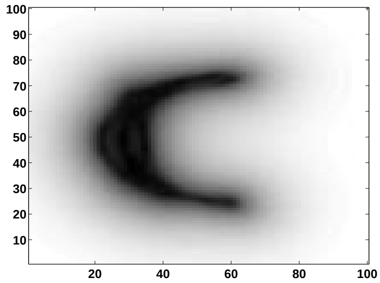
Putting the above measures together, we have the following decision rules:

- If PKPCA densities are learned for all classes, i.e., for class c, we learn $\{\rho_c = \rho, \mathsf{Q}_c\}$, it is easy to check that the classifier performs the following:

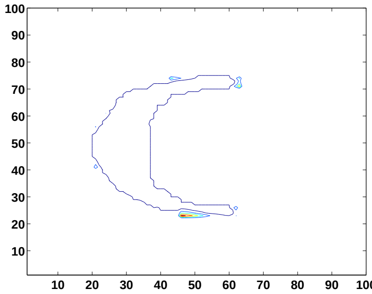
$$\hat{c} = \arg \min_{c=1,\ldots,C} \hat{\mathsf{L}}_c(\mathsf{x}). \tag{47}$$



(a)



(b)



(c)

Fig. 5. The approximation of the Jacobi matrix. (a) The contour plots of the true density: uniform inside the C-shaped region. (b) The map of $\log(\mathsf{L})$. (c) The contour plots of $\hat{\mathsf{L}}$ inside the C-shaped region.

- If mixture of PKPCA densities are learned for all classes, i.e., for the class c, we learn $\{\rho_c = \rho, m_{c,1}, \mathsf{Q}_{c,1}, ..., m_{c,I_c}, \mathsf{Q}_{c,I_c}\}$ with I_c being the number of mixture components, then the classifier decides as follows:

$$\hat{c} = \arg \max_{c=1,...,C} \sum_{j=1}^{I_c} m_{c,j} \exp\{-\frac{1}{2}\hat{\mathsf{L}}_{c,j}(\mathsf{x})\}. \tag{48}$$

Algorithm	Single C-shape	Single O-shape	Double C-shapes
PKPCA-d	1.57% $q = 30, \rho = 10^{-8}, \sigma_1 = 15, \sigma_2 = 35$	3.80% $q = 20, \rho = 10^{-6}, \sigma_1 = 15, \sigma_2 = 35$	7.49% $q = 20, \rho = 10^{-6}, \sigma_1 = 15, \sigma_2 = 35$
PKPCA-s	1.95% $q = 30, \rho = 10^{-8}, \sigma = 1$	5.50% $q = 30, \rho = 10^{-8}, \sigma = 1$	1.85% $q = 20, \rho = 10^{-6}, \sigma = 1$
SVM	1.80% $\sigma = 1$	5.45% $\sigma = 1$	1.69% $\sigma = 1$
KFDA	1.84% $\sigma = 1$, 30 components	5.47% $\sigma = 1$, 20 components	1.82% $\sigma = 1$, 20 components
mix. PKPCA	NA	NA	0.70% $q = 20, \rho = 10^{-6}, I_1 = 2, \sigma_1 = 8, I_2 = 1, \sigma_2 = 35$

TABLE II

CLASSIFICATION ERROR ON THE SINGLE C-SHAPED, THE SINGLE O-SHAPE, AND THE DOUBLE C-SHAPES.

B. Experiments

Synthetic Data: We design a 2-class problem with foreground (class 1) and background (class 2) classes given in Fig. 1(a), where letter 'C' means the foreground class. We then draw 200 samples for both classes as shown in Fig. 1 and Fig. 7. Fig. 6 presents the classification results obtained by the PKPCA classifier with different kernel widths for different classes (PKPCA-d), the PKPCA classifier with same kernel widths for different classes (PKPCA-s), the support vector machine (SVM) [10], and the kernel Fisher discriminant analysis (KFDA) [11]. In PKPCA-s, SVM and KFD, the kernel width σ is tuned (via exhaustive search from 1 to 100) to yield the best empirical classification results and reported in Table II. The PKPCA-d parameters actually used are also reported in Table II, where the kernel widths for the background and foreground classes are found via the procedures described in the appendix I. As shown in Fig. 6, the classification boundary obtained by PKPCA-d is very smooth and very similar to the original boundary, while those of PKPCA-s, SVM and KFDA seem to only replicate the training samples, with holes and gaps. Table II indicates that our PKPCA-d classifier outperforms the SVM and KFDA classifiers by some margin. Similar observations can be made based on the experimental results on a single O-shape as shown in Fig. 7.

The superior performance of PKPCA-d classifer mainly arises from its ability to model different classes with different kernel functions, while the PKPCA-s, SVM and KFDA employ only one kernel. This is a big advantage since as seen in our synthetic examples we clearly need different kernel widths for the foreground and background classes. More importantly, PKPCA provides a regularized approximation to the data structure; thus its decision boundary is very smooth. Also, the probabilistic interpretation of PKPCA enables the PKPCA classifier to deal with a m-class problem as easily as KFDA, while the SVM is basically designed for a 2-class problem and extending it to m-class is not very straightforward.

We now illustrate the mixture of PKPCA classifier by applying it to the double C-shapes shown in Fig. 1(d). We fit the mixture of PKPCA density for the foreground class based on the samples shown in Fig. 1(e) and the PKPCA density for the background class based on the samples shown in Fig. 1(f). Fig. 6 and Table II present the classification results. Clearly mixture of PKPCA classifier produces the best performance in terms of both the classification error and the decision boundary.

One important observation is that the PKPCA classifier with different kernel widths performs worst. This is because that the selected kernel width attempts to cover both nonlinear substructures simultaneously, which actually over-smoothes each substructure (see Fig. 6(d)). Hence, caution should be taken when modeling mixture data via PKPCA densities with different kernel widths.

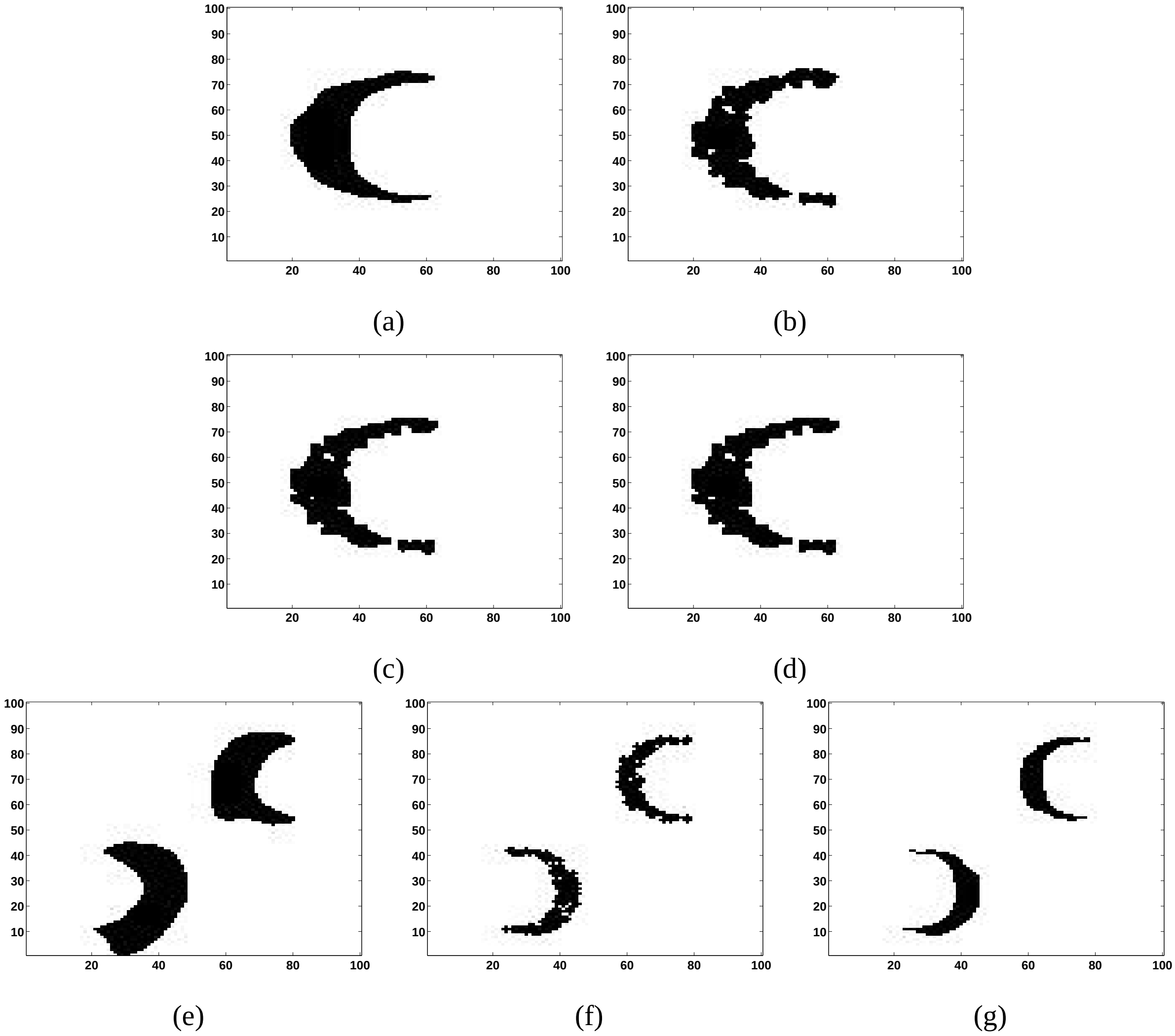


Fig. 6. The classification results on the single C-shape obtained by (a) PKPCA-d, (b) PKPCA-s, (c) SVM, and (d) KFDA and on the double C-shape obtained by (e) PKPCA-d classfier, (f) SVM, and (g) mixture of PKPCA classfier with different kernel widths.

IDA Benchmark [2]*:* We also test our classifier on the IDA benchmark repository [5]. To make our results comparable, we use the cross-validation (the same procedure as in [5]) to choose our parameters; also we invoke the PKPCA density without mixture modeling and the same kernel parameter for different classes. As tabulated in Table. III, our PKPCA classifier produce competitive performances to those of kernel classifiers such as SVM and KFD. We believe that the classification results can be improved by using PKPCA-d or even mixture of PKPCA classifier.

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

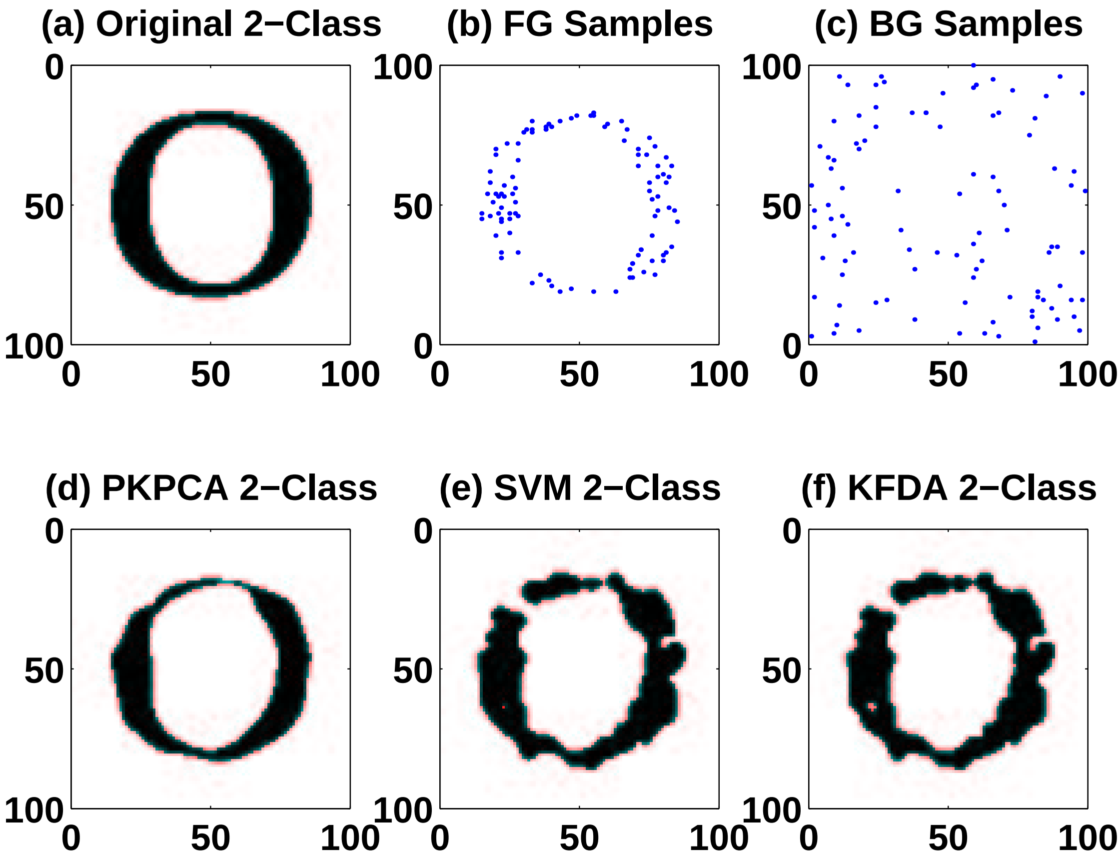


Fig. 7. The classification results on the single O-shape.

VI. A REAL APPLICATION: FACE RECOGNITION

We perform face recognition using a subset of the FERET database [19] with 200 subjects only. Each subject has 3 images: (i) one taken under controlled lighting condition with neutral expression; (ii) one taken under the same lighting condition as (i) but with different facial expressions (mostly smiling); and (iii) one taken under different lighting condition and mostly with a neutral expression. Fig. 8 shows some face examples in this database.

Our experiment focuses on testing the generalization capability of our algorithm. It is our hope that the training stage can learn the intrinsic characteristics of the space we are interested in. Therefore, we always keep the gallery and probe sets separate. We randomly select 300 images belonging to 100 subjects as the gallery set for learning and the remaining 300 images as the probe set for testing. This random division is repeated 20 times and we take their averages as the final result.

General component analysis is not geared towards discrimination, thus yielding inferior recognition

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

TABLE III

CLASSIFICATION ERROR ON IDA BENCHMARK REPOSITORY. THE SVM AND KFD RESULTS ARE REPORTED IN [5].

Fig. 8. Top row: neutral faces. Middle row: faces with facial expression. Bottom row: faces under different illumination. Image size is 24 by 21 in pixels.

results in practice. To this end, Moghaddam *et al.* [20], [21] introduce the concept of intra-personal space (IPS). The IPS is constructed by collecting all the difference images between any two image pairs belonging to the same individual. The construction of the IPS is meant to capture all the possible intra-personal variations introduced during the image acquisition.

Suppose that we have learned some density p_{IPS} on top of the IPS space and we are given the gallery set consisting of images $\{\mathsf{x}_1, \mathsf{x}_2, \ldots, \mathsf{x}_C\}$ for C different individuals. Given a probe image y, its identity

$\hat{c}$ is determined by

$$\hat{c} = \arg \max_{c=1,\dots,C} \mathsf{p}_{IPS}(\mathsf{y} - \mathsf{x}_c). \quad (49)$$

For comparison, we have implemented the following four methods. In PKPCA/IPS and PPCA/IPS, the IPS is constructed based on the gallery set and the PKPCA/PPCA density is fitted on top of that. In KPCA and PCA, all 300 training images are regarded lying in one face space (FS) and KPCA/PCA is then learned on that FS. The classifier sets the identity of a probe image as the identity of its nearest neighbor in the gallery set.

Table IV lists the recognition rate, averaging those of 20 simulations, using the top 1 match. The PKPCA/IPS algorithm attains the best performance since it combines the discriminative power of the IPS model and the merit of PKPCA. However, compared to PPCA/IPS, the improvement is not significant, indicating that second-order statistics might be enough after IPS modeling for the face recognition problem. However, PKPCA may be more effective since it also takes into account high-order statistics. Another observation is that variations in illumination are easier to model than facial expression using subspace methods.

	PKPCA/IPS	PPCA/IPS	KPCA	PCA
Expression	78.55%	78.35%	63.85%	67.65%
Illumination	83.9%	81.85%	51.9%	73.1%
Average	81.23%	80.1%	57.88%	70.38%

TABLE IV

RECOGNITION RATE

VII. CONCLUSIONS

We have presented a new approach to analyze the kernel principal components in a probabilistic manner. It has been discovered 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 reconstruction error and Mahalanobis distance) using the dot products, which can be evaluated using the kernel trick. We have also demonstrated that the probabilistic nature enables a mixture modeling of PKPCAs and an effective classification scheme.

VIII. ACKNOWLEDGEMENT

The author would like to thank Prof. Rama Chellappa, Prof. Eric V. Slud, and Dr. Baback Moghaddam for helpful discussions.

REFERENCES

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

[2] Michael E. Tipping and Christopher M. Bishop, "Mixtures of probabilistic principal component analysers," *Neural Computation*, vol. 11, no. 2, pp. 443–482, 1999.

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

[4] M. E. Tipping and C. M. Bishop, "Probabilistic principal component analysis," *Journal of the Royal Statistical Society, Series B*, vol. 61, no. 3, pp. 611–622, 1999.

[5] K. R. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf, "An introducation to kernel-based learning algorithms," *IEEE Trans. Neutral Networks*, vol. 12, no. 2, pp. 181–202, 2001.

[6] 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.

[7] M. Kirby and L. Sirovich, "Application of karhunen-loéve procedure of the characterization of human faces," *IEEE Trans. PAMI*, vol. PAMI-12, no. 1, pp. 103–108, 1990.

[8] M. Turk and A. Pentland, "Eigenfaces for recognition," *Journal of Cognitive Neutoscience*, vol. 3, pp. 72–86, 1991.

[9] J. Mercer, "Functions of positive and negative type and their connection with the thoery of integral equations," *Philos. Trans. Roy. Soc. London*, vol. A 209, pp. 415–446, 1909.

[10] V. N. Vapnik, *The Nature of Statistical Learning Theory*, Springer-Verlag, New York, ISBN 0-387-94559-8, 1995.

[11] S. Mika, G. Rätsch, J. Weston, B. Schölkopf, and K.-R. Müller, "Fisher discriminant analysis with kernels," in *Neural Networks for Signal Processing IX*, Y.-H. Hu, J. Larsen, E. Wilson, and S. Douglas, Eds. 1999, pp. 41–48, IEEE.

[12] G. Baudat and F. Anouar, "Generalized discriminant analysis using a kernel approach," *Neural Computation*, vol. 12, no. 10, pp. 2385–2404, 2000.

[13] F. Bach and M. I. Jordan, "Kernel independent component analysis," *Journal of Machine Learning Research*, vol. 3, pp. 1–48, 2002.

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

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

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

[17] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, Wiley, 1991.

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

[19] P. J. Philipps, H. Moon, S. Rivzi, and P. Ross, "The FERET evaluation methodology fro face-recognition algorithms," *IEEE Trans. PAMI*, vol. 22, pp. 1090–1104, 2000.

[20] B. Moghaddam, T. Jebara, and A. Pentland, "Bayesian modeling of facial similarity," *Advances in Neural Information Processing System*, 1998.

[21] B. Moghaddam, "Principal manifolds and probabilistic subspaces for visual recognition," *IEEE Trans. PAMI*, vol. 24, no. 6, pp. 780–788, 2002.

[22] C. K. I. Williams, “On a connection between kernel PCA and metric multidimensional scaling,” *NIPS*, 2001.

APPENDIX I
KERNEL SELECTION

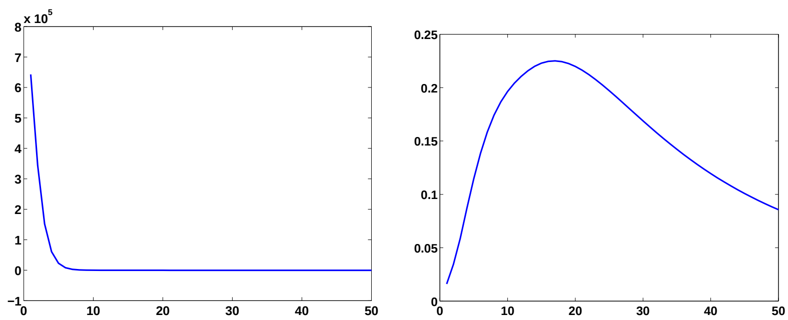
Only those functions satisfying the Mercer’s Theorem [9] can be used as kernel functions. In general, the kernel function lies in some parameterized function family. Denote the parameter of interest by θ. For example, θ can be the polynomial degree in the polynomial kernel, or the kernel width in the Gaussian kernel. The choice of θ remains an open question with the reason being that there is no systematic criteria to judge the goodness. Again, we only focus on the Gaussian kernel case; so $\theta = \sigma$ and $f = \infty$.

It seems that PKPCA offers us a systematic ML principle to follow, i.e., picking the σ which maximizes the likelihood or log-likelihood. However, it turns out that the ML principle fails as it has an inherent bias towards a large σ. The maxima of the log-likelihood $\mathcal{L}$ is given in (57) in the appendix. In case that $f = \infty$, we have to fix ρ due to the reasons mentioned in Sec. III-A, then we define the following quantity, as in (58) in the appendix II,

$$\mathcal{E}(\sigma) = -\frac{2}{N}\mathcal{L}_2 \propto \sum_{i=1}^{q} log(\lambda_i) + \frac{1}{\rho}(\mathtt{tr}(\mathsf{K}) - \mathtt{tr}(\Lambda_q)), \tag{50}$$

and the goal is to

$$\min_{\sigma} \quad \mathcal{E}(\sigma) \quad \boxed{\text{subject to}} \quad \lambda_q(\sigma) > \rho \;. \tag{51}$$

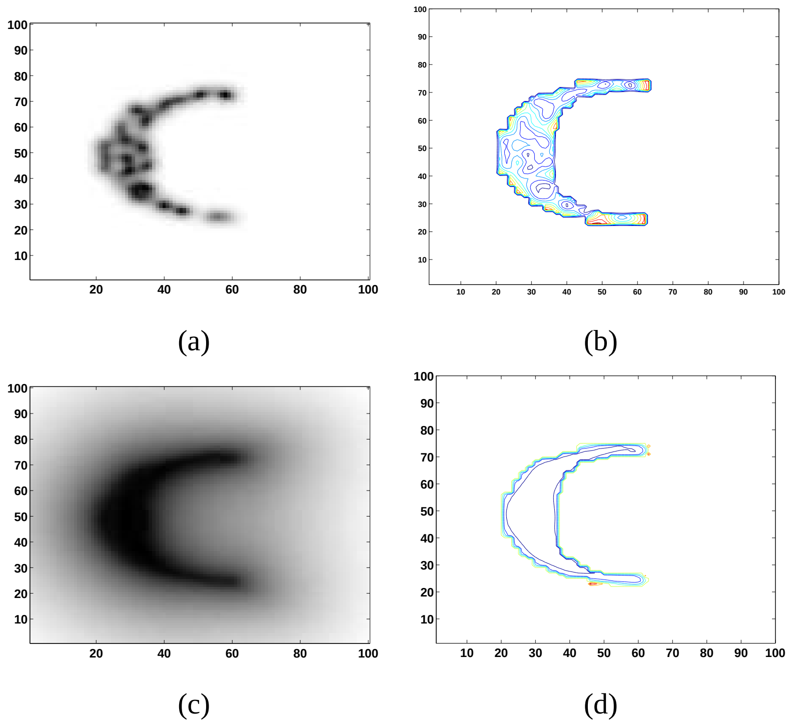


(a) (b)

Fig. 9. (a) The curve of $\mathcal{E}(\sigma)$. (d) The curve of $\lambda_1(\sigma)$. We have set $q = 30$ and $\rho = 1e^{-6}$.

We now show how it works. Fig. 9(a) presents the curve of $\mathcal{E}(\sigma)$ obtained using (50) for the C-shaped data (Fig. 1(a)), which always has a bias of favoring a large σ. This is not surprising since a large σ makes the matrix K_0 close to the matrix of ones; hence the matrix K close to the matrix of zeros, the data variation is reduced, and therefore the likelihood is increased. If σ goes to ∞, all data essentially reduces to one point in the feature space. This is also explained by Williams in [22]. Williams [22] has also studied the ratio of the sum of the top q eigenvalues to that of all eigenvalues, and discovered the same bias.

We propose an alternative approach by examining the first eigenvalue, which equals to the maximum variance of the projected data where the projection occurs in the feature space induced by the kernel function. Fig. 9(b) shows the plot of the first eigenvalue $\lambda_1(\sigma)$ against σ. There is a unique maximum. We pick this as our kernel width. This choice of the kernel width seems to have a close relationship with the assumption on the Jacobi matrix in (45). Fig. 10(a) present the map of $\log(\mathsf{L}(\mathsf{x}))$ for the single C-shape (with $\sigma = 3$) and Fig. 10(b) the contour plots of $\hat{\mathsf{L}}(\mathsf{x})$. The map is very granular and the uniformity inside the C-shaped region disappears. Fig. 10(c) shows the map of $\log(\mathsf{L}(\mathsf{x}))$ with $\sigma = 36$ and Fig. 10(d) the contour plots of $\hat{\mathsf{L}}(\mathsf{x})$. Now, the map is over-smoothed (compare the intensity change inside and outside the C-shaped region with that of Fig. 5(b)).



(a) (b)

(c) (d)

Fig. 10. (a) The map of $\log(\mathsf{L})$ and (b) the contour plots of $\hat{\mathsf{L}}$ inside the C-shaped region, when $\sigma = 3$. (c) The map of $\log(\mathsf{L})$ and (d) the contour plots of $\hat{\mathsf{L}}$ inside the C-shaped region, when $\sigma = 36$.

APPENDIX II
SOME USEFUL COMPUTATIONS

A. Related to M

Plugging in (18) into M, we have

$$\begin{aligned}
\mathsf{M} &= \rho\mathsf{I}_q + \mathsf{W}^\mathsf{T}\mathsf{W} = \rho\mathsf{I}_q + \mathsf{Q}^\mathsf{T}\mathsf{K}\mathsf{Q} \\
&= \rho\mathsf{I}_q + \mathsf{R}^\mathsf{T}(\mathsf{I}_q - \rho\Lambda_q^{-1})^{1/2}\mathsf{V}_q^\mathsf{T}\mathsf{K}\mathsf{V}_q(\mathsf{I}_q - \rho\Lambda_q^{-1})^{1/2}\mathsf{R} \\
&= \rho\mathsf{I}_q + \mathsf{R}^\mathsf{T}(\mathsf{I}_q - \rho\Lambda_q^{-1})^{1/2}\Lambda_q(\mathsf{I}_q - \rho\Lambda_q^{-1})^{1/2}\mathsf{R} \\
&= \rho\mathsf{I}_q + \mathsf{R}^\mathsf{T}(\Lambda_q - \rho\mathsf{I}_q)\mathsf{R} \\
&= \mathsf{R}^\mathsf{T}\Lambda_q\mathsf{R}.
\end{aligned} \tag{52}$$

Therefore,

$$|\mathsf{M}| = |\Lambda_q| = \prod_{i=1}^{q}\lambda_i, \quad \mathsf{M}^{-1} = \mathsf{R}^\mathsf{T}\Lambda_q^{-1}\mathsf{R}. \tag{53}$$

B. Related to Σ

Using the Sherman-Morrison-Woodbury matrix inversion identity, we have

$$\begin{aligned}
\Sigma^{-1} &= (\rho\mathsf{I}_f + \mathsf{W}\mathsf{W}^\mathsf{T})^{-1} = \rho^{-1}(\mathsf{I}_f - \mathsf{W}\mathsf{M}^{-1}\mathsf{W}^\mathsf{T}) \\
&= \rho^{-1}(\mathsf{I}_f - \Psi\mathsf{V}_q\Lambda_q^{-1}(\mathsf{I}_q - \rho\Lambda_q^{-1})\mathsf{V}_q^\mathsf{T}\Psi^\mathsf{T}).
\end{aligned} \tag{54}$$

$$\begin{aligned}
\mathtt{tr}(\Sigma^{-1}\mathsf{C}) &= \mathtt{tr}(\Sigma^{-1}\Psi\Psi^\mathsf{T}) = \mathtt{tr}(\Psi^\mathsf{T}\Sigma^{-1}\Psi) \\
&= \rho^{-1}(\mathtt{tr}(\mathsf{K}) - \mathtt{tr}(\mathsf{K}\mathsf{V}_q\Lambda_q^{-1}(\mathsf{I}_q - \rho\Lambda_q^{-1})\mathsf{V}_q^\mathsf{T}\mathsf{K}^\mathsf{T}) \\
&= \rho^{-1}(\mathtt{tr}(\mathsf{K}) - \mathtt{tr}(\mathsf{V}_q\Lambda_q\Lambda_q^{-1}(\mathsf{I}_q - \rho\Lambda_q^{-1})\mathsf{V}_q^\mathsf{T}\Lambda_q\mathsf{V}_q^\mathsf{T})) \\
&= \rho^{-1}(\mathtt{tr}(\mathsf{K}) - \mathtt{tr}(\mathsf{V}_q(\Lambda_q - \rho\mathsf{I}_q)\mathsf{V}_q^\mathsf{T})) \\
&= \rho^{-1}(\mathtt{tr}(\mathsf{K}) - \mathtt{tr}(\Lambda_q - \rho\mathsf{I}_q)) \\
&= \rho^{-1}(\mathtt{tr}(\mathsf{K}) - \sum_{i=1}^{q}\lambda_i) + q.
\end{aligned} \tag{55}$$

Also, the determinant of Σ is given by

$$|\Sigma| = \rho^{f-q}|\mathsf{M}| = \rho^{f-q}\prod_{i=1}^{q}\lambda_i. \tag{56}$$

C. Related to $\mathcal{L}$

$$\begin{aligned}
\mathcal{L} &= -\frac{Nf}{2}\log(2\pi) - \frac{N}{2}\log(|\Sigma|) \\
&\quad -\frac{1}{2}\sum_{n=1}^{N}(\phi(\mathsf{x}_n) - \bar{\phi}_0)^{\mathsf{T}}\Sigma^{-1}(\phi(\mathsf{x}) - \bar{\phi}_0) \\
&= -\frac{Nf}{2}\log(2\pi) - \frac{N(f-q)}{2}\log(\rho) - \frac{N}{2}\sum_{i=1}^{q}\log(\lambda_i) \\
&\quad -\frac{N}{2}\mathtt{tr}(\Sigma^{-1}\mathsf{C}) \\
&= -\frac{Nf}{2}\log(2\pi) - \frac{N(f-q)}{2}\log(\rho) - \frac{N}{2}\sum_{i=1}^{q}\log(\lambda_i) \\
&\quad -\frac{N}{2}\rho^{-1}(\mathtt{tr}(\mathsf{K}) - \sum_{i=1}^{q}\lambda_i) - \frac{Nq}{2} \qquad (57)
\end{aligned}$$

In case that f is infinite, we have to fix ρ. The quantity of interest becomes:

$$\mathcal{L}_2 = -\frac{N}{2}\rho^{-1}(\mathtt{tr}(\mathsf{K}) - \sum_{i=1}^{q}\lambda_i) - \frac{N}{2}\sum_{i=1}^{q}\log(\lambda_i). \qquad (58)$$