Kernel Pooled Local Subspaces for Classification

Peng Zhang, *Student Member, IEEE*, Jing Peng, *Member, IEEE*, and Carlotta Domeniconi

***Abstract*—We investigate the use of subspace analysis methods for learning low-dimensional representations for classification. We propose a kernel-pooled local discriminant subspace method and compare it against competing techniques: kernel principal component analysis (KPCA) and generalized discriminant analysis (GDA) in classification problems. We evaluate the classification performance of the nearest-neighbor rule with each subspace representation. The experimental results using several data sets demonstrate the effectiveness and performance superiority of the kernel-pooled subspace method over competing methods such as KPCA and GDA in some classification problems.**

***Index Terms*—Classification, Kernel machines, nearest neighbors, subspace analysis.**

I. Introduction

SUBSPACE analysis methods play an important role in pattern classification and computer vision research. For example, in visual learning and modeling, the principal modes are extracted and utilized for description, detection, and classification. Using these "principal modes" to represent data can be found in parametric descriptions of shape [1], target detection [2]–[4], visual learning [5], face recognition [4], [6], [7], linear discriminant analysis [8], and Fisherfaces [9].

Subspace analysis often significantly simplifies tasks such as regression and classification by computing low-dimensional subspaces having statistically uncorrelated or independent variables. Principal component analysis (PCA) [10] is a prime example that employs eigenvector-based techniques to reduce dimensionality and extract features. Independent component analysis (ICA) [11] is another technique that performs linear decomposition by computing statistically independent and non-Gaussian components and modeling the observed data as a linear mixture of (unknown) independent sources. Nonlinear PCA [12], [13], kernel principal component analysis (KPCA) [14], and generalized discriminant analysis (GDA) [15] extend these linear techniques in a nonlinear fashion.

While GDA and KPCA have shown promise in dimension reduction, they are entirely global techniques and, hence, may not be adequate for representing high-dimensional data exhibiting local variations. Here, we propose a subspace method for learning low-dimensional representations by pooling local dimension information. That is, we perform a global dimensionality reduction by pooling local discriminants in feature space and using the kernel trick [16] to capture nonlinearity. As a result, our method has the potential of achieving the best of both global and local dimension reduction techniques. The resulting curved subspace is discriminant and compact, whereby better classification performance and greater computational efficiency can be expected.

The rest of the paper is organized as follows. Section II discusses related work in subspace analysis methods. Section III describes local pooling for dimension reduction in the input space. After that, Section IV introduces the kernelized version of input space local pooling. Section VII shows experimental results evaluating the efficacy of the proposed method using a number of data sets. Finally, Section IX concludes the paper by highlighting the main contributions of the work.

Manuscript received August 2, 2003. This work was supported in part by the Army Research Office under Grant DAAD19-03-C-0111. This paper was recommended by Associate Editor B. Draper.

P. Zhang and J. Peng are with the Electrical Engineering and Computer Science Department, Tulane University, New Orleans, LA 70118 USA (e-mail: zhangp@eecs.tulane.edu; jp@eecs.tulane.edu).

C. Domeniconi is with ISE Department, George Mason University, Fairfax, VA 22030 USA (e-mail: carlotta@ise.gmu.edu).

Digital Object Identifier 10.1109/TSMCB.2005.846641

II. Subspace Methods

The objective of subspace analysis is to represent high-dimensional data in a low-dimensional subspace according to some optimality criteria. Classification then takes place on the chosen subspace. Here, we briefly describe several methods for computing both linear and nonlinear subspaces and highlight their corresponding characteristics. We assume that the data can be captured by a compact and connected subspace, which is often the case, for example, in face recognition [4], [6], [7], [9], [17].

A. Principal Component Analysis

In PCA [10], the basis vectors are obtained by solving the eigenvalue problem $\Lambda = Q^T \Sigma Q$, where Σ is the covariance matrix of the data, Q is the eigenvector matrix of Σ, and Λ is the corresponding diagonal matrix of eigenvalues. Matrix Q defines a transformation (rotation) that decorrelates the data and makes explicit the invariant subspace of the matrix "operator" Σ. Often in PCA, only the largest (or principal) eigenvectors are used to project the data. That is, $\mathbf{z} = Q_m^T \mathbf{x}$, where Q_m is a submatrix of Q representing the first m principal eigenvectors. PCA represents a global linear projection of the data onto a lower dimensional subspace corresponding to the maximal eigenvalues.

B. Kernel Principal Component Analysis

PCA is capable of discovering linear correlations among data. KPCA is a nonlinear generalization of PCA [14]. KPCA first applies a nonlinear mapping to the input $\mathbf{x}$, $\phi(\mathbf{x}) : \Re^d \rightarrow \Re^h$ and then solves for a linear PCA in the induced feature space $\Re^h$, where $h \gg d$ and possibly infinite. In KPCA, the mapping ϕ is made implicit by the use of kernel functions satisfying Mercer's theorem [18]

$$k(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j). \quad (1)$$

Since computing covariance involves only dot products, performing a PCA in the feature space can be formulated with kernels in the input space without the explicit (and possibly prohibitive) direct computation of ϕ. Assuming that the projection of the data in feature space is zero-mean, the covariance matrix in feature space is given by

$$\Sigma_K = E\left(\phi(\mathbf{x})\phi(\mathbf{x})^T\right). \tag{2}$$

Like linear PCA, we solve the eigenvector equation $\lambda\mathbf{v} = \Sigma_K\mathbf{v}$. Here, $\mathbf{v}$ is a vector of possibly infinite dimensions, and thus, it is not proper to solve it directly. Notice that since the solution $\mathbf{v}$ must lie in the span of the training data $\phi(\mathbf{x}_i)$, there must exist coefficients $\{w_i\}$ such that

$$\mathbf{v} = \sum_{i=1}^{l} w_i\phi(\mathbf{x}_i) \tag{3}$$

where l is the number of training samples. It must be also true for each training data

$$\lambda\left(\phi(\mathbf{x}_i)\cdot\mathbf{v}\right) = \left(\phi(\mathbf{x}_i)\cdot\Sigma_K\mathbf{v}\right), \quad \text{for} \quad i = 1,\ldots,l. \tag{4}$$

Substituting (2) and (3) into (4), we obtain the equavalent eigenvalue problem:

$$l\lambda\mathbf{w} = K\mathbf{w} \tag{5}$$

where K is the Gram matri. $K_{ij} := \phi(\mathbf{x}_i)\cdot\phi(\mathbf{x}_j)$, and $\mathbf{w} = (w_1,\ldots,w_l)^T$ is the vector of expansion coefficients of a given eigenvector $\mathbf{v}$, as defined in (3). Equation (5) is solved in the same way as in PCA.

Then, the KPCA principal components of any input data can be computed with kernel evaluations against the data set. For example, the qth principal component z_q of $\mathbf{x}$ is given by

$$z_q = (\mathbf{v}^q\cdot\phi(\mathbf{x})) = \sum_{i=1}^{l} w_i^q k(\mathbf{x},\mathbf{x}_i). \tag{6}$$

Similar to PCA, the eigenvectors $\mathbf{v}^q$ can be ranked by decreasing order of their eigenvalues λ_q. An m-dimensional projection of $\mathbf{x}$ is $\mathbf{z} = (z_1,\ldots,z_m)^T$, with individual components defined by (6).

A major advantage of KPCA over principal curves [19] is that KPCA does not require nonlinear optimization. On the other hand, selecting the optimal kernel (and its associated parameters) remains an engineering problem.

C. Linear Discriminant Analysis

Linear discriminant analysis (LDA) is a traditional statistical method that has been successfully used as a dimensionality reduction technique in many classification problems [20], [21]. The objective of LDA is to find a projection W that maximizes the ratio (Fisher's criterion)

$$F(W) = \frac{|W^T S_b W|}{|W^T S_w W|} \tag{7}$$

where S_b is the between-sum-of-squares matrix and S_w the within-sum-of-squares matrix. It turns out that it is equivalent to solving an eigenvalue problem. The columns of an optimal W are the generalized eigenvectors that correspond to the largest eigenvalues in

$$\lambda S_w\mathbf{w} = S_b\mathbf{w}. \tag{8}$$

D. Kernel Fisher Discriminant Analysis

Similar to KPCA, GDA [15] is a kernelized version of LDA. Nonlinear mapping $\phi(\mathbf{x})$ is used to replace $\mathbf{x}$. Let $\bar{\phi}_j$ be the mean of class j in feature space. For simplicity, we assume that each mapping $\phi(\mathbf{x}_i)$ has been centered in the feature space, that is, $\sum_{i=1}^{l}\phi(\mathbf{x}_i) = 0$.

For a given set of training data, the within-sum-of-squares matrix is

$$S_w = \frac{1}{l}\sum_{i=1}^{l}\phi(\mathbf{x}_i)\phi^T(\mathbf{x}_i). \tag{9}$$

The between-sum-of-squares matrix is

$$S_b = \frac{1}{l}\sum_{j=1}^{J} l_j\bar{\phi}_j\bar{\phi}_j^T \tag{10}$$

where J is the number of classes. Similar to LDA, we maximize the ratio

$$F(V) = \frac{|V^T S_b V|}{|V^T S_w V|} \tag{11}$$

which is equavalent to solving the eigenvector equation

$$\lambda S_w\mathbf{v} = S_b\mathbf{v}. \tag{12}$$

For the same arguments as in KPCA, there must exist coefficients $\{w_i\}$ such that

$$\mathbf{v} = \sum_{i=1}^{l} w_i\phi(\mathbf{x}_i). \tag{13}$$

Substituting (9), (10), and (13) into (12), we obtain a new eigenvector equation

$$\lambda M_w\mathbf{w} = M_b\mathbf{w} \tag{14}$$

where $\mathbf{w} = \{w_1,\ldots,w_l\}^T$ is the coefficient vector of $\mathbf{v}$, and M_w and M_b are matrices by some manipulation of Gram matrix K. The technique used here is somewhat involved, but it is similar to the technique used in our kernel local pooling method that is described in the latter part of this paper. Equation (14) is solved, and the eigenvectors that correspond to the largest eigenvalues are chosen. The projection of any input data can be computed with kernel evaluations against the data set. It follows the same way as in KPCA.

The major problem associated with LDA (or GDA) is that the within-sum-of-squares matrix is usually degenerated in practice. Often, this problem is solved by using techniques such as

pseudo inverse or PCA to remove the null space of the within-sum-of-squares matrix. However, it can be shown that the null space potentially contains significant discriminant information [22]. Recently, a direct version of LDA, known as DLDA, has been proposed [22], [23], and its nonlinear version using the kernel trick is described in [17].

E. Other Techniques

The Fukunaga–Koontz Transform [24] is a subspace method that has been successfully used in target and face detection. Huo *et al.* [25] recently showed that under certain conditions, the Fukunaga–Koontz Transform is the optimal low-rank approximation to a statistically optimal classifier.

It should be noted that LDA can have many criteria [21] other than Fisher's criterion (7) described above. For example, Biasmap, which is an LDA-related algorithm, has been proposed for multimedia Retrieval [26].

More recently, Fukumizu *et al.* [27] proposed a novel dimension-reduction approach. The idea is to find a low-dimensional "effective subspace" for $\mathbf{x}$, which retains the statistical relationship between $\mathbf{x}$ and its label y.

III. Pooled Local Subspace Method

Hastie and Tibshirani [28] proposed a global dimensionality-reduction technique by pooling local discriminant information. The technique computes a global subspace that summarizes local class centroid deviations. By doing so, it attempts to achieve the best of both global and local dimensionality reduction techniques. It produces a unique global subspace, which is highly desirable for interpretation and visualization purposes. It is also cost effective, because it can be computed offline. At the same time, it preserves local discriminant information, to the extent possible, which is critical to achieve accurate classification performance.

The idea is to compute a subspace corresponding to the eigenvectors of the average local between-sum-of-squares matrices. More specifically, for each training point, local pooling calculates the local centroid deviations $\tilde{\mathbf{x}}_j^{(i)} = \bar{\mathbf{x}}_j^{(i)} - \bar{\mathbf{x}}^{(i)}$, where $\bar{\mathbf{x}}_j^{(i)}$ denotes the mean of points from class j in a neighborhood of the ith training point, and $\bar{\mathbf{x}}^{(i)}$ denotes the overall mean. Then, it seeks a subspace that is close in average weighted squared distance to all these deviations. If U denotes an orthonormal basis for the subspace, this subspace can be computed by minimizing the total weighted residual sum of squares

$$\mathrm{RSS}(U) = \sum_{i=1}^{l}\sum_{j=1}^{J} \pi_j^{(i)} \left(\tilde{\mathbf{x}}_j^{(i)}\right)^T (I - UU^T)\tilde{\mathbf{x}}_j^{(i)} \tag{15}$$

where $\pi_j^{(i)}$ represents the local class membership proportions.

It turns out, as shown in [28], that this subspace is spanned by the largest eigenvectors of the average between-sum of squares matrix

$$B = \frac{1}{l}\sum_{i=1}^{l} B^{(i)} \tag{16}$$

where $B^{(i)}$ denotes the local between-sum-of-squares matrices at the ith training point

$$B^{(i)} = \sum_{j=1}^{J} \pi_j^{(i)} \left(\bar{\mathbf{x}}_j^{(i)} - \bar{\mathbf{x}}^{(i)}\right)\left(\bar{\mathbf{x}}_j^{(i)} - \bar{\mathbf{x}}^{(i)}\right)^T. \tag{17}$$

The experimental results presented in [28] show that the pooled local subspace method is very promising. Nevertheless, the approach has a major drawback. In high-dimensional input spaces where data become sparse, the estimation of local linear discriminant dimensions may be highly biased. This is because we are forced to look far away from the ith training point to find samples to compute the local between-sum-of-squares matrices. As such, we may violate local linear constraints. One way to compensate for the potential loss of linearity is to try to estimate curved local discriminant subspaces by using the kernel trick [16], thereby reducing bias.

It is important to note that local pooling does not sphere the data locally before calculating the centroid deviations. An argument given in [28] is that any local spherical window containing two classes will likely have a linear decision boundary orthogonal to the line joining the two means. As a result, local pooling will not suffer the small size sample problem (degenerate within class matrices) facing FDA or GDA [15], [22]. It is interesting to note that locally linear embedding also uses pooled locally linear constraints to compute a global subspace [29]. Table I shows the notations used in this paper.

IV. Kernel Pooled Local Subspace Method

We now show how to compute a nonlinear pooled local discriminant subspace by using the kernel trick [16]. Let $\phi : \mathbf{x} \rightarrow \phi(\mathbf{x})$ be the nonlinear mapping from $\Re^d$ to $\Re^h$. Consider, in the feature space, that a neighborhood of the ith point contains l_N points: $\mathbf{x}_1^{(i)}, \ldots \mathbf{x}_{l_N}^{(i)}$. Let $\bar{\phi}_j(\mathbf{x}^{(i)})$ be the mean of points from class j in the neighborhood: $\bar{\phi}_j(\mathbf{x}^{(i)}) = 1/l_j \sum_{y_k=j} \phi(\mathbf{x}_k^{(i)})$ (y_k is the class label of sample $x_k^{(i)}$), and let $\bar{\phi}(\mathbf{x}^{(i)})$ be the overall mean in the same neighborhood: $\bar{\phi}(\mathbf{x}^{(i)}) = 1/l_N \sum_{k=1}^{l_N} \phi(\mathbf{x}_k^{(i)})$. Here, l_j represents the number of points from class j in the neighborhood, and l_N represents the number (which is given) of all points in the neighborhood. In addition, we have

$$l_N = \sum_{j=1}^{J} l_j. \tag{18}$$

Then, the local between-sum-of-squares matrix at the ith training point in the feature space is

$$B_\phi^{(i)} = \sum_{j=1}^{J} \pi_j^{(i)} \left(\bar{\phi}_j\left(\mathbf{x}^{(i)}\right) - \bar{\phi}\left(\mathbf{x}^{(i)}\right)\right) \times \left(\bar{\phi}_j\left(\mathbf{x}^{(i)}\right) - \bar{\phi}\left(\mathbf{x}^{(i)}\right)\right)^T \tag{19}$$

$$= \tilde{\Phi}_J^{(i)} \left(\tilde{\Phi}_J^{(i)}\right)^T \tag{20}$$

TABLE I
NOTATION

$\mathbf{x}_i$	the observation vector of ith training data
y_i	the class label of the ith (training) data, $y_i \in \{1, ..., J\}$
d	dimension of input data $\mathbf{x}$
h	dimension of feature mapping $\phi(\mathbf{x})$
l	number of training data
J	number of classes
l_N	number of training points in a neighborhood
l_j	number of class j points (in a neighborhood)
B	between sum-of-squares matrix
W	within sum-of-squares matrix
$\mathbf{x}_k^{(i)}$	the kth point in a neighborhood of the ith training data
$\bar{\mathbf{x}}^{(i)}$	$= \frac{1}{l_N}\sum_k^{l_N} x_k^{(i)}$, the mean of points in a neighborhood of the ith training data
$\bar{\mathbf{x}}_j^{(i)}$	$= \frac{1}{l_j}\sum_{y_k=j} x_k^{(i)}$, the mean of points from class j in a neighborhood of the ith training data
ϕ	nonlinear mapping
$\phi(\mathbf{x}_i)$	the mapping of the ith training data
$\bar{\phi}(\mathbf{x}^{(i)})$	$= \frac{1}{l_N}\sum_k^{l_N} \phi(x_k^{(i)})$, the mean of points in a neighborhood of the ith training data in a feature space
$\bar{\phi}_j(\mathbf{x}^{(i)})$	$= \frac{1}{l_j}\sum_{y_k=j} \phi(x_k^{(i)})$, the mean of points from class j in a neighborhood of the ith training data in a feature space
Q^T	transpose of Q

where $\tilde{\Phi}_J^{(i)}$ is defined by

$$\tilde{\Phi}_J^{(i)} = \left[\sqrt{\hat{\pi}_1^{(i)}}\left(\bar{\phi}_1\left(\mathbf{x}^{(i)}\right) - \bar{\phi}\left(\mathbf{x}^{(i)}\right)\right) \sqrt{\hat{\pi}_2^{(i)}}\left(\bar{\phi}_2\left(\mathbf{x}^{(i)}\right) - \bar{\phi}\left(\mathbf{x}^{(i)}\right)\right) \cdots \sqrt{\hat{\pi}_J^{(i)}}\left(\bar{\phi}_J\left(\mathbf{x}^{(i)}\right) - \bar{\phi}\left(\mathbf{x}^{(i)}\right)\right)\right]. \quad (21)$$

The pooled local subspace method seeks a discriminant subspace that is close to all of $B^{(i)}$s. The average between-sum-of-squares matrix B in the feature space is

$$B_\phi = \frac{1}{l}\sum_{i=1}^{l} B_\phi^{(i)} = \frac{1}{l}\sum_{i=1}^{l} \tilde{\Phi}_J^{(i)}\left(\tilde{\Phi}_J^{(i)}\right)^T. \quad (22)$$

Similar to KPCA [14], we have the eigenvector equation $\lambda\mathbf{v} = B_\phi\mathbf{v}$. Clearly, all solutions must lie in the span of $\phi(\mathbf{x}_1), \cdots, \phi(\mathbf{x}_l)$. Therefore, there exist coefficients $\alpha_i (i = 1, \cdots, l)$ such that

$$\mathbf{v} = \sum_{i=1}^{l} \alpha_i \phi(\mathbf{x}_i). \quad (23)$$

It is also true that for all $k = 1, \cdots, l$, we have

$$\lambda\left(\phi(\mathbf{x}_k)\cdot\mathbf{v}\right) = \left(\phi(\mathbf{x}_k)\cdot B_\phi\mathbf{v}\right). \quad (24)$$

Substituting (23) and (22) into (24), we obtain the left-hand side of (24)

$$\lambda\left(\phi(\mathbf{x}_k)\cdot\mathbf{v}\right) = \lambda\sum_{i=1}^{l}\alpha_i\phi(\mathbf{x}_k)\cdot\phi(\mathbf{x}_i). \quad (25)$$

For the right-hand side of (24), we have (see Appendix A)

$$l\left(\phi(\mathbf{x}_k)\cdot B_\phi\mathbf{v}\right) = \left[\phi(\mathbf{x}_k)\cdot\tilde{\Phi}_J^{(1)}\cdots\phi(\mathbf{x}_k)\cdot\tilde{\Phi}_J^{(l)}\right]K_J\alpha \quad (26)$$

for all $k = 1, \cdots, l$. Here, $\alpha = (\alpha_1, \cdots, \alpha_l)^T$, and

$$K_J = \begin{pmatrix} \left(\tilde{\Phi}_J^{(1)}\right)^T\phi(\mathbf{x}_1) & \cdots & \left(\tilde{\Phi}_J^{(1)}\right)^T\phi(\mathbf{x}_l) \\ \cdots & \cdots & \cdots \\ \left(\tilde{\Phi}_J^{(l)}\right)^T\phi(\mathbf{x}_1) & \cdots & \left(\tilde{\Phi}_J^{(l)}\right)^T\phi(\mathbf{x}_l) \end{pmatrix}. \quad (27)$$

Define

$$K = [k_{ij}] = [\phi(\mathbf{x}_i)\cdot\phi(\mathbf{x}_j)] \quad (28)$$

and

$$\tilde{K} = K_J^T K_J. \quad (29)$$

We obtain

$$l\lambda K\alpha = \tilde{K}\alpha \quad (30)$$

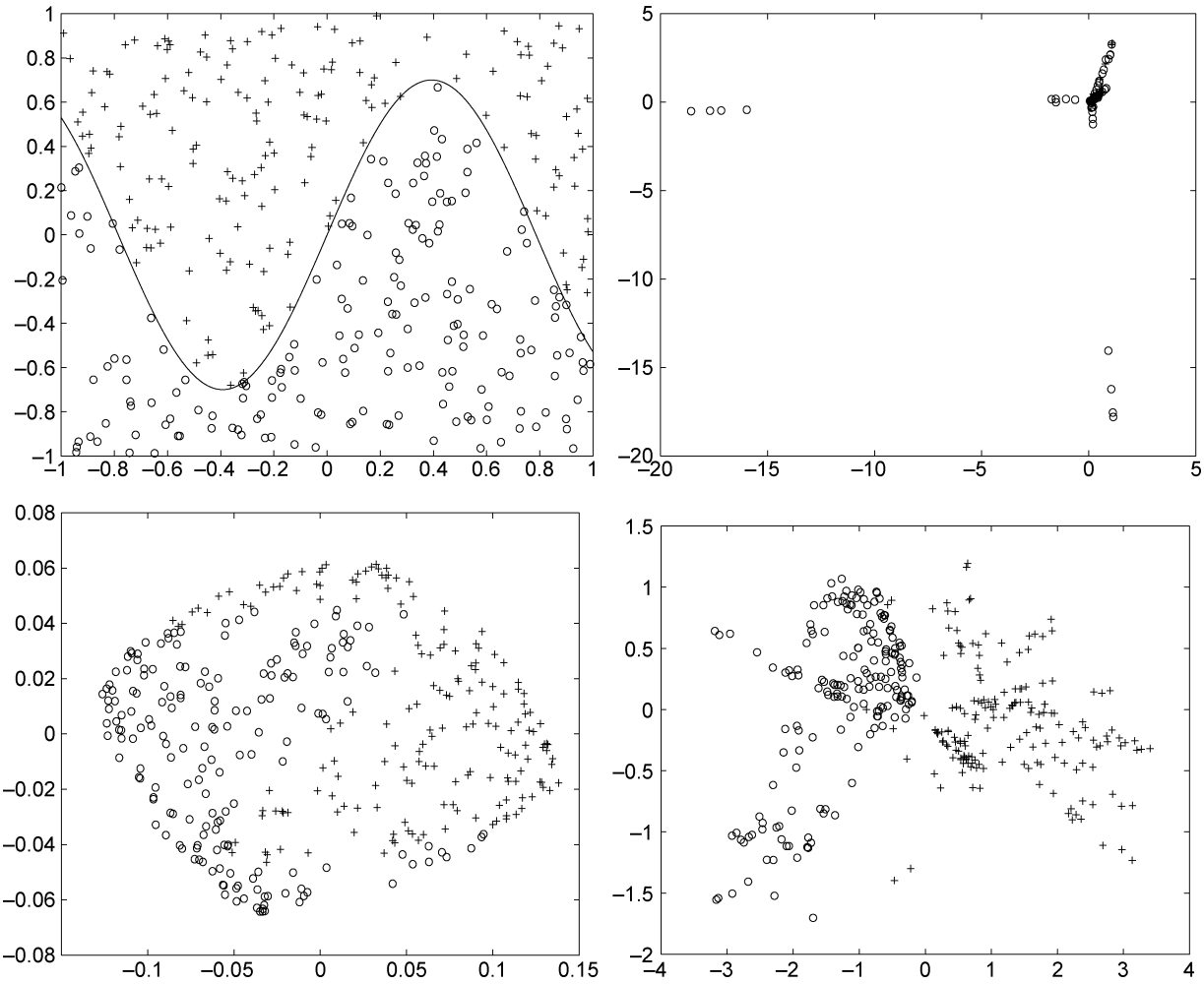


Fig. 1. Upper left panel. Two-dimensional toy example. Upper right panel. Subspace computed by KPCA. Lower left panel. Subspace computed by KPooLS with σ equal to 1. Lower right panel. Subspace computed by KPooLS with σ equal to 0.1.

which is a generalized eigenvector problem [20]. By solving (30), the ith principal component u_i of $\mathbf{x}$ can be calculated according to

$$u_i = \mathbf{v}_i \cdot \phi(\mathbf{x}) = \sum_{k=1}^{l} \alpha_k^i k(\mathbf{x}, \mathbf{x}_k) \tag{31}$$

where $\mathbf{v}_i$ denotes the ith eigenvector of the feature space.

A. Centering in Feature Space

Computing both K (28) and $\tilde{K}$ (29) requires centering the data. Similar to KPCA [14], the data can be centered in the feature space as follows:

$$\tilde{\phi}(\mathbf{x}_k) = \phi(\mathbf{x}_k) - \frac{1}{l}\sum_{i=1}^{l}\phi(\mathbf{x}_i). \tag{32}$$

Let $\mathbf{1}_l$ be a $l \times l$ matrix with entries $(\mathbf{1}_l)_{ij} := 1/l$. Then, the centered Gram matrix K^c is given by

$$K^c = K - \mathbf{1}_l K - K\mathbf{1}_l + \mathbf{1}_l K \mathbf{1}_l. \tag{33}$$

On the other hand, $\tilde{\Phi}_J^{(i)}$ is different. It is easy to see that centered $\tilde{\Phi}_J^{(i)}$ is the same. Let K_J^c be the centered matrix $\tilde{K}$. We have

$$\tilde{K}^c = (K_J^c)^T K_J^c \tag{34}$$

where

$$K_J^c = K_J - K_J \mathbf{1}_l. \tag{35}$$

Thus, the generalized eigenvector problem (30) becomes

$$(l)\lambda K^c \alpha = \tilde{K}^c \alpha. \tag{36}$$

V. Kernel Pooled Local Subspace Algorithm

A. Implementation Detail

Calculating K_J is the key step in the implementation. Here, we show how it is done. We need only show how to compute each submatix $(\tilde{\Phi}_J^{(i)})^T \phi(\mathbf{x}_k)$ of K_J. Let $\mathbf{x}_{j,1}^{(i)}, \ldots, \mathbf{x}_{j,l_j}^{(i)}$ be the l_j points from class j in the neighborhood of the ith point. Notice that

$$\begin{aligned}
&\sqrt{\hat{\pi}_j^{(i)}}\left(\bar{\phi}_j\left(\mathbf{x}^{(i)}\right) - \bar{\phi}\left(\mathbf{x}^{(i)}\right)\right)^T \phi(\mathbf{x}_k) \\
&= \sqrt{\hat{\pi}_1^{(i)}}\left(\frac{1}{l_j}\sum_{y_k=j}\phi\left(\mathbf{x}_k^{(i)}\right)^T\phi(\mathbf{x}_k) - \frac{1}{l_N}\sum_{k=1}^{l_N}\phi\left(\mathbf{x}_k^{(i)}\right)^T\phi(\mathbf{x}_k)\right) \\
&= \mathbf{1}_{l_j}^T K_{j,k}^{(i)} - \mathbf{1}_{l_N}^T K_k^{(i)}
\end{aligned}$$

where $\mathbf{1}_{l_j} = (1/l_j, \cdots, 1/l_j)^T$, $\mathbf{1}_{l_N} = (1/l_N, \cdots, 1/l_N)^T$, $K_{j,k}^{(i)} = (k(\mathbf{x}_{j,1}^{(i)}, \mathbf{x}_k), \cdots, k(\mathbf{x}_{j,l_j}^{(i)}, \mathbf{x}_k))^T$, and $K_k^{(i)} =$

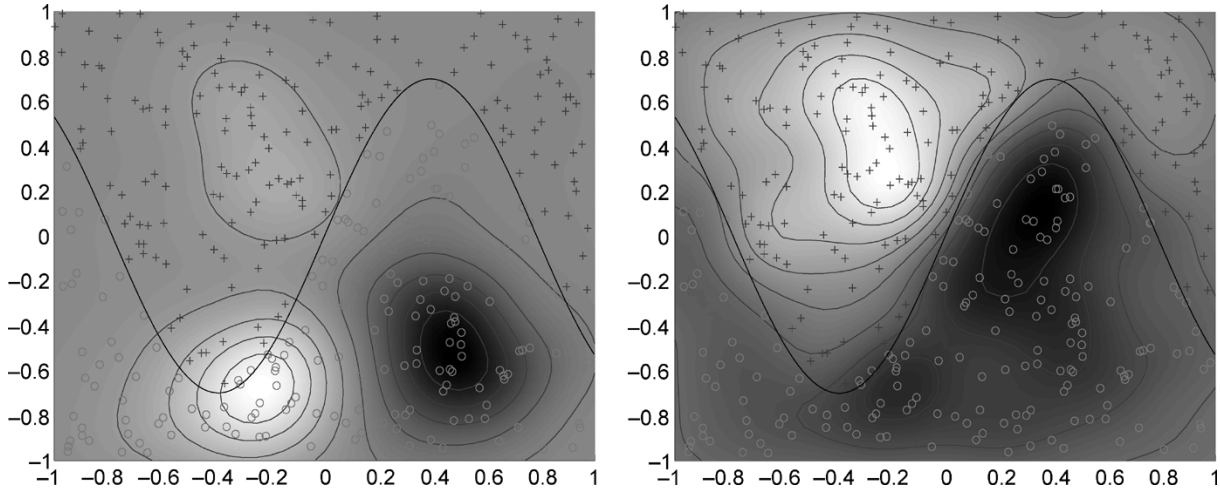


Fig. 2. Left panel. First principal component values of KPCA. Right panel. First principal component values of KPooLS.

$(k(\mathbf{x}_1^{(i)}, \mathbf{x}_k), \cdots, k(\mathbf{x}_{l_N}^{(i)}, \mathbf{x}_k))^T$, $k = 1, \cdots, l$. Thus, a submatrix of K_J corresponding to i and k, $(\tilde{\Phi}_J^{(i)})^T \phi(\mathbf{x}_k)$ can be calculated according to

$$\begin{bmatrix} \mathbf{1}_{l_1}^T K_{1,k}^{(i)} - \mathbf{1}_{l_N}^T K_k^{(i)} \\ \cdots \\ \mathbf{1}_{l_J}^T K_{J,k}^{(i)} - \mathbf{1}_{l_N}^T K_k^{(i)} \end{bmatrix}. \tag{37}$$

B. Kernel Local Pooling Algorithm

Here, we summarize the main steps of the kernel pooled local discriminant subspace (KPooLS) algorithm. Note that KPooLS has two procedural ("meta") parameters: l_N used to determine the local neighborhood for pooling and σ in a Gaussian kernel or n in a polynomial kernel. In the experiments reported below, these procedural parameters are determined through cross-validation over training data.

Algorithm 1 (KPooLS algorithm)

1) Calculate K (28).
2) Calculate K^c (33).
3) For each training point $\phi(\mathbf{x}_i)$ ($i = 1, \cdots, l$), find l_N nearest neighbors using kernelized dynamic neural network (DANN), and calculate (37).
4) Stack all (37), for $i = 1, \cdots, l$ and $k = i, \ldots l$ to calculate K_J.
5) Calculate $\tilde{K}^c$ by (35) and (34).
6) Solve the generalized eigenvector (36).

Note that nearest-neighbor computation in Step 3 involves a kernelized version of the DANN algorithm (KDANN). KDANN is iterated only once in the experiments reported in this paper. The KDANN algorithm is described in Appendix B. When l_N in Step 3 is set to l, KPooLS results in GDA (assuming data will be sphered). In this sense, KPooLS can be viewed as a generalization of GDA.

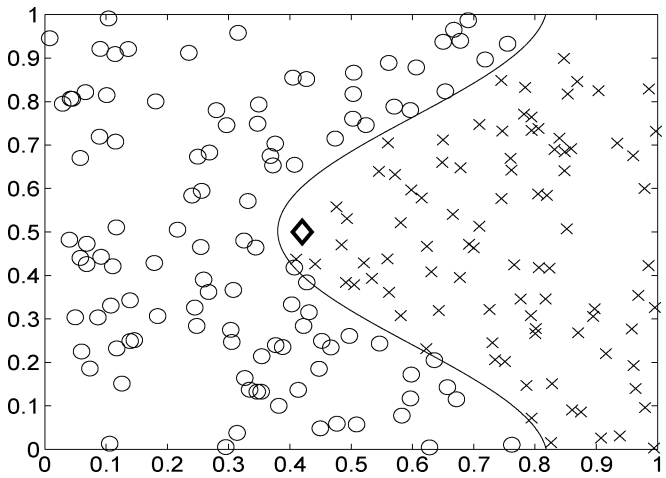


Fig. 3. Two class data in two dimensions. The test point is chosen to be close to the boundary.

VI. Two-Dimensional Data Illustration

Here, we use a two-dimensional (2-D) toy example to illustrate subspace computation by KPCA and KPooLS using Gaussian kernels

$$k(\mathbf{x}, \mathbf{x}') = e^{-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{\sigma^2}}. \tag{38}$$

The upper left panel in Fig. 1 shows the 2–D toy example, where the two class data are uniformly distributed in two dimensions, separated by a sinusoidal decision boundary. The upper right panel in the figure shows the projection of the data onto the first two eigenvectors computed by KPCA. The two lower panels plot the projection of the data onto the first two eigenvectors of the feature space computed by the KPooLS algorithm with σ equal to 1 and 0.1, respectively. The lower right panel shows that the first eigenvector of the feature space sufficiently separates the two class. On the other hand, KPCA fails to do so (we tried different σ values and obtained similar results).

Fig. 2 shows the intensity values of the first principal component when the 2-D space is projected onto the first eigenvector of the feature space. The intensity values of the two classes

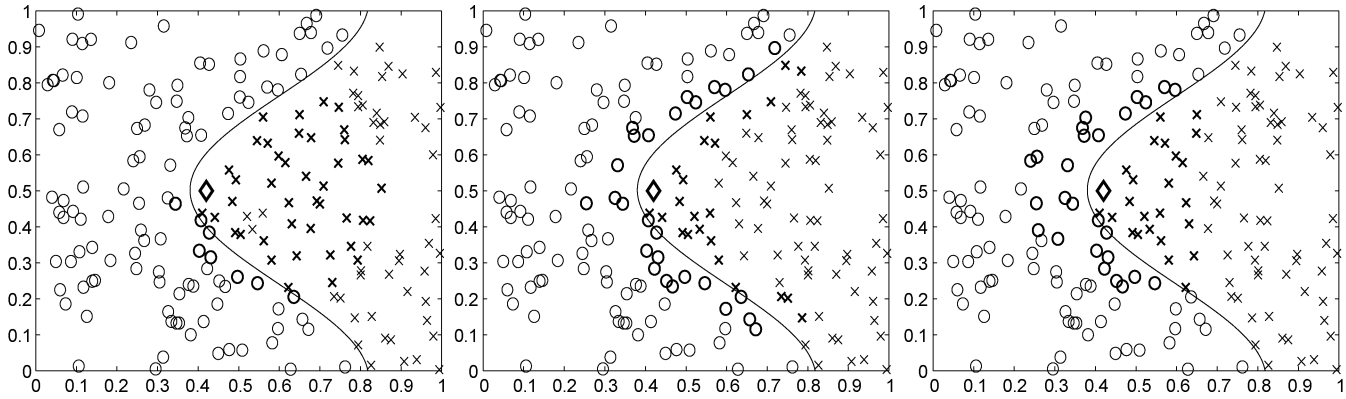


Fig. 4. Nearest neighborhoods obtained by KDANN using Gaussian kernels. Left panel: $\sigma = 0.2$. Middle panel: $\sigma = 0.5$. Right panel: $\sigma = 1$.

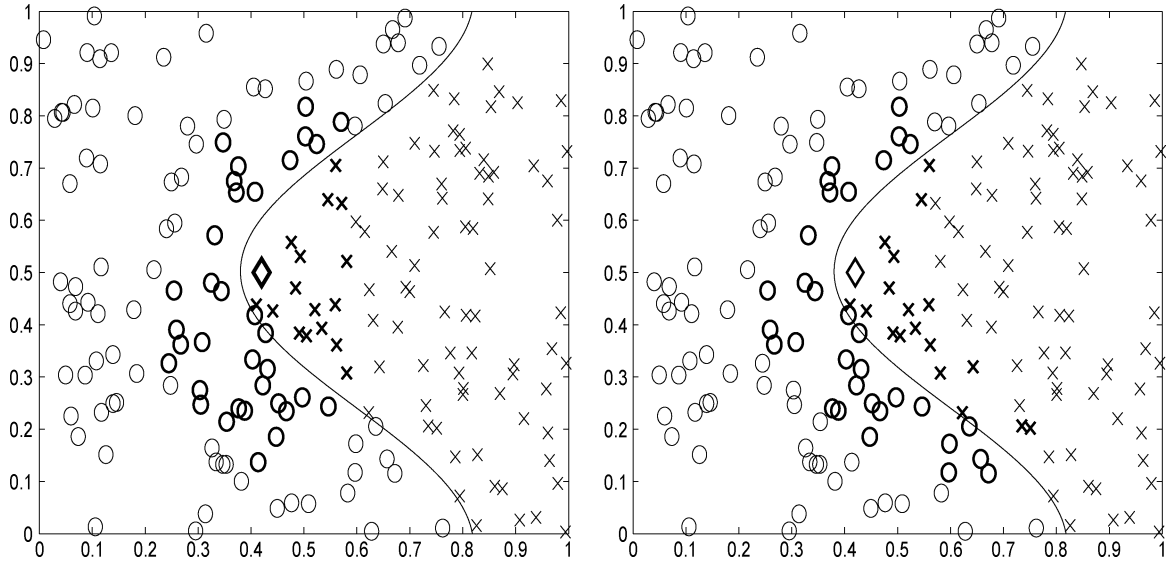


Fig. 5. Nearest neighborhood computed by KDANN with a polynomial kernel. Left panel: Second order $n = 2$. Right panel: Fourth order $n = 4$.

are visibly different for KPooLS, thus showing that this one-dimensional nonlinear subspace captures discriminant information. However, this is not the case for KPCA.

A crucial step of the KPooLS algorithm is the estimation of local between sum-of-squares matrices. This estimation depends critically on how the local neighborhood of each training point is computed. KPooLS invokes KDANN to compute these neighborhoods, where different kernels result in different neighborhoods. Here, we use two class data in two dimensions to illustrate local calculation. Fig. 3 shows the 2-D data, where the two classes are separated by a sinusoidal class boundary. The test point close to the boundary is chosen whose nearest neighbors are computed using KDANN.

Fig. 4 shows the nearest neighborhoods of the test point computed by KDANN using a Gaussian kernel. The left panel shows the neighborhood with small σ: $\sigma = 0.2$ (38). The neighborhood is somewhat errant. The middle panel uses a moderate σ: $\sigma = 0.5$. The neighborhood extends exactly along the nonlinear boundary. In the right panel, a larger σ is used: $\sigma = 1$. It also extends a little bit along the nonlinear boundary, but not far at all. It still stays closer to the test point. This implies that the choice of σ is very critical in KDANN. In practice, cross-validation is used to pick up a good parameter σ.

Fig. 5, on the other hand, shows the nearest neighborhood of the test point computed by KDANN using a polynomial kernel. In the left panel, the second-degree polynomial kernel is used (39): $n = 2$. This case is similar to the Gaussian kernel with large σ. The neighborhood extends along with the decision boundary slightly. In the right panel, the fourth-degree polynomial kernel is used: $n = 4$. This case is similar to the Gaussian kernel with $\sigma = 0.5$ and the neighborhood region extends nicely along the decision boundary.

VII. Experimental Results

In the following, we use six data sets to examine the classification performance of the nearest neighbor rule (3NN) with each subspace representation:

- KPCA—Kernel PCA;
- GDA—Kernel Fisher discriminant analysis;
- KPooLS—Kernel pooled local subspace method proposed here.

Since our focus here is on subspace methods, a simple classifier is preferred.

For each data set, three experiments were performed. In the first experiment, each subspace method employs Gaussian ker-

TABLE II
UCI DATA DESCRIPTION FOR BINARY CLASSIFICATION PROBLEMS

Data set	dim. of X	neg. data	pos.
BreastCancer	10	196	81
Ionosphere	33	225	126
Heart Cleve	13	160	137
Heart Hun	13	188	106

nels (38) only. In the second experiment, each subspace method employs polynomial kernels only

$$k(\mathbf{x}, \mathbf{x}') = (\mathbf{x} \cdot \mathbf{x}' + 1)^n \tag{39}$$

where n is allowed to take on values 2, 3, or 4. In the third experiment, each method selects either Gaussian (38) or polynomial (39) kernels automatically through cross-validation. In addition, each method determines kernel parameters through cross-validation over training data.

For each data set, we randomly select 60% of the data as training and remaining 40% as testing. This process is repeated ten times, and the everage error over ten runs are plotted.

Our analysis is also performed with respect to M: the number of selected feature vectors. Usually, there is an optimal number of feature vectors for each algorithm. The classification error initially declines with the addition of new features, achieves a minimum, and then starts to increase [30]. To illustrate the effects of the number of selected feature vectors M, we plot error rates as a function of M for all the data sets.

A. UCI Data Sets

The data sets used are Ionosphere, Breast Cancer, Heart Cancer Cleveland, Heart Cancer Hungarian. They all contain two classes: negative and positive classes. The details of the data sets are given in Table II.

Before applying the subspace methods, we first normalize the data so that on each dimension the variable is in the range [0,1]. Remember that d is the dimension of data $\mathbf{x}$. Let x_i be the variable on the ith dimension, x_i^{min} the minimum, and x_i^{max} the maximum of x_i over all the data. Then, x_i is normalized according to

$$\tilde{x}_i = \frac{x_i - x_i^{min}}{x_i^{max} - x_i^{min}}, \quad \text{for} \quad i = 1, \ldots, d. \tag{40}$$

Note that for all the three algorithms (KPCA, GDA, and KPooLS), the data are centered in feature space by indirectly manipulating Gram and other matrices involved. Therefore, there is no need to center the data in the input space. The normalization process has two benefits without introducing bias in favor of or against a particular method. First, it scales each dimension to the same range so that no single dimension dominates. Second, it significantly cuts the cost of cross-validation. Often, in order to search for the best Gaussian kernel parameter, a wide range of σ has to be examined. For the Gaussian kernel (38), small (or large) σ is needed when the norm $||\mathbf{x} - \mathbf{x}'||^2$ is small (or large). By scaling all variables by (40), we can greatly reduce the search range of the Gaussian kernel parameter σ.

The classification performance of 3NN with each subspace method on the UCI data sets are shown in Fig. 6. GDA usually achieves the best performance when $M = 1$. This is justified by the Fisher criteria (7): Only one eigenvector has nonzero eigenvalue, so only one dimension is enough for binary classification. Let us take a look at the performance of KPCA, GDA, and KPooLS at $M = 1$. If Gaussian kernel is employed, KPooLS is the best on heart cleve and heart hungery. If polynomial kernel is employed, KPooLS is the best on breast cancer, heart cleve, and ionosphere. If any kernel is allowed, KPooLS is the best on heart cleve and heart hungery. In the cases that KPooLS is not the best, either all of the three methods achieve close performance, or GDA is slightly better than KPooLS.

B. Cat and Dog Image Data

In this experiment, the data set is composed of 200 images of cat and dog faces.[1] Each image is a black-and-white 64×64 pixel image, and the images have been registered by aligning the eyes. Sample cat and dog images are shown in Fig. 7. The cat and dog data then has 4096 features. The data are normalized so that on each dimension, the variable is in the range [0,1].

The results are shown in Fig. 8. KPCA performed poorly on the Cat and Dog data, especially when the number of principal dimensions is small. KPooLS is better than KPCA but worse than GDA when the Gaussian kernel is employed.

C. Feret Face Image Data

This data set consists of 160 FERET faces: 20 individuals (classes), each of which contains eight views. All these face images were aligned, normalized, and resized into a standard image size of 150×130 that is commonly used in face recognition tasks, as described in [31]. The resulting vector space has a dimensionality of 19 500. The distribution of face images is highly complex. Fig. 9 shows sample images of the FERET database.

The performance results are shown in Fig. 10. The KPooLS algorithm is better than KPCA but somewhat worse than GDA.

VIII. DISCUSSIONS

Here, we discuss the experimental results. The experiments show that on the UCI data sets, our kernel local pooling method performs better than GDA, whereas on the two image data, GDA is better. KPooLS' performance depends greatly on the computation of local neighborhoods. In the experiments reported here, we used a fixed number $(l_N = 30)$ of the nearest neighbors to estimate the local between sum-of-squares matrices. This number might be too small in some cases. When two classes are close to each other, the nearest neighborhood contains data points from both class, as shown in the left panel in Fig. 11. In this case, KPooLS can adapt well the neighborhood along the boundary. If two classes are far from each other (as shown in the right panel in Fig. 11), the nearest neighborhood may contain data points only from one class or very few points from the other class. In this case, KPooLS may not capture critical discriminant directions. We can pontetially enlarge the neighborhood (l_N) to mitigate this problem. However, when increasing the size of the neighborhood, it is necessary to whiten the between sum-of-squares matrices $B^{(i)}$s, as in LDA. In fact, if we add a whitening step for computing local $B^{(i)}$s

[1]We would like to thank B. Draper for providing us with the data.

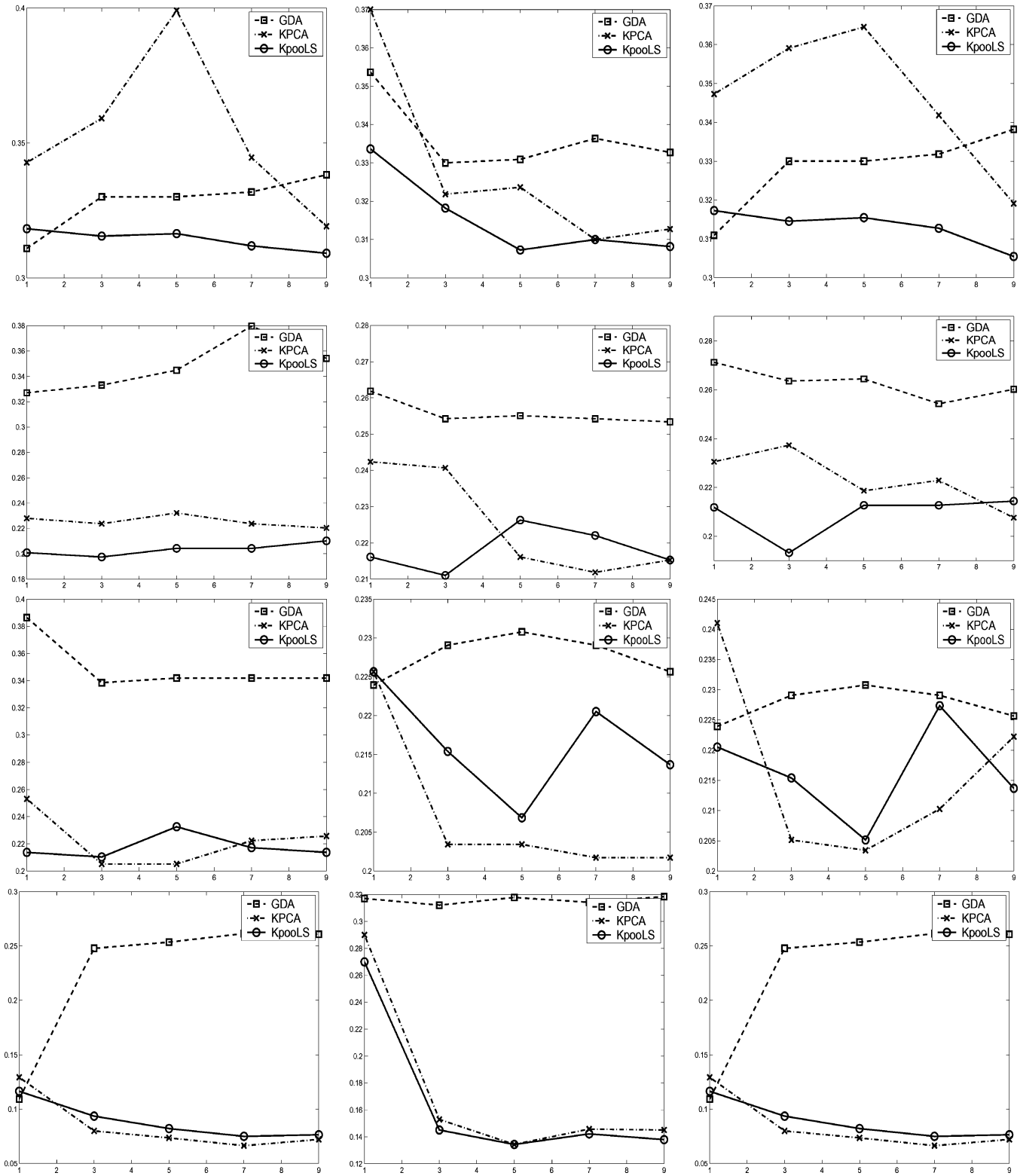


Fig. 6. Average error rates on the UCI data sets as a function of M. Each row corresponds to a data set. From top to bottom, they are breast cancer, heart cleve, heart hungery, and ionosphere. The columns, from left to right, correspond to Gaussian, polynomial, and any kernels (Gaussian or polynomial).

and increase the number of the nearest neighbors, the performance of KPooLS will at least be the same as GDA. By adapting this neighborhood, KPooLS can potentially outperform GDA on high-dimensional data.

It can be noticed in the experiments that GDA performed quite well on the image data sets but relatively poorly on the UCI data sets. This can be explained from the point of view of learning theory [32]. In [33] and [34], we demonstrate that reg-

Fig. 7. Sample cat and dog images.

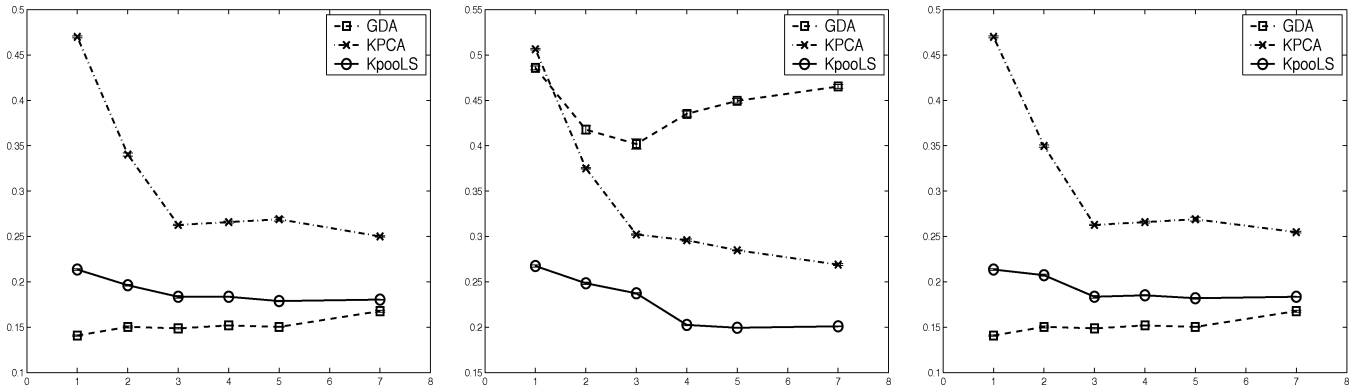


Fig. 8. Average error rates on the cat and dog data as a function of M. From left to right: Gaussian, polynomial, and any kernels.

Fig. 9. Sample Feret face images.

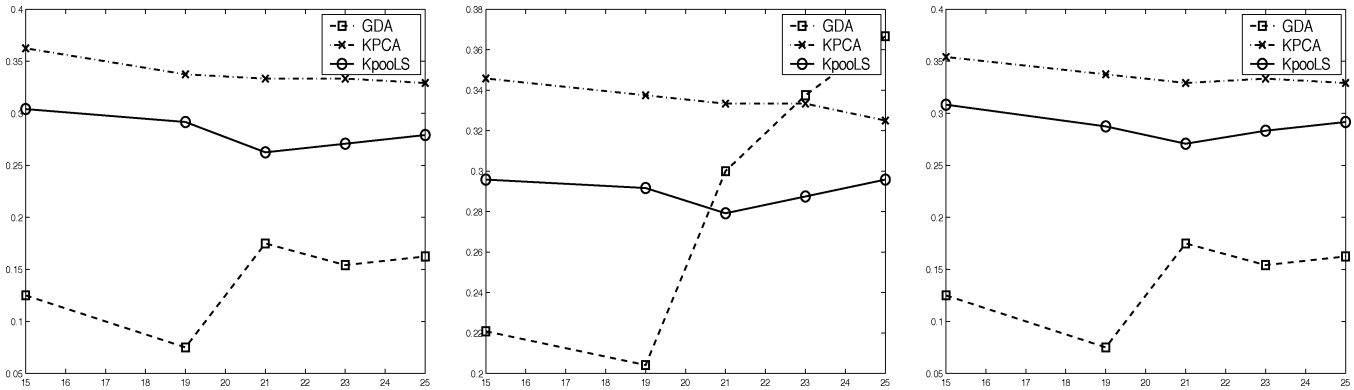


Fig. 10. Average error rates on the Feret facial image data as a function of M. From left to right: Gaussian, polynomial, and any kernels.

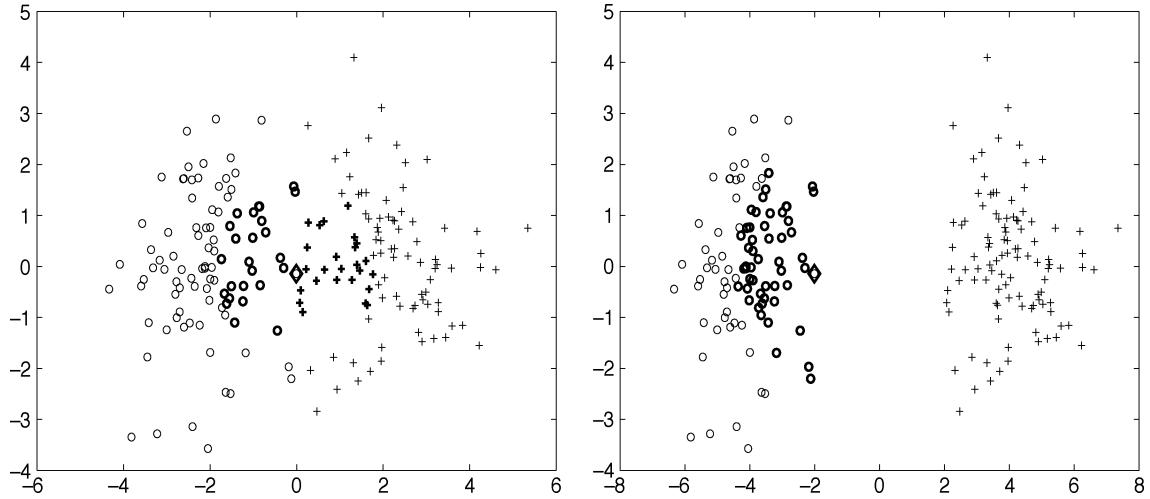


Fig. 11. When does KPooLS perform well? Left panel: Nearest neighborhood contains data from both classes. Right panel: Nearest neighborhood contains data only from one class.

ularized least squares (RLS) classifiers achieve similar performance to SVMs on a number of data sets, including the UCI data sets and the Cat and Dog data used in this paper. We also show that GDA can be viewed as a special case of RLS by setting a regularization constant to zero [34]. The regularization constant is used to control the complexity of function approximators. RLS chooses the regularization constant through cross validation. A small regularization constant prefers more complex functions, whereas a large value gives rise to simple or smooth functions.

It is shown [34] that on the UCI data sets, RLS prefers to have a larger regularization constant, which means that smoother functions produce better generalization performance. However, on the image data such as the Cat and Dog data, RLS chooses a very small regularization constant, implying that more complex functions are needed to fit the Cat and Dog data. Since GDA is at one end of the complexity spectrum of RLS (the complex end), its performance is similar to that of RLS when RLS prefers more complex classifiers. However, its performance is far worse than RLS when simple (less complex) classifiers are preferred.

IX. Summary

This paper presents a kernel pooled local subspace method for learning low-dimensional representations for classification. This method performs a nonlinear global dimensionality reduction by pooling local dimension information and using the kernel trick to extend to the nonlinear case. The resulting subspaces are nonlinear and discriminant, whereby better classification performance and greater computational efficiency can be achieved. The experiments show that KPooLS outperformed both KPCA and GDA on several UCI data sets, whereas it did not perform as well as GDA on the image data sets, especially when the Gaussian kernel was employed. The performance of KPooLS can be boosted by adapting local neighborhoods for computing locally between sum-of-squares matrices. We conclude that KPooLS adds another tool to subspace computation and can achieve better performance on some classification problems over competing methods such GDA and KPCA.

Appendix

A. Derivation of Equation (26)

To derive (26), from the right-hand side of (24), we have

$$\begin{aligned} & l\left(\phi(\mathbf{x}_k)\cdot B_\phi \mathbf{v}\right) \\ &= \phi(\mathbf{x}_k)\cdot \sum_{i=1}^{l} \tilde{\Phi}_J^{(i)} \left(\tilde{\Phi}_J^{(i)}\right)^T \sum_{m=1}^{l} \alpha_m \phi(\mathbf{x}_m) \\ &= \phi(\mathbf{x}_k)\cdot \sum_{i=1}^{l} \tilde{\Phi}_J^{(i)} \left(\tilde{\Phi}_J^{(i)}\right)^T \left[\phi(\mathbf{x}_1)\cdots\phi(\mathbf{x}_l)\right]\alpha \\ &= \phi(\mathbf{x}_k)\cdot \sum_{i=1}^{l} \tilde{\Phi}_J^{(i)} \left[\left(\tilde{\Phi}_J^{(i)}\right)^T \phi(\mathbf{x}_1)\cdots \left(\tilde{\Phi}_J^{(i)}\right)^T \phi(\mathbf{x}_l)\right]\alpha \\ &= \left[\phi(\mathbf{x}_k)\cdot\tilde{\Phi}_J^{(1)}\cdots\phi(\mathbf{x}_k)\cdot\tilde{\Phi}_J^{(l)}\right] K_J\alpha \end{aligned} \tag{41}$$

for all $k = 1, \cdots, l$. Here, $\alpha = (\alpha_1, \cdots, \alpha_l)^T$, and K_J is given by (27).

B. Kernelized DANN

1) Review of the DANN Algorithm: For the pth data $(p = 1, \ldots l)$, KDANN is computed once to get adapted nearest neighborhood. For simplicity, in this section, we drop the notation (p). For example, $B^{(p)}$ is replaced by B and $\bar{x}^{(p)}$ is replaced by $\bar{x}$, but remember that thus, $\bar{x}$ is the overall mean of all data points within the nearest neighborhood of pth data. Later when we mention "within the nearest neighborhood," we mean the neighborhood around the pth data.

In the input space, the local between and within sum-of-squares matrices are

$$B = \sum_{j=1}^{J} \hat{\pi}_j (\bar{x}_j - \bar{x})(\bar{x}_j - \bar{x})^T \tag{42}$$

and

$$W = \frac{\sum_{j=1}^{J} \sum_{y_i = j} \omega_i (x_i - \bar{x}_j)(x_i - \bar{x}_j)^T}{\sum_{i=1}^{l_N} \omega_i} \tag{43}$$

where

$$\hat{\pi}_j = \frac{\sum_{y_i=j} \omega_i}{\sum_{i=1}^{l} \omega_i}. \tag{44}$$

Define $d_i = ||\Sigma_0^{1/2}(x_i - x_p)||$ and $h = \max_{i \in l_N} d_i$, where Σ_0 is the initial metric. Then, the weight assigned to the x_i are given by

$$\omega_i = \left[1 - \left(\frac{d_i}{h}\right)^{\beta}\right]^{\beta} \tag{45}$$

where $\beta = 3$ by default.

The DANN metric is defined as

$$\begin{aligned}\Sigma &= W^{-1}BW^{-1} \\ &= W^{-\frac{1}{2}}\left(W^{-\frac{1}{2}}B^{-1}W^{-\frac{1}{2}}\right)W^{-\frac{1}{2}} \\ &= W^{-\frac{1}{2}}B^{*}W^{-\frac{1}{2}}\end{aligned} \tag{46}$$

where $B^* = W^{-1/2}B^{-1}W^{-1/2}$, which is the between sum-of-squares in the sphered space. To prevent neighborhoods from extending infinitely in the null space of B^*, a better metric is given by

$$\Sigma = W^{-\frac{1}{2}}(B^* + \varepsilon I)W^{-\frac{1}{2}} \tag{47}$$

where ε is some small number. In our experiments, we set $\varepsilon = 1$.

The DANN algorithm is briefly as follows. First, a initial metric is given ($\Sigma_0 = I$); find l_N nearest neighbors with respect to the distance $d_i = ||\Sigma_0^{1/2}(x_i - x_p)||$. Then, update B, W and Σ; Set $\Sigma_0 = \Sigma$, and repeat the process if needed.

2) Kernelized DANN: To kernelize DANN, we must compute the between (42) and within (43) the sum of squares matrices in the feature space:

$$\begin{aligned}B_\phi(x_0, \Sigma_0, h) &= \sum_{j=1}^{J} \hat{\pi}_j \left(\bar{\phi}_j(x) - \bar{\phi}(x)\right)\left(\bar{\phi}_j(x) - \bar{\phi}(x)\right)^T \\ &= \tilde{\Phi}_J \tilde{\Phi}_J^T\end{aligned} \tag{48}$$

and

$$\begin{aligned}W_\phi(x_0, \Sigma_0, h) &= \frac{\sum_{j=1}^{J} \sum_{y_i=j} \omega_i \left(\phi(x_i) - \bar{\phi}_j(x)\right)\left(\phi(x_i) - \bar{\phi}_j(x)\right)^T}{\sum_{i=1}^{l_N} \omega_i} \\ &= \tilde{\Phi}_{IJ} \tilde{\Phi}_{IJ}^T\end{aligned} \tag{49}$$

where

$$\hat{\pi}_j = \frac{\sum_{y_i=j} \omega_i}{\sum_{i=1}^{l_N} \omega_i}. \tag{50}$$

Here, matrices $\tilde{\Phi}_J$ and $\tilde{\Phi}_{IJ}$ are given by

$$\begin{aligned}\tilde{\Phi}_J = \Big[&\sqrt{\hat{\pi}_1}\left(\bar{\phi}_1(x) - \bar{\phi}(x)\right) \quad \sqrt{\hat{\pi}_2}\left(\bar{\phi}_2(x) - \bar{\phi}(x)\right) \\ &\cdots \quad \sqrt{\hat{\pi}_J}\left(\bar{\phi}_J(x) - \bar{\phi}(x)\right)\Big]\end{aligned} \tag{51}$$

$$\begin{aligned}\tilde{\Phi}_{IJ} = \Big[&\sqrt{\tilde{\omega_1}}\left(\phi(x_1) - \bar{\phi}_1(x)\right) \quad \sqrt{\tilde{\omega_2}}\left(\phi(x_2) - \bar{\phi}_1(x)\right) \\ &\cdots \quad \sqrt{\tilde{\omega_{l_N}}}\left(\phi(x_{l_N}) - \bar{\phi}_J(x)\right)\Big]\end{aligned} \tag{52}$$

where

$$\tilde{\omega_i} = \frac{\omega_i}{\sum_{i=1}^{l_N} \omega_i}. \tag{53}$$

Note that both $\tilde{\Phi}_J \tilde{\Phi}_J^T$ and $\tilde{\Phi}_{IJ} \tilde{\Phi}_{IJ}^T$ can be in an infinite-dimensional space. Thus, care must be taken. Outer products must be transformed into inner products so that the kernel trick can be applied. Let us consider $\tilde{\Phi}_J^T \tilde{\Phi}_J$ and $\tilde{\Phi}_{IJ}^T \tilde{\Phi}_{IJ}$.

Let V_J be the matrix whose columns are the eigenvectors of $\tilde{\Phi}_J^T \tilde{\Phi}_J$, and let Λ_J be the diagonal matrix whose diagonal entries are the eigenvalues of $\tilde{\Phi}_J^T \tilde{\Phi}_J$ (we will show later how to compute $\tilde{\Phi}_J^T \tilde{\Phi}_J$). We then have

$$\Psi_J = \tilde{\Phi}_J V_J \Lambda_J^{-\frac{1}{2}}. \tag{54}$$

Similarly, we define matrices V_{IJ} and Λ_{IJ} corresponding to the eigenvectors and eigenvalues of $\tilde{\Phi}_{IJ}^T \tilde{\Phi}_{IJ}$. Let

$$\Psi_{IJ} = \tilde{\Phi}_{IJ} V_{IJ} \Lambda_{IJ}^{-\frac{1}{2}}. \tag{55}$$

It follows that

$$\begin{aligned}W_\phi^{-1} &= \Psi_{IJ} \Lambda_{IJ}^{-1} \Psi_{IJ}^T \\ &= \tilde{\Phi}_{IJ} V_{IJ} \Lambda_{IJ}^{-\frac{1}{2}} \Lambda_{IJ}^{-1} \Lambda_{IJ}^{-\frac{1}{2}} V_{IJ}^T \tilde{\Phi}_{IJ}^T \\ &= \tilde{\Phi}_{IJ} V_{IJ} \Lambda_{IJ}^{-\frac{1}{2}} \Lambda_{IJ}^{-1} \Lambda_{IJ}^{-\frac{1}{2}} V_{IJ}^T \tilde{\Phi}_{IJ}^T \\ &= \tilde{\Phi}_{IJ} V_{IJ} \Lambda_{IJ}^{-2} V_{IJ}^T \tilde{\Phi}_{IJ}^T\end{aligned} \tag{56}$$

and

$$\begin{aligned}W_\phi^{-\frac{1}{2}} &= \Psi_{IJ} \Lambda_{IJ}^{-\frac{1}{2}} \Psi_{IJ}^T \\ &= \tilde{\Phi}_{IJ} V_{IJ} \Lambda_{IJ}^{-\frac{3}{2}} V_{IJ}^T \tilde{\Phi}_{IJ}^T.\end{aligned} \tag{57}$$

Therefore, the DANN metric in feature space becomes

$$\begin{aligned}\Sigma_\phi &= W_\phi^{-\frac{1}{2}} \left[B_\phi^* + \varepsilon I_\phi\right] W_\phi^{-\frac{1}{2}} \\ &= W_\phi^{-\frac{1}{2}} B_\phi W_\phi^{-\frac{1}{2}} + \varepsilon W_\phi^{-1} \\ &= \tilde{\Phi}_{IJ} V_{IJ} \Lambda_{IJ}^{-\frac{3}{2}} V_{IJ}^T \tilde{\Phi}_{IJ}^T \tilde{\Phi}_J \tilde{\Phi}_J^T \tilde{\Phi}_{IJ} V_{IJ} \Lambda_{IJ}^{-\frac{3}{2}} V_{IJ}^T \tilde{\Phi}_{IJ}^T \\ &\quad + \varepsilon \tilde{\Phi}_{IJ} V_{IJ} \Lambda_{IJ}^{-2} V_{IJ}^T \tilde{\Phi}_{IJ}^T.\end{aligned} \tag{58}$$

For any two points $\phi(x_p)$ and $\phi(x_q)$ in feature space, we have

$$\begin{aligned}&\phi(x_p)^T \Sigma_\phi \phi(x_q) \\ &\quad = \phi(x_p)^T \tilde{\Phi}_{IJ} V_{IJ} \Lambda_{IJ}^{-\frac{3}{2}} V_{IJ}^T \tilde{\Phi}_{IJ}^T \tilde{\Phi}_J \tilde{\Phi}_J^T \tilde{\Phi}_{IJ} V_{IJ} \Lambda_{IJ}^{-\frac{3}{2}} \\ &\qquad \times V_{IJ}^T \tilde{\Phi}_{IJ}^T \phi(x_q) + \varepsilon \phi(x_p)^T \tilde{\Phi}_{IJ} V_{IJ} \Lambda_{IJ}^{-2} V_{IJ}^T \tilde{\Phi}_{IJ}^T \phi(x_q) \\ &\quad = K_{pIJ} V_{IJ} \Lambda_{IJ}^{-\frac{3}{2}} V_{IJ}^T K_{IJ\_J} K_{IJ\_J}^T V_{IJ} \Lambda_{IJ}^{-\frac{3}{2}} V_{IJ}^T K_{qIJ}^T \\ &\qquad + \varepsilon K_{pIJ} V_{IJ} \Lambda_{IJ}^{-2} V_{IJ}^T K_{qIJ}^T\end{aligned}$$

where $K_{IJ\_J} = \tilde{\Phi}_{IJ}^T \tilde{\Phi}_J$, and $K_{pIJ} = \phi(x_p)^T \tilde{\Phi}_{IJ}$. This allows us to calculate

$$\left(\phi(x_p) - \phi(x_q)\right)^T \Sigma_\phi \left(\phi(x_p) - \phi(x_q)\right) \tag{59}$$

between a pair of data points in the feature space.

3) Calculation of $\tilde{\Phi}_J^T\tilde{\Phi}_J$*:* Let $\sqrt{\hat{\pi}_i} = a_i$ (50). We have

$$\begin{aligned}\tilde{\Phi}_J &= \Big[\sqrt{\hat{\pi}_1}\left(\bar{\phi}_1(x)-\bar{\phi}(x)\right) \quad \sqrt{\hat{\pi}_2}\left(\bar{\phi}_2(x)-\bar{\phi}(x)\right) \\ &\qquad \cdots \quad \sqrt{\hat{\pi}_J}\left(\bar{\phi}_J(x)-\bar{\phi}(x)\right)\Big] \\ &= \left[\left(\bar{\phi}_1(x)-\bar{\phi}(x)\right) \quad \left(\bar{\phi}_2(x)-\bar{\phi}(x)\right)\right. \\ &\qquad \left.\cdots \quad \left(\bar{\phi}_J(x)-\bar{\phi}(x)\right)\right]\Lambda \\ &= \left\{\left[\bar{\phi}_1(x) \quad \cdots \quad \bar{\phi}_J(x)\right]-\left[\bar{\phi}(x) \quad \cdots \quad \bar{\phi}(x)\right]\right\}\Lambda \\ &= \Phi\left\{\left[\frac{a_1}{l_1}I_1 \quad \cdots \quad \frac{a_J}{l_J}I_J\right]-\left[\frac{a_1}{l_N}I \quad \cdots \quad \frac{a_J}{l_N}I\right]\right\} \\ &= \Phi(A_J-B_J)\end{aligned}$$

where $\Lambda = \text{diag}(a_1 \cdots a_J)$, and $\Phi = [\phi(x_1)\cdots\phi(x_l)]$ is the feature space mapping of all training data. Here, I_j is $[\delta_j(1)\cdots\delta_j(l)]^T$, where

$$\delta_j(k) = \begin{cases} 1, & \text{if the } k\text{th data point is in class } j \\ & \text{and it is within nearest neighborhood} \\ 0, & \text{otherwise} \end{cases}$$

and $I = \sum_j^J I_j$. We define

$$\begin{aligned}A_J &= \left[\frac{a_1}{l_1}I_1 \quad \cdots \quad \frac{a_J}{l_J}I_J\right] \\ B_J &= \left[\frac{a_1}{l_N}I \quad \cdots \quad \frac{a_J}{l_N}I\right].\end{aligned} \tag{60}$$

Thus

$$\begin{aligned}\tilde{\Phi}_J^T\tilde{\Phi}_J &= (A_J-B_J)^T\Phi^T\Phi(A_J-B_J) \\ &= (A_J-B_J)^TK(A_J-B_J).\end{aligned}$$

4) Calculation of $\tilde{\Phi}_{IJ}^T\tilde{\Phi}_{IJ}$*:* Let $\sqrt{\omega_i/\sum_{i=1}^{l_N}\omega_i} = b_i$.

$$\begin{aligned}\tilde{\Phi}_{IJ} &= \left[b_1\left(\phi(x_1)-\bar{\phi}(x_1)\right) \quad b_2\left(\phi(x_2)-\bar{\phi}(x_1)\right)\right. \\ &\qquad \left.\cdots \quad b_{l_N}\left(\phi(x_{l_J})-\bar{\phi}(x_J)\right)\right] \\ &= \Phi\left\{[b_1\Delta_{11} \quad \cdots \quad b_{l_1}\Delta_{l_11} \quad b_{l_1+1}\Delta_{12} \quad \ldots \quad b_{l_N}\Delta_{l_JJ}]\right. \\ &\qquad \left.-\left[\frac{b_1}{l_1}I_1 \quad \ldots \quad \frac{b_{l_1}}{l_1}I_1 \quad \frac{b_{l_1+1}}{l_2}I_2 \quad \cdots \quad \frac{b_{l_N}}{l_J}I_J\right]\right\} \\ &= \Phi(C_J-D_J)\end{aligned}$$

where $\Delta_{ij} = [\delta_{ij}(1)\ldots\delta_{ij}(l)]^T$, for $i = 1,\ldots l_j$. It is a $l \times 1$ vector, and

$$\delta_{ij}(k) = \begin{cases} 1, & \text{if the } k\text{th data point is in class } j \\ & \text{and } \sum_{\substack{m=1 \\ m\neq i}}^{l_j}\delta_{mj}(k) = 0;\ \sum_{m=1}^{l_j}\sum_{k=1}^{l}\delta_{mj}(k) = l_j \\ & \text{and the } k\text{th data point is within the} \\ & \text{nearest neighborhood} \\ 0, & \text{otherwise.} \end{cases}$$

Basically, Δ_{ij} is a pick-up vector for each i, and it picks up a distinct neighbor, which is in class j.

$$\begin{aligned}C_J &= [b_1\Delta_{11} \quad \cdots \quad b_{l_1}\Delta_{l_11} \quad b_{l_1+1}\Delta_{12} \quad \ldots \quad b_{l_N}\Delta_{l_JJ}] \\ D_J &= \left[\frac{b_1}{l_1}I_1 \quad \ldots \quad \frac{b_{l_1}}{l_1}I_1 \quad \frac{b_{l_1+1}}{l_2}I_2 \quad \cdots \quad \frac{b_{l_N}}{l_J}I_J\right].\end{aligned} \tag{61}$$

Therefore

$$\tilde{\Phi}_{IJ}^T\tilde{\Phi}_{IJ} = (C_J-D_J)^TK(C_J-D_J). \tag{62}$$

5) Calculation of $K_{IJ\_J} = \tilde{\Phi}_{IJ}^T\tilde{\Phi}_J$*:*

$$K_{IJ\_J} = (C_J-D_J)^TK(A_J-B_J). \tag{63}$$

6) Calculation of $K_{pIJ} = \phi(x_p)^T\tilde{\Phi}_{IJ}$*:*

$$\phi(x_p) = \Phi E_p$$

where $E_p = [\delta_{1p}\ \delta_{2p}\ldots\delta_{lp}]^T$, $\delta_{lp} = 1$ when $l = p$; otherwise, $\delta_{lp} = 0$; Thus

$$\begin{aligned}K_{pIJ} &= \phi(x_p)^T\tilde{\Phi}_{IJ} \\ &= E_p^T\Phi^T\Phi(C_J-D_J) \\ &= K_p(C_J-D_J).\end{aligned} \tag{64}$$

K_p is the pth row of gram matrix K.

7) KDANN Algorithm: According to the above analysis, the main steps of the KDANN algorithm are given by the following.

Algorithm 2

1) Calculate the gram matrix $K = [\phi(x_i)\phi(x_j)]$.
2) Calculate A_J, B_J (60), C_J, D_J (61).
3) Calculate $\tilde{\Phi}_{IJ}^T\tilde{\Phi}_{IJ}$ (62), get its eigenvectors V_{IJ} and eigenvalues Λ_{IJ}.
4) Calculate K_{IJ_J} (63), K_{pIJ} (64).
5) Calculate the distance from every data point to the test data by (59).
6) Update the local neighbor according to the distance and go back to step 2.

REFERENCES

[1] T. F. Cootes and C. J. Taylor, "Active shape models: smart snakes," in *Proc. British Machine Vision Conf.*, 1992, pp. 9–18.

[2] M. C. Burl, U. M. Fayyad, P. Perona, P. Smyth, and M. P. Burl, "Automating the hunt for volcanos on venus," in *Proc. IEEE Conf. Comput. Vision Pattern Recogn.*, 1994.

[3] S. K. Nayar, S. Baker, and H. Murase, "Parametric feature detection," in *Proc. IEEE Conf. Comput. Vision Pattern Recogn.*, 1994, pp. 471–477.

[4] A. Pentland, B. Moghaddam, and T. Starner, "View-based and modular eigenspaces for face recognition," in *Proc. IEEE Conf. Comput. Vision Pattern Recogn.*, 1994.

[5] B. Moghaddam and A. Pentland, "Probabilistic visual learning for object representation," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 19, no. 7, pp. 696–710, Jul. 1997.

[6] M. Turk and A. Pentland, "Eigenfaces for recognition," *Cognitive Neurosci.*, vol. 3, no. 1, 1991.

[7] B. Moghaddam, "Principal manifolds and probabilistic subspaces for visual recognition," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 24, no. 6, pp. 780–788, Jun. 2002.

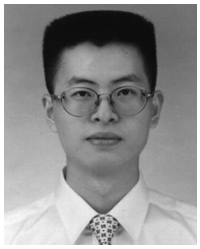
[8] K. Etemad and R. Chellappa, "Discriminant analysis for recognition of human faces," in *Proc. Int. Conf. Acoust., Speech, Signal Process.*, 1996, pp. 2148–2151.

[9] V. I. Belhumeur, J. P. Hespanha, and D. J. Kriegman, "Eigenfaces versus fisherfaces: recognition using class specific linear projection," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 19, no. 7, pp. 711–720, Jul. 1997.

[10] I. T. Jolliffe, *Principal Component Analysis*. New York: Springer-Verlag, 1986.

[11] P. Comon, "Independent, component analysis—A new concept?," *Signal Process.*, vol. 36, pp. 287–314, 1994.

[12] D. DeMers and G. Cottrell, "Nonlinear dimensionality reduction," in *Advances in Neural Information Processing Systems*. Cambridge, MA: MIT Press, 1993, vol. 5.
[13] M. A. Kramer, "Nonlinear principal components analysis using autoassociative neural networks," *Amer. Inst. Chem. Eng. J.*, vol. 32, no. 2, pp. 233–234, 1991.
[14] B. Scholkopf, A. Smola, and K. R. Muller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Comput.*, vol. 10, pp. 1299–1319, 1998.
[15] G. Baudat and F. Anouar, "Generalized discriminant analysis using a kernel approach," *Neural Comput.*, vol. 12, pp. 2385–2404, 2000.
[16] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge, U.K.: Cambridge Univ. Press, 2000.
[17] J. Lu, K. N. Plataniotis, and A. N. Venetsanopoulos, "Face recognition using kernel direct discriminant analysis algorithms," *IEEE Trans. Neural Networks*, vol. 14, no. 1, pp. 117–126, Jan. 2003.
[18] R. Courant and D. Hilbert, Eds., *Methods of Mathematical Physics*. New York: Interscience, 1953, vol. 1.
[19] T. Hastie and W. Stuetzle, "Principal curves," *J. Amer. Statist. Assoc.*, vol. 11, no. 1, pp. 193–213, 1999.
[20] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. New York: Wiley, 1973.
[21] K. Fukunaga, *Introduction to Statistical Pattern Recognition*. New York: Academic, 1990.
[22] H. Yu and J. Yang, "A direct LDA algorithm for high-dimension data with application to face recognition," *Pattern Recogn.*, vol. 34, pp. 2067–2070, 2001.
[23] L.-F. Chen, H.-Y. M. Liao, M.-T. Ko, J.-C. Lin, and G.-J. Yu, "A new lda-based face recognition system which can solve the small sample size problem," *Pattern Recogn.*, vol. 33, pp. 1713–1726, 2001.
[24] F. Fukunaga and W. Koontz, "Applications of the Karhunen-Loève expansion to feature selection and ordering," *IEEE Trans. Comput.*, vol. C–19, no. 5, pp. 311–318, 1970.
[25] X. Huo *et al.*, "Optimal Reduced-Rank Quadratic Classifiers Using the Fukunaga–Koontz Transform, With Applications to Automated Target Recognition," Tech. Rep., Stanford Univ., Georgia Inst. Technol., Lockheed Martin Co., Mar. 2003.
[26] X. S. Zhou and T. S. Huang, "Small sample learning during multimedia retrieval using biasmap," in *Proc. IEEE Conf. Comput. Vision Pattern Recogn.*, 2001.
[27] K. Fukumizu, F. R. Bach, and M. I. Jordan, "Dimensionality reduction for supervised learning with reproducing kernel hilbert spaces," *J. Machine Learning Res.*, vol. 5, pp. 73–99, 2004.
[28] T. Hastie and R. Tibshirani, "Discriminant adaptive nearest neighbor classification and regression," in *Advances in Neural Information Processing Systems*, D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, Eds. Cambridge, MA: MIT Press, 1996, vol. 8, pp. 409–415.
[29] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Sci.*, vol. 290, pp. 2323–2326, 2000.
[30] S. J. Raudys and A. K. Jain, "Small sample size effects in statistical pattern recognition: recommendations for practitioners," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 13, no. 3, pp. 252–264, Mar. 1991.
[31] R. Beveridge, K. She, B. Draper, and G. Givens, "A nonparametric statistical comparison of principal component and linear discriminant subspaces for face recognition," in *Proc. IEEE Computer Soc. Conf. Comput. Vision Pattern Recogn.*, vol. 1, 2001, pp. 535–542.
[32] F. Cucker and S. Smale, "On the mathematical foundations of learning," *Bull. Amer. Math. Soc.*, vol. 39, no. 1, pp. 1–49, 2001.
[33] P. Zhang and J. Peng, "Svm vs regularized least squares classification," in *Proc. IEEE Int. Conf. Pattern Recogn.*, vol. 1, 2004, pp. 176–179.
[34] ——, "Parzen Windows, Regularized Least Squares and Discriminant Analysis," Tech. Rep., Tulane Univ., New Orleans, LA, 2004.



Peng Zhang (S'02) received the B.S. degree in electronics engineering from Peking University, Beijing, China, in 1997 and the M.E. degree in electrical engineering and computer science from Institute of Acoustics, Chinese Academy of Sciences, Beijing, in 2000. He is currently pursuing the Ph.D. degree with the Department of Electrical Engineering and Computer Science at Tulane University, New Orleans, LA.

From September 2000 to December 2002, he was with the Department of Electrical Engineering, University of New Orleans. His research interests include machine learning, statistical learning theory, pattern recognition, data mining, statistical signal processing, and information fusion.

Jing Peng (M'95) received the B.S. degree in computer science from the Beijing Institute of Aeronautics and Astronautics, Beijing, China, the M.A. degree in computer science from Brandeis University, Waltham, MA, and the Ph.D. degree in computer science from Northeastern University, Boston, MA in 1994.

Since 2001, he has been an Assistant Professor of electrical engineering and computer science at Tulane University, New Orleans, LA. Previously, he has been on the faculty of computer science at Oklahoma State University, Stillwater. He was a research scientist with the Center for Research in Intelligent Systems, University of California, Riverside. His research interests include machine learning, classification, image databases, and learning and vision applications.

Dr. Peng has served as the Program Co-Chair for the IEEE Workshop in Computer Vision and Pattern Recognition.

Carlotta Domeniconi received the Laurea degree in computer science from the University of Milano, Milano, Italy, in 1992, the M.S. degree in information and communication technologies from the International Institute for Advanced Scientific Studies, Salerno, Italy, in 1997, and the Ph.D. degree in computer science from the University of California, Riverside in 2002.

She is currently an Assistant Professor with the Information and Software Engineering Department, George Mason University, Fairfax, VA. Her research interests include machine learning, pattern recognition, data mining, and bioinformatics.

Dr. Domeniconi received the 2004 Ralph E. Powe Junior Faculty Enhancement Award.