Kernel Pooled Local Subspaces for Classification

Peng Zhang & Jing Peng
Electrical Engr. & Comp. Sci. Dept.
Tulane University
New Orleans, LA 70118
{zhangp,jp}@eecs.tulane.edu

Carlotta Domeniconi
ISE Department
George Mason University
Fairfax, VA 22030
carlotta@ise.gmu.edu

Abstract

We study the use of kernel subspace methods for learning low-dimensional representations for classification. We propose a kernel pooled local discriminant subspace method and compare it against several competing techniques: Principal Component Analysis (PCA), Kernel PCA (KPCA), and linear local pooling in classification problems. We evaluate the classification performance of the nearest-neighbor rule with each subspace representation. The experimental results demonstrate the effectiveness and performance superiority of the kernel pooled subspace method over competing methods such as PCA and KPCA in some classification problems.

1. Introduction

Subspace analysis methods such as PCA and KPCA play an important role in computer vision research. In visual modeling and recognition 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 [5], target detection [3, 15, 16], visual learning [14], face recognition [16, 18], Linear Discriminant Analysis [10], and Fisherfaces [1].

Subspace analysis often significantly simplifies tasks such as regression, classification, and density estimation by computing low-dimensional subspaces having statistically uncorrelated or independent variables. PCA [12] is a prime example that employs eigenvector-based techniques to reduce dimensionality and extract features. Independent Component Analysis (ICA) [4] is another example 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 [8, 13] and KPCA [17] extend these linear techniques in a nonlinear fashion.

In this paper we propose a kernel pooled local subspace method for learning low-dimensional representations for classification. We perform a nonlinear global dimensionality reduction by pooling local discriminant dimension information in feature space and applying the kernel trick [7]. The resulting subspaces are nonlinear, discriminant and compacted, whereby better classification performance greater computational efficiency can be expected.

2 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.

2.1 PCA

In PCA [12], 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. The 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 for projecting the data: $\mathbf{y} = Q_m^t \mathbf{x}$ are identified, where Q_m is a submatrix of Q representing the principal eigenvectors. PCA represents a global linear projection of the data onto the lower-dimensional subspace corresponding to the maximal eigenvalues.

2.2 Kernel PCA

Kernel PCA is a nonlinear version of principal component analysis [17]. KPCA applies a nonlinear mapping to the input $\phi(\mathbf{x}) : \Re^q \rightarrow \Re^N$ and then solve for a linear PCA in the induced feature space $\Re^N$, where $N \gg q$ and possibly infinite. In KPCA, the mapping ϕ is made implicit by the use of kernel functions satisfying Mercer's theorem [6]

$$k(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x}) \cdot \phi(\mathbf{y}). \tag{1}$$

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.

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

3 Kernel Pooled local Subspace Method

3.1 Pooled Local Subspace Method

Hastie and Tibshrani [11] propose a global dimension reduction technique by pooling local dimension information. The idea is to compute a subspace corresponding to the eigenvectors of the average local between-sum of squares matrices. More specifically, let $\bar{\mathbf{x}}_j^{(i)}$ be the mean of class j samples in the a neighborhood of the ith training point, and $\bar{\mathbf{x}}^{(i)}$ be the overall mean in the same neighborhood. Then the local between-sum of squares matrix at the ith training point is

$$B_i = \frac{1}{J} \sum_{j=1}^{J} (\bar{\mathbf{x}}_j^{(i)} - \bar{\mathbf{x}}^{(i)})(\bar{\mathbf{x}}_j^{(i)} - \bar{\mathbf{x}}^{(i)})^t \tag{2}$$

The pooled local subspace method seeks a discriminant subspace that is close to all of B_is.

It turns out, as shown in [11], that this subspace is spanned by the largest eigenvectors of the average between-sum of squares matrix

$$B = \frac{1}{l} \sum_{i=1}^{l} B_i \tag{3}$$

where l denotes the number of training samples. The experimental results presented in [11] show that the pooled local subspace method is very promising.

3.2 Kernel Pooled Local Subspace Analysis

We now show how to compute a nonlinear pooled local discriminant subspace by using the kernel trick [7]. Let $\phi : \mathbf{x} \rightarrow \phi(\mathbf{x})$ be the nonlinear mapping from $\Re^q$ to $\Re^N$. Also, let $\bar{\phi}_j(\mathbf{x}^{(i)}) = \frac{1}{l_{ij}} \sum_{y_k=j} \phi(\mathbf{x}_k^{(i)})$ be the mean of class j samples in a neighborhood of the ith training point in the feature space, and $\bar{\phi}(\mathbf{x}^{(i)}) = \frac{1}{l_i} \sum_{k=1}^{l_i} \phi(\mathbf{x}_k^{(i)})$ be the overall mean in the same neighborhood, where l_j represents the number of class j training samples in the neighborhood of the ith training point, and l_i the total number of training samples in the neighborhood. Then the local between-sum of squares matrix at the ith training point in the feature space is

$$B_i^{\phi} = \frac{1}{J} \sum_{j=1}^{J} (\bar{\phi}_j(\mathbf{x}^{(i)}) - \bar{\phi}(\mathbf{x}^{(i)}))(\bar{\phi}_j(\mathbf{x}^{(i)}) - \bar{\phi}(\mathbf{x}^{(i)}))^t \tag{4}$$

The pooled local subspace method seeks a discriminant subspace that is close to all of B_is. The average between-sum of squares matrix B in the feature space is

$$\begin{aligned} B^{\phi} &= \frac{1}{l} \sum_{i=1}^{l} B_i^{\phi} \\ &= \frac{1}{lJ} \sum_{i=1}^{l} \sum_{j=1}^{J} \tilde{\phi}_j(\mathbf{x}^{(i)}) \tilde{\phi}_j(\mathbf{x}^{(i)})^t \end{aligned} \tag{5}$$

where $\tilde{\phi}_j(\mathbf{x}^{(i)}) = \bar{\phi}_j(\mathbf{x}^{(i)}) - \bar{\phi}(\mathbf{x}^{(i)})$.

Similar to KPCA [17], 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 α_i $(i = 1, \cdots, l)$ such that

$$\mathbf{v} = \sum_{i=1}^{l} \alpha_i \phi(\mathbf{x}_i). \tag{6}$$

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

$$\lambda(\phi(\mathbf{x}_k) \cdot \mathbf{v}) = (\phi(\mathbf{x}_k) \cdot B^{\phi} \mathbf{v}) \tag{7}$$

Substituting (6) and (5) into (7), we obtain, the left hand side of (7)

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

For the right hand side of (7), we have

$$
\begin{aligned}
& lJ(\phi(\mathbf{x}_k)\cdot B^{\phi}\mathbf{v}) \\
&= \phi(\mathbf{x}_k)\cdot\sum_{i=1}^{l}\sum_{j=1}^{J}\tilde{\phi}_j(\mathbf{x}^{(i)})\tilde{\phi}_j(\mathbf{x}^{(i)})^t\sum_{m=1}^{l}\alpha_m\phi(\mathbf{x}_m) \\
&= \phi(\mathbf{x}_k)\cdot\sum_{i=1}^{l}\sum_{j=1}^{J}\tilde{\phi}_j(\mathbf{x}^{(i)})\tilde{\phi}_j(\mathbf{x}^{(i)})^t(\phi(\mathbf{x}_1)\cdots\phi(\mathbf{x}_l))\alpha \\
&= \phi(\mathbf{x}_k)\cdot\sum_{i=1}^{l}\sum_{j=1}^{J}\tilde{\phi}_j(\mathbf{x}^{(i)})[\tilde{\phi}_j(\mathbf{x}^{(i)})\cdot\phi(\mathbf{x}_1)\cdots \\
&\qquad \tilde{\phi}_j(\mathbf{x}^{(i)})\cdot\phi(\mathbf{x}_l)]\alpha \\
&= \sum_{j=1}^{J}[\phi(\mathbf{x}_k)\cdot\tilde{\phi}_j(\mathbf{x}^{(1)})\cdots\phi(\mathbf{x}_k)\cdot\tilde{\phi}_j(\mathbf{x}^{(l)})]K_j\alpha
\end{aligned}
$$

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

$$
K_j = \begin{pmatrix} \tilde{\phi}_j(\mathbf{x}^{(1)})\cdot\phi(\mathbf{x}_1) & \cdots & \tilde{\phi}_j(\mathbf{x}^{(1)})\cdot\phi(\mathbf{x}_l) \\ \cdots & \cdots & \cdots \\ \tilde{\phi}_j(\mathbf{x}^{(l)})\cdot\phi(\mathbf{x}_1) & \cdots & \tilde{\phi}_j(\mathbf{x}^{(l)})\cdot\phi(\mathbf{x}_l) \end{pmatrix}.
$$

Define

$$K = [k_{ij}] = [\phi(\mathbf{x}_i)\cdot\phi(\mathbf{x}_j)] \tag{8}$$

and

$$\tilde{K} = \sum_{j=1}^{J} K_j^t K_j \tag{9}$$

We obtain

$$(lJ)\lambda K\alpha = \tilde{K}\alpha \tag{10}$$

which is a generalized eigenvector problem [9]. By solving (10), the ith principal component u_i of $\mathbf{x}$ can be calculated according to

$$u_i = \mathbf{v}_i\cdot\phi(\mathbf{x}) = \sum_{k=1}^{l}\alpha_k^i k(\mathbf{x},\mathbf{x}_k) \tag{11}$$

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

3.3 Centering in Feature Space

Computing both K (8) and $\tilde{K}$ requires centering the data. Similar to KPCA [17], the data can be centered in the feature space as follows. Let

$$\Phi(\mathbf{x}_k) = \phi(\mathbf{x}_k) - \frac{1}{l}\sum_{i=1}^{l}\phi(\mathbf{x}_i) \tag{12}$$

and $\mathbf{1}_l$ as a $l \times l$ matrix with entries $(\mathbf{1}_N)_{ij} := 1/l$. Then the Gram matrix K on the left side of (10) becomes

$$K^c = K - \mathbf{1}_l K - K\mathbf{1}_l + \mathbf{1}_l K\mathbf{1}_l \tag{13}$$

On the other hand,

$$
\begin{aligned}
\tilde{\Phi}_j(\mathbf{x}^{(i)}) &= \bar{\Phi}_j(\mathbf{x}^{(i)}) - \bar{\Phi}(\mathbf{x}^{(i)}) \\
&= \frac{1}{l_{ij}}\sum_{y_k=j}(\phi(\mathbf{x}_k^{(i)}) - \frac{1}{l}\sum_{k=1}^{l}\phi(\mathbf{x}_k)) \\
&- (\frac{1}{l_i}\sum_{k=1}^{l_i}(\phi(\mathbf{x}_k^{(i)}) - \frac{1}{l}\sum_{m=1}^{l}\phi(\mathbf{x}_m)) \\
&= \bar{\phi}_j(\mathbf{x}^{(i)}) - \bar{\phi}(\mathbf{x}^{(i)}).
\end{aligned} \tag{14}
$$

It follows that the centered matrix $\tilde{K}$ becomes

$$\tilde{K}^c = \sum_{j=1}^{J}(K_j^c)^t K_j^c, \tag{15}$$

where

$$K_j^c = K_j - K_j\mathbf{1}_l. \tag{16}$$

Thus the generalized eigenvector problem (10) becomes

$$(lJ)\lambda K^c\alpha = \tilde{K}^c\alpha \tag{17}$$

4 Kernel Pooled Local Subspace Algorithm

4.1 Details of the Implementation

Calculating K_j^c is the key step in the implementation. Here we show it is done. Notice that

$$
\begin{aligned}
& (\bar{\phi}_j(\mathbf{x}^{(i)}) - \bar{\phi}(\mathbf{x}^{(i)}))\cdot\phi(\mathbf{x}_k) \\
&= \frac{1}{l_{ij}}\sum_{y_k=j}\phi(\mathbf{x}_k^{(i)}\cdot\phi(\mathbf{x}_k) - \frac{1}{l_i}\sum_{k=1}^{l_i}\phi(\mathbf{x}_k^{(i)}\cdot\phi(\mathbf{x}_k) \\
&= \mathbf{1}_{l_{ij}}^t K_{j,k}^{(i)} - \mathbf{1}_{l_i}^t K_k^{(i)}
\end{aligned}
$$

where $\mathbf{1}_{l_{ij}} = (\frac{1}{l_{ij}}, \cdots, \frac{1}{l_{ij}})^t$, $\mathbf{1}_{l_i} = (\frac{1}{l_i}, \cdots, \frac{1}{l_i})^t$, $K_{j,k}^{(i)} = (k(\mathbf{x}_{j,1}^{(i)}, \mathbf{x}_k), \cdots, k(\mathbf{x}_{j,l_{ij}}^{(i)}, \mathbf{x}_k))^t$, and $K_k^{(i)} = (k(\mathbf{x}_{i,1}^{(i)}, \mathbf{x}_k), \cdots, k(\mathbf{x}_{i,l_i}^{(i)}, \mathbf{x}_k))^t$, $k = 1, \cdots, l$. Thus, each row of K_j^c can be calculated according to

$$\mathbf{1}_{l_{ij}}K_j^{(i)} - \mathbf{1}_{l_i}K^{(i)} \tag{18}$$

where $K_j^{(i)} = (K_{j,1}^{(i)}, \cdots, K_{j,l}^{(i)})$ is a $l_{ij} \times l$ matrix, and $K^{(i)} = (K_1^{(i)}, \cdots, K_l^{(i)})$ is a $l_i \times l$ matrix.

4.2 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_i used to determine the local neighborhood for pooling, and γ in a Gaussian kernel or d in a polynomial kernel. In the experiments reported below, these procedural parameters are determined through cross-validation.

Algorithm 1 (KPooLS algorithm)

1. Calculate K (8).

2. Calculate K^c (13).

3. For each training point $\phi(\mathbf{x}_i)$ ($i = 1, \cdots, l$), find l_i nearest neighbors, and calculate (18).

4. Pool all $\mathbf{1}_{l_{ij}} K_j^{(i)} - \mathbf{1}_{l_i} K^{(i)}$ to calculate K_j^c.

5. Calculate $\tilde{K}^c$ by (16) and (15).

6. Solve the generalized eigenvector equation (17).

5 Experimental Results

In the following we use two data sets to examine the classification performance of the nearest neighbor rule (3NN) with each subspace representation, i.e., PCA, KPCA, PCA plus linear local pooling (i.e., we first perform dimensionality reduction using PCA, followed by another reduction by linear local pooling) and the proposed KPooLS algorithm.

For each data set, we randomly select 60% of the data as training and remaining 40% as testing. This process is repeated 20 times and the average error distributions are reported. The dimensions of subspaces computed by each method are determined so that only eigenvectors remain whose eigenvalues are great than or equal to $0.01\lambda_{max}$. Also, we use Gaussian kernels for all nonlinear subspace methods and kernel parameters are determined through cross-validation.

5.1 Letter Data

The first experiment involves the letter image recognition data sets taken from UCI Machine Learning Repository. This data set consists of a large number of black-and-white rectangular pixel arrays as one of the 26 upper-case letters in the English alphabet. The characters are based on 20 Roman alphabet fonts. They represent five different stroke styles and six different letter styles. Each letter is randomly distorted through a quadratic transformation to produce a set of 20,000 unique letter images that are then converted into 16 primitive numerical features. Basically, these numerical features are statistical moments and edge counts. For this experiment we randomly select 20 letter images from letters K to T (10 classes). Thus, the data set consists of 200 letter images.

5.1.1 Results

Figure 1 shows the error distributions of the four methods on the letter image data. Each distribution has lines at the lower quartile, median, and upper quartile values. The whiskers are lines extending from each end of the distribution to show the extent of the rest of the errors. Outliers are errors with values beyond the ends of the whiskers. If there is no data outside the whisker, a dot is placed at the bottom whisker.

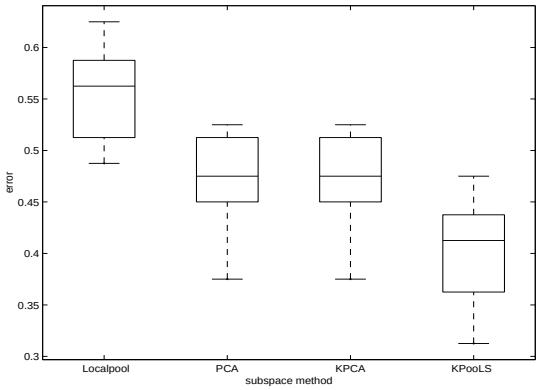

Figure 1. Classification performance achieved by the nearest neighbor rule with each subspace method.

The error distributions clearly are in favor of KPooLS. Furthermore, KPooLS uses much more compacted subspaces to achieve the error distributions. Figure 2 shows normalized eigenvalues calculated by each method averaged over 20 runs. The eigenvalues of the principal spaces computed by KPooLS decrease rapidly, demonstrating its ability to choose discriminant principal components. On average, the subspaces computed by KPooLS are represented only by 11 principal modes. In contrast, KPCA uses 117 and PCA 16. Linear local pooling uses a similar number of principal modes to that of KPooLS to represent the subspaces. However, its performance is significantly worse that KPooLS.

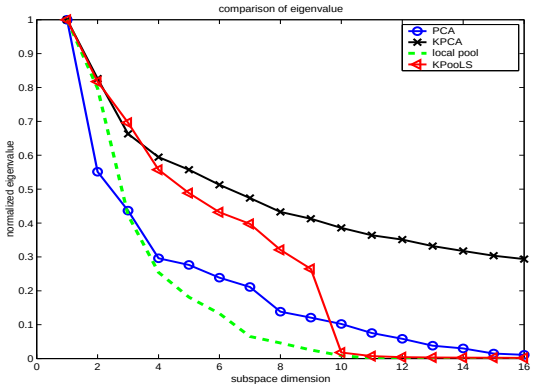

Figure 2. Eigenvalues computed by each subspace method in the letter image data set.

5.2 Glass Data

The second experiment involves the Glass data, again taken from UCI Machine Learning Repository. This data set consists of $q = 9$ chemical attributes measured for each of $N = 214$ data of six classes. The problem is to classify each test point in the 9-dimensional space to its correct class.

5.2.1 Results

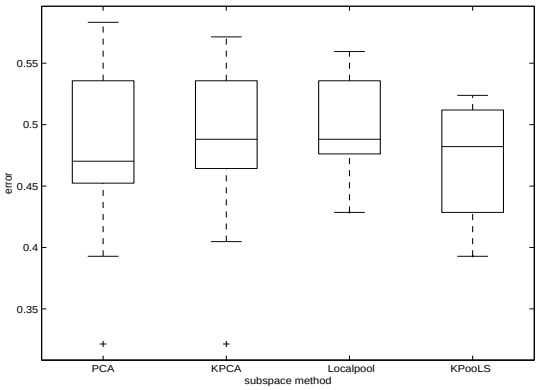

Figure 3. Classification performance achieved by the nearest neighbor rule with each subspace method.

Figure 3 shows the error distributions of the four methods on the Glass data. The four methods registered similar error distributions on the Glass data. Again KPooLS and linear local pooling use more compacted subspaces to obtain the error distributions. Figure 4 shows normalized eigenvalues calculated by each method over 20 runs. The eigenvalues of the principal spaces computed by both KPooLS and local pooling decrease rapidly to five principal modes–the minimal number of dimensions required to correctly classify the Glass data. In contrast, KPCA uses 26 and PCA 8.

5.3 Cat and Dog Image Data

In this experiment, the data set is composed of two hundred images of cat faces 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 Figure 5.

The performance task is to distinguish cats from dogs. For each query image the nearest gallery image to each is returned as its match. A match is said to be correct if it of

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

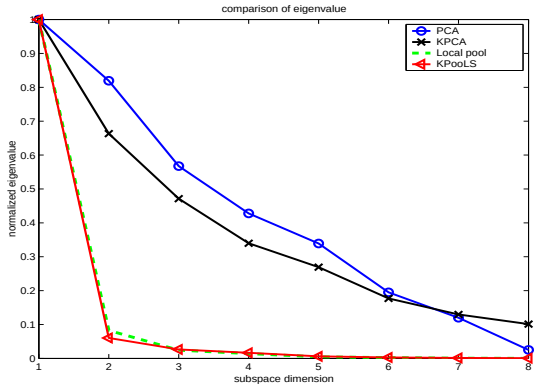

Figure 4. Eigenvalues computed by each subspace method in the Glass data set.

the same species as the query. For the subspace methods, the features are the 4096 pixel values. As reported in [2], only 5% of the pixels carry meaningful information.

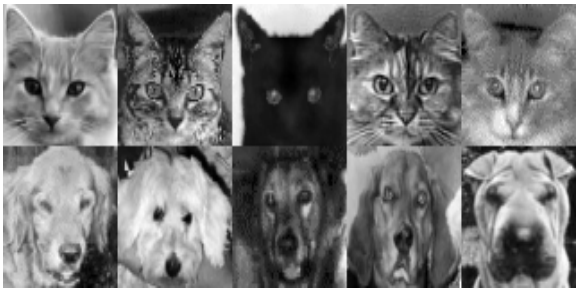

Figure 5. Sample cat and dog images.

5.3.1 Results

Figure 6 shows the error distributions obtained by each method on the cat and dog image data. The results again are in favor of KPooLS. In this case, PCA performed significantly worse, as expected.

Figure 7 shows normalized eigenvalues calculated by each method. The eigenvalues of the principal spaces computed by KPooLS again decrease rapidly, followed by KPCA and PCA. On average, the subspaces computed by KPooLS are represented by three principal components. In contrast, both KPCA and PCA (49) use 119 principal components to represent the subspace. This number (119) is exactly the number of training images used in this experiment. It is rather surprising to see that KPCA fails to achieve significant dimensionality reduction. However, it does perform significantly better than PCA due to ite nonlinearity. Similar to the letter data experiment, PCA with local pooling uses basically the same number of principal components as

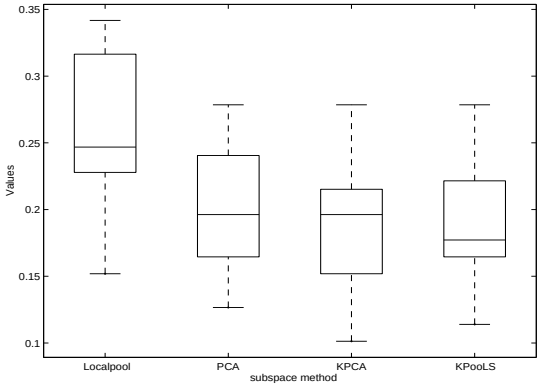

Figure 6. Classification performance achieved by the nearest neighbor rule with each subspace method on the cat and dog data.

that of KPooLS to represent the subspaces.

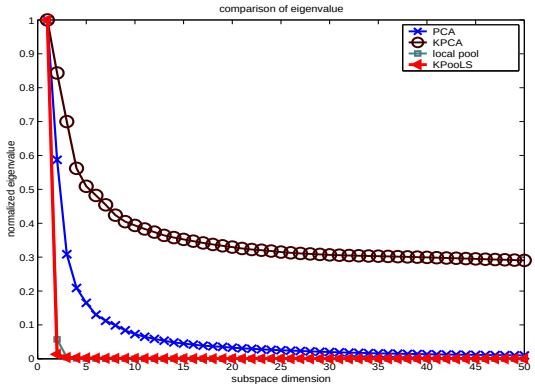

Figure 7. Eigenvalues computed by each subspace method on the cat and dog image data set.

6 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 applying the kernel trick. The resulting Subspaces are nonlinear, discriminant and compacted, whereby better classification performance and greater computational efficiency can be achieved. The experimental results show that the KPooLS algorithm can learn discriminant subspaces that are much more compacted than that computed by KPCA and other competing subpace methods in some classification problems.

References

[1] V. Belhumeur, J. Hespanha, and D. Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 19(7):711–720, 1997.

[2] J. Bins and B. Draper. Feature selection from huge feature sets. In *Proc. Int'l Conf. on Computer Vision*, 2001.

[3] M. Burl, U. Fayyad, P. P. P. Smyth, and M. Burl. Automating the hunt for volcanos on venus. In *IEEE Conf. Computer Vision and Pattern Recognition*, 1994.

[4] P. Comon. Independent, component analysis - a new concept? *Signal Processing*, 36:287–314, 1994.

[5] T. Cootes and C. Taylor. Active shape models: Smart snakes. In *Proc. British Machine Vision Conf.*, pages 9–18, 1992.

[6] R. Courant and D. Hilbert, editors. *Methods of Mathematical Physiacs, vol. 1*. Interscience, New York, 1953.

[7] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, Cambridge, UK, 2000.

[8] D. DeMers and G. Cottrell. Nonlinear dimensionality reduction. In *Advances in Neural Information Processing Systems*, volume 5. The MIT Press, 1993.

[9] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, Inc., 1973.

[10] K. Etemad and R. Chellappa. Discriminant analysis for recognition of human faces. In *Proc. Int'l Conf. Acoustics, Speech, and Signal Processing*, pages 2148–2151, 1996.

[11] T. Hastie and R. Tibshirani. Discriminant adaptive nearest neighbor classification and regression. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8, pages 409–415. The MIT Press, 1996.

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

[13] M. Kramer. Nonlinear principal components analysis using autoassociative neural networks. *Am. Inst. Chemical Eng. J.*, 32(2):233–234, 1991.

[14] B. Moghaddam and A. Pentland. Probabilistic visual learning for object representation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 19(7):696–710, 1997.

[15] S. Nayar, S. Baker, and H. Murase. Parametric feature detection. In *IEEE Conf. Computer Vision and Pattern Recognition*, pages 471–477, 1994.

[16] A. Pentland, B. Moghaddam, and T. Starner. View-based and modular eigenspaces for face recognition. In *IEEE Conf. Computer Vision and Pattern Recognition*, 1994.

[17] B. Scholkopf, A. Smola, and K.-R. Muller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.

[18] M. Turk and A. Pentland. Eigenfaces for recognition. *Cognitive Neuroscience*, 3(1), 1991.