Kernel Principal Component Analysis and its Applications in Face Recognition and Active Shape Models

Article · July 2012

Source: arXiv

1 author:

Quan Wang

Google Inc.

43 PUBLICATIONS   **1,183** CITATIONS

Kernel Principal Component Analysis and its Applications in Face Recognition and Active Shape Models

Quan Wang   WANGQ10@RPI.EDU

Rensselaer Polytechnic Institute, 110 Eighth Street, Troy, NY 12180 USA

Abstract

Principal component analysis (PCA) is a popular tool for linear dimensionality reduction and feature extraction. Kernel PCA is the nonlinear form of PCA, which is promising in exposing the more complicated correlation between original high-dimensional features. In this paper, we first talk about the basic ideas of PCA and kernel PCA, and then focus on the reconstruction of pre-images for kernel PCA. We also give an introduction on how PCA is used in active shape models (ASMs), and discuss how kernel PCA can be applied to improve traditional ASMs. Then we show some experiment results to compare the performance of kernel PCA and traditional PCA for pattern classification. We also implement the kernel PCA-based ASMs, and use it to construct human face models.

1. Introduction

In this section, we briefly review the principal component analysis method and the active shape models.

1.1. Principal Component Analysis

Principal component analysis, or PCA, is a very popular technique for dimensionality reduction and feature extraction. PCA attempts to find a linear subspace of lower dimensionality of the original feature space in which the new features have the largest variance (Bishop, 2006).

Consider a dataset $\{\mathbf{x}_n\}$ where $n = 1, 2, \ldots, N$, and $\mathbf{x}_n$ is a D-dimensional vector. Now we want to project the data onto an M-dimensional subspace where $M < D$. We assume the projection is denoted as $\mathbf{y} = \mathbf{A}\mathbf{x}$ where

This work originally appears as the final project of Professor Qiang Ji's course *Pattern Recognition* at RPI, Troy, NY, USA, 2011. Copyright 2011 by Quan Wang.

$\mathbf{A} = [\mathbf{u}_1^{\mathrm{T}}, \ldots, \mathbf{u}_M^{\mathrm{T}}]$ and $\mathbf{u}_i^{\mathrm{T}}\mathbf{u}_i = 1$ for $i = 1, 2, \ldots, M$. We want to maximize the variance of $\mathbf{y}_n$, which is the trace of the covariance matrix of $\mathbf{y}_n$. Thus, we want to find

$$\max_{\mathbf{A}} \operatorname{tr}(\mathbf{S_y}), \tag{1}$$

where

$$\mathbf{S_y} = \frac{1}{N} \sum_{n=1}^{N} (\mathbf{y}_n - \bar{\mathbf{y}})(\mathbf{y}_n - \bar{\mathbf{y}})^{\mathrm{T}}, \tag{2}$$

and

$$\bar{\mathbf{y}} = \frac{1}{N} \sum_{n=1}^{N} \mathbf{x}_n. \tag{3}$$

Let $\mathbf{S_x}$ be the covariance matrix of $\mathbf{x}_n$. Since $\operatorname{tr}(\mathbf{S_y}) = \operatorname{tr}(\mathbf{A}\mathbf{S_x}\mathbf{A}^{\mathrm{T}})$, by using the Lagrangian multiplier and taking the derivative, we get

$$\mathbf{S_x}\mathbf{u}_i = \lambda_i \mathbf{u}_i, \tag{4}$$

which means that $\mathbf{u}_i$ is an eigenvector of $\mathbf{S_x}$. Now $\mathbf{x}_n$ can be represented as

$$\mathbf{x}_n = \sum_{i=1}^{D} \left(\mathbf{x}_n^{\mathrm{T}}\mathbf{u}_i\right)\mathbf{u}_i. \tag{5}$$

$\mathbf{x}_n$ can be also approximated by

$$\widetilde{\mathbf{x}}_n = \sum_{i=1}^{M} \left(\mathbf{x}_n^{\mathrm{T}}\mathbf{u}_i\right)\mathbf{u}_i, \tag{6}$$

where $\mathbf{u}_i$ is the eigenvector of $\mathbf{S_x}$ corresponding to the ith largest eigenvalue.

1.2. Active Shape Models

The active shape model, or ASM, is one of the most popular top-down computer vision approaches. It is designed to discover the hidden deformation patterns of the shape of object or region of interest (ROI), and to locate the object or ROI in new images. ASMs use the point distribution model (PDM) to describe the shape (Cootes, 1995). If a shape consists of n points,

and the coordinates of the ith point are (x_i, y_i), then the shape can be represented as a vector

$$\mathbf{x} = [x_1, y_1, \ldots, x_n, y_n]^{\mathrm{T}}. \tag{7}$$

To simplify the problem, we now assume that all shapes have already been aligned. Otherwise, a rotation by θ, a scaling by s and a translation by $\mathbf{t}$ should be applied to $\mathbf{x}$. Given N aligned shapes as training data, the mean shape can be calculated by

$$\bar{\mathbf{x}} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}_i. \tag{8}$$

For each shape $\mathbf{x}_i$ in the training set, its deviation from the mean $\bar{\mathbf{x}}$ is

$$\mathrm{d}\mathbf{x}_i = \mathbf{x}_i - \bar{\mathbf{x}}. \tag{9}$$

Then the $2n \times 2n$ covariance matrix $\mathbf{S}$ can be calculated by

$$\mathbf{S} = \frac{1}{N} \sum_{i=1}^{N} \mathrm{d}\mathbf{x}_i \mathrm{d}\mathbf{x}_i^{\mathrm{T}}. \tag{10}$$

Now we perform the PCA on $\mathbf{S}$. Assume that

$$\mathbf{S}\mathbf{p}_k = \lambda_k \mathbf{p}_k, \tag{11}$$

where $\mathbf{p}_k$ is the eigenvector of $\mathbf{S}$ corresponding to the kth largest eigenvalue, and

$$\mathbf{p}_k^{\mathrm{T}} \mathbf{p}_k = 1. \tag{12}$$

Let $\mathbf{P}$ be the matrix of the first t eigenvectors:

$$\mathbf{P} = [\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_t]. \tag{13}$$

Then we can approximate a shape in the training set as

$$\mathbf{x} = \bar{\mathbf{x}} + \mathbf{P}\mathbf{b}, \tag{14}$$

where $\mathbf{b} = [b_1, b_2, \ldots, b_t]^{\mathrm{T}}$ is the vector of weights for different PCA features. By varying the parameters b_k, we can generate new examples of the shape. We can also limit b_k to constrain the deformation patterns of the shape. Typical limits are

$$-3\sqrt{\lambda_k} \leq b_k \leq 3\sqrt{\lambda_k}, \tag{15}$$

where $k = 1, 2, \ldots, t$. An important issue of AMSs is using point distribution models to search for the shape in images. We do not talk about this problem in our paper, and focus on the statistic model.

2. Kernel PCA

Traditional PCA only allows linear dimensionality reduction. However, if the data has more complicated structures which cannot be simplified in a linear subspace, traditional PCA will become invalid. Fortunately, kernel PCA allows us to generalize traditional PCA to nonlinear dimensionality reduction (Scholkopf et al., 1999).

2.1. Constructing the Kernel Matrix

Assume we have a nonlinear transformation $\phi(\mathbf{x})$ from the original D-dimensional feature space to an M-dimensional feature space, where usually $M \gg D$. Then each data point $\mathbf{x}_n$ is projected to a point $\phi(\mathbf{x}_n)$. We can perform the traditional PCA in the new feature space, but this might be extremely costly. Thus kernel methods are used to simplify the computation (Schlkopf et al., 1996).

First we assume that the projected new features have zero mean:

$$\sum_n \phi(\mathbf{x}_n) = \mathbf{0}. \tag{16}$$

The covariance matrix of the projected features is $M \times M$, calculated by

$$\mathbf{C} = \frac{1}{N} \sum_{n=1}^{N} \phi(\mathbf{x}_n)\phi(\mathbf{x}_n)^{\mathrm{T}}, \tag{17}$$

and its eigenvalues and eigenvectors are given by

$$\mathbf{C}\mathbf{v}_i = \lambda_i \mathbf{v}_i, \tag{18}$$

where $i = 1, 2, \ldots, M$. From (17) and (18), we have

$$\frac{1}{N} \sum_{n=1}^{N} \phi(\mathbf{x}_n)\{\phi(\mathbf{x}_n)^{\mathrm{T}} \mathbf{v}_i\} = \lambda_i \mathbf{v}_i, \tag{19}$$

which can be written as

$$\mathbf{v}_i = \sum_{n=1}^{N} a_{in}\phi(\mathbf{x}_n). \tag{20}$$

Now by substituting $\mathbf{v}_i$ in (19) with (20), we have

$$\frac{1}{N} \sum_{n=1}^{N} \phi(\mathbf{x}_n)\phi(\mathbf{x}_n)^{\mathrm{T}} \sum_{m=1}^{N} a_{im}\phi(\mathbf{x}_m) = \lambda_i \sum_{n=1}^{N} a_{in}\phi(\mathbf{x}_n). \tag{21}$$

By defining the kernel function

$$k(\mathbf{x}_n, \mathbf{x}_m) = \phi(\mathbf{x}_n)^{\mathrm{T}} \phi(\mathbf{x}_m), \tag{22}$$

and multiplying both sides of Equation (21) by $\phi(\mathbf{x}_l)^{\mathrm{T}}$, we have

$$\frac{1}{N} \sum_{n=1}^{N} k(\mathbf{x}_l, \mathbf{x}_n) \sum_{m=1}^{N} a_{im}k(\mathbf{x}_n, \mathbf{x}_m) = \lambda_i \sum_{n=1}^{N} a_{in}k(\mathbf{x}_l, \mathbf{x}_n), \tag{23}$$

or the matrix notation

$$\mathbf{K}^2 \mathbf{a}_i = \lambda_i N \mathbf{K} \mathbf{a}_i, \tag{24}$$

where

$$\mathbf{K}_{n,m} = k(\mathbf{x}_n, \mathbf{x}_m), \tag{25}$$

and $\mathbf{a}_i$ is the N-dimensional column vector of a_{ni}. $\mathbf{a}_i$ can be solved by

$$\mathbf{K}\mathbf{a}_i = \lambda_i N \mathbf{a}_i, \qquad (26)$$

and the resulting kernel principal components can be calculated using

$$y_i(\mathbf{x}) = \phi(\mathbf{x})^{\mathrm{T}} \mathbf{v}_i = \sum_{n=1}^{N} a_{in} k(\mathbf{x}, \mathbf{x}_n). \qquad (27)$$

If the projected dataset $\{\phi(\mathbf{x}_n)\}$ does not have zero mean, we can use the Gram matrix $\widetilde{\mathbf{K}}$ to substitute the kernel matrix $\mathbf{K}$. The Gram matrix is given by

$$\widetilde{\mathbf{K}} = \mathbf{K} - \mathbf{1}_N \mathbf{K} - \mathbf{K} \mathbf{1}_N + \mathbf{1}_N \mathbf{K} \mathbf{1}_N, \qquad (28)$$

where $\mathbf{1}_N$ is the $N \times N$ matrix with all elements equal to $1/N$ (Bishop, 2006).

The power of kernel methods is that we do not have to compute $\phi(\mathbf{x}_n)$ explicitly. We can directly construct the kernel matrix from the training data set $\{\mathbf{x}_n\}$ (Weinberger et al., 2004). Two commonly used kernels are the polynomial kernel

$$k(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^{\mathrm{T}} \mathbf{y})^d, \qquad (29)$$

or

$$k(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^{\mathrm{T}} \mathbf{y} + c)^d, \qquad (30)$$

where $c > 0$ is a constant, and the Gaussian kernel

$$k(\mathbf{x}, \mathbf{y}) = \exp\left(-\|\mathbf{x} - \mathbf{y}\|^2 / 2\sigma^2\right) \qquad (31)$$

with parameter σ.

The standard steps of kernel PCA dimensionality reduction can be summarized as:

1. Construct the kernel matrix $\mathbf{K}$ from the training data set $\{\mathbf{x}_n\}$ using (25).

2. Compute the Gram matrix $\widetilde{\mathbf{K}}$ using (28).

3. Use (26) to solve for the vectors $\mathbf{a}_i$ (substitute $\mathbf{K}$ with $\widetilde{\mathbf{K}}$).

4. Compute the kernel principal components $y_i(\mathbf{x})$ using (27).

2.2. Reconstructing Pre-Images

So far, we have discussed how to generate new features $y_i(\mathbf{x})$ using kernel PCA. This is enough for applications such as feature extraction and pattern classification. However, for some other applications, we need to reconstruct the pre-images $\{\mathbf{x}_n\}$ from the kernel PCA

features $\{\mathbf{y}_n\}$. This is the case in active shape models, where we not only need to use PCA features to describe the deformation patterns, but also have to reconstruct the shapes from the PCA features (Romdhani et al., 1999; Twining & Taylor, 2001).

In traditional PCA, the pre-image $\mathbf{x}_n$ can simply be approximated by Equation (6). However, this cannot be achieved for kernel PCA (Bakr et al., 2004). Now we define a projection operator P_n which projects $\phi(\mathbf{x})$ to its approximation

$$P_n \phi(\mathbf{x}) = \sum_{i=1}^{n} y_i(\mathbf{x}) \mathbf{v}_i, \qquad (32)$$

where $\mathbf{v}_i$ is the eigenvector of the $\mathbf{C}$ matrix, which is define by Equation (17). If n is large enough, we have $P_n \phi(\mathbf{x}) \approx \phi(\mathbf{x})$. Since finding the exact pre-image $\mathbf{x}$ is difficult, we turn to find an approximation $\mathbf{z}$ such that

$$\phi(\mathbf{z}) \approx P_n \phi(\mathbf{x}). \qquad (33)$$

This can be approximated by minimizing

$$\rho(\mathbf{z}) = \|\phi(\mathbf{z}) - P_n \phi(\mathbf{x})\|^2. \qquad (34)$$

2.3. Pre-Images for Gaussian Kernels

There are some existing techniques to compute $\mathbf{z}$ for specific kernels (Mika et al., 1999). For a Gaussian kernel $k(\mathbf{x}, \mathbf{y}) = \exp\left(-\|\mathbf{x} - \mathbf{y}\|^2 / 2\sigma^2\right)$, $\mathbf{z}$ should satisfy

$$\mathbf{z} = \frac{\sum_{i=1}^{N} \gamma_i \exp\left(-\|\mathbf{z} - \mathbf{x}_i\|^2 / 2\sigma^2\right) \mathbf{x}_i}{\sum_{i=1}^{N} \gamma_i \exp\left(-\|\mathbf{z} - \mathbf{x}_i\|^2 / 2\sigma^2\right)}, \qquad (35)$$

where

$$\gamma_i = \sum_{k=1}^{n} y_k a_{ik}. \qquad (36)$$

We can use an iterative manner to compute $\mathbf{z}$:

$$\mathbf{z}_{t+1} = \frac{\sum_{i=1}^{N} \gamma_i \exp\left(-\|\mathbf{z}_t - \mathbf{x}_i\|^2 / 2\sigma^2\right) \mathbf{x}_i}{\sum_{i=1}^{N} \gamma_i \exp\left(-\|\mathbf{z}_t - \mathbf{x}_i\|^2 / 2\sigma^2\right)}. \qquad (37)$$

3. Experiments

In this section, we show the setup and results of our three experiments. The first two experiments are classification problems without pre-image reconstruction. The third experiment combines active shape models with kernel PCA, and involves the pre-image reconstruction algorithm.

3.1. Pattern Classification for Synthetic Data

Before we work on real data, we would like to generate some synthetic datasets and test our algorithm on them. In this paper, we use the two-concentric-spheres data.

3.1.1. Data Description

We assume that we have equal number or data points uniformly distributed on two concentric sphere surfaces. If N is the total number of all data points, then we have $N/2$ class 1 points on a sphere of radius r_1, and $N/2$ class 2 points on a sphere of radius r_2. In the spherical coordinate system, the inclination (polar angle) θ is uniformly distributed in $[0, \pi]$ and the azimuth (azimuthal angle) ϕ is uniformly distributed in $[0, 2\pi)$ for both classes. Our observations of the data points are the (x, y, z) coordinates in the Cartesian coordinate system, and all the three coordinates are perturbed by a Gaussian noise of standard deviation σ_{noise}. We set $N = 200$, $r_1 = 10$, $r_2 = 15$, $\sigma_{\text{noise}} = 0.1$, and give a 3D plot of the data in Figure 1.
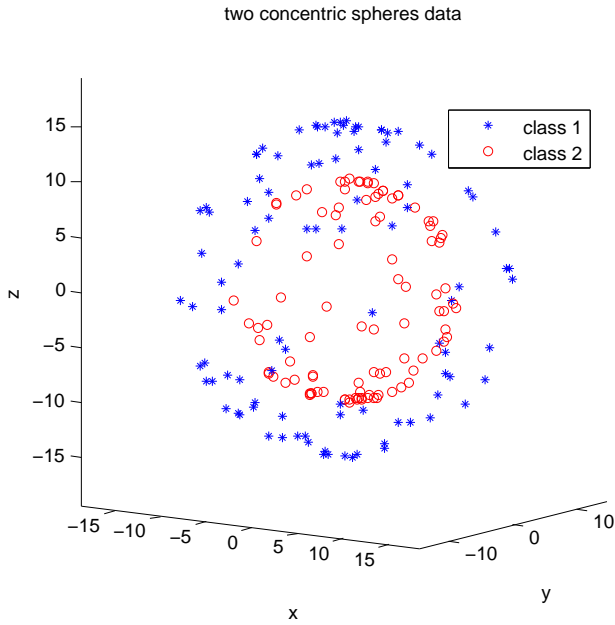

Figure 1. 3D plot of the two-concentric-spheres synthetic data.

3.1.2. PCA and Kernel PCA Results

To visualize our results, we project the original 3-dimensional data into a 2-dimensional feature space by using traditional PCA and kernel PCA respectively. For kernel PCA, we use a polynomial kernel with $d = 5$

and a Gaussian kernel with $\sigma = 20$. The results of traditional PCA, polynomial kernel PCA and Gaussian kernel PCA are given in Figure 2, Figure 3, and Figure 4 respectively.
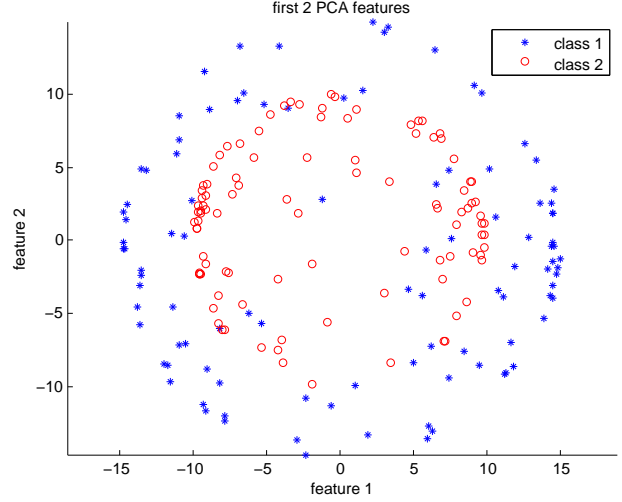

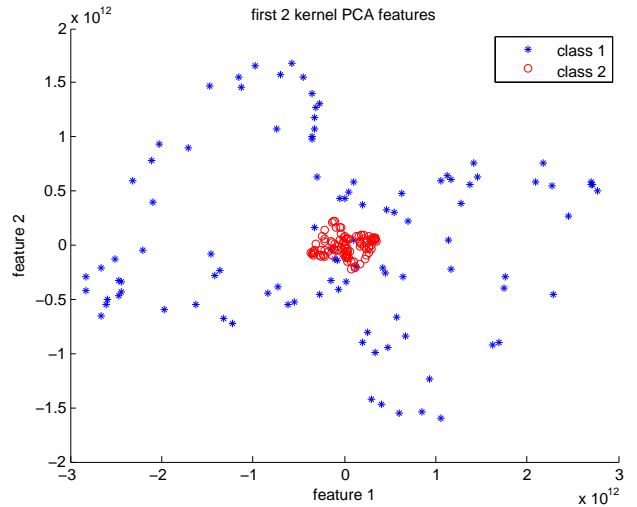*Figure 2.* Traditional PCA results for the two-concentric-spheres synthetic data.

Figure 3. Polynomial kernel PCA results for the two-concentric-spheres synthetic data with $d = 5$.

We note that here though we mark points in different classes with different colors, we are actually doing unsupervised learning. Neither PCA nor kernel PCA takes the class labels as their input.

In the results we can see that, traditional PCA does not reveal any structural information of the original
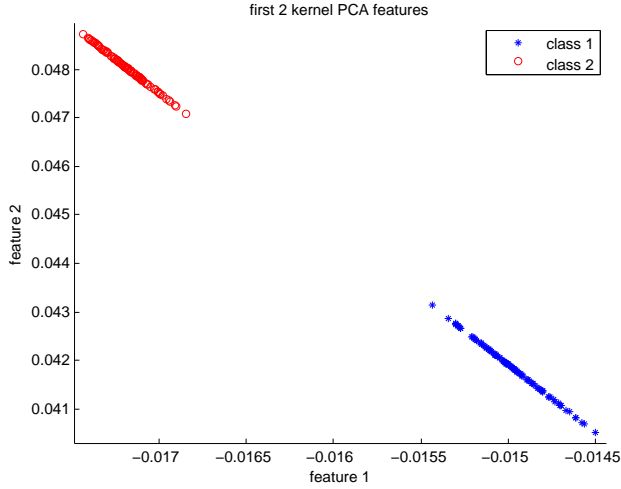
Figure 4. Gaussian kernel PCA results for the two-concentric-spheres synthetic data with $\sigma = 20$.

data. For polynomial kernel PCA, in the new feature space, class 1 data points are clustered while class 2 data points are scattered. But they are still not linearly separable. For Gaussian kernel PCA, the two classes are completely linearly separable, and both features can reveal the radius information of the original data.

3.2. Classification for Aligned Human Face Images

After we have tested our algorithm on synthetic data, we would like to use it for real data classification. Here we use PCA and kernel PCA to extract features from human face images, and use the simplest linear classifier for classification. Then we compare the error rates of using PCA and kernel PCA.

3.2.1. DATA DESCRIPTION

For this task, we use images from the Yale Face Database B (Georghiades et al., 2001), which contains 5760 single light source gray-level images of 10 subjects, each seen under 576 viewing conditions. We take 51 images of the first and third subject respectively as the training data, and 13 images of each of them as testing data. Then all the images are aligned, and each has 168×192 pixels. Sample images of the Yale Face Database B are shown in Figure 5.

3.2.2. CLASSIFICATION RESULTS

We use the 168×192 pixel intensities as the original features for each image, thus the original feature is
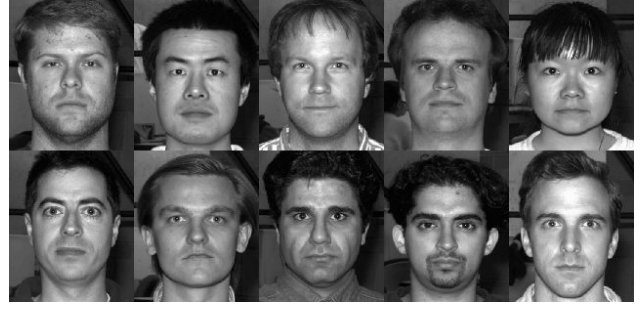

Figure 5. Sample images from the Yale Face Database B.

Table 1. Classification error rates on training data and testing data for traditional PCA and Gaussian kernel PCA with $\sigma = 45675$.

ERROR RATE	PCA	KERNEL PCA
TRAINING DATA	6.86%	5.88%
TESTING DATA	19.23%	11.54%

32256-dimensional. Then we use PCA and kernel PCA to extract the 10 most significant features from the training data, and record the eigenvectors.

For traditional PCA, only the eigenvectors are needed to extract features from testing data. For kernel PCA, both the eigenvectors and the training data are needed to extract features from testing data. Note that for traditional PCA, there are particular fast algorithms to compute the eigenvectors when the dimensionality is much larger than the number of data points (Bishop, 2006).

For kernel PCA, we use a Gaussian kernel with $\sigma = 45675$ (we will talk about how to select the parameters in Section 4). For classification, we use the simplest linear classifier. The training error rates and the testing error rates for traditional PCA and kernel PCA are given in Table 1. We can see that Gaussian kernel PCA achieves much lower error rates than traditional PCA.

3.3. Kernel PCA-Based Active Shape Models

In ASMs, the shape of an object is described with point distribution models, and traditional PCA is used to extract the principal deformation patterns from the shape vectors $\{\mathbf{x}_i\}$. If we use kernel PCA instead of traditional PCA here, it is promising that we will be able to discover more hidden deformation patterns.

3.3.1. Data Description

In our work, we use Tim Cootes' manually annotated points of 1521 human face images from the BioID database. For each face image, 20 feature points (landmarks) are labelled, as shown in Figure 6. Thus the original feature vector for each image is 40-dimensional (two coordinates for each landmark).
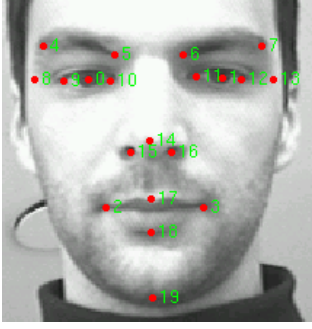

Figure 6. The 20 manually labelled points for an image (286×384) from BioID.

3.3.2. Experiment Results

In our work, we first normalize all the shape vectors by restricting both the x coordinates and y coordinates in the range $[0, 1]$. Then we perform PCA and Gaussian kernel PCA on the normalized shape vectors. For traditional PCA, the reconstruction of the shape is given by

$$\mathbf{x} = \bar{\mathbf{x}} + \mathbf{Pb}. \qquad (38)$$

For kernel PCA, the reconstruction of the shape is given by

$$\mathbf{z} = r(\mathbf{y}), \qquad (39)$$

where $r(\mathbf{y})$ denotes the reconstruction algorithm defined by (37).

For traditional PCA, we focus on studying the deformation pattern associated with each component of $\mathbf{b}$. That is to say, each time we uniformly select different values of b_k in $[-3\sqrt{\lambda_k}, 3\sqrt{\lambda_k}]$, and set $b_{k'} = 0$ for all $k' \neq k$. The effect of varying the first PCA feature and the second PCA feature are shown in Figure 7 and Figure 8 respectively. The face is drawn by using line segments to represent eye brows, eyes, the nose, using circles to represent eye balls, using a quadrilateral to represent the mouth, and fitting a parabola to represent the contour of the face.

For Gaussian kernel PCA, we set $\sigma = 0.7905$ for the Gaussian kernel (the parameter selection method will be given in Section 4). To study the effects of each
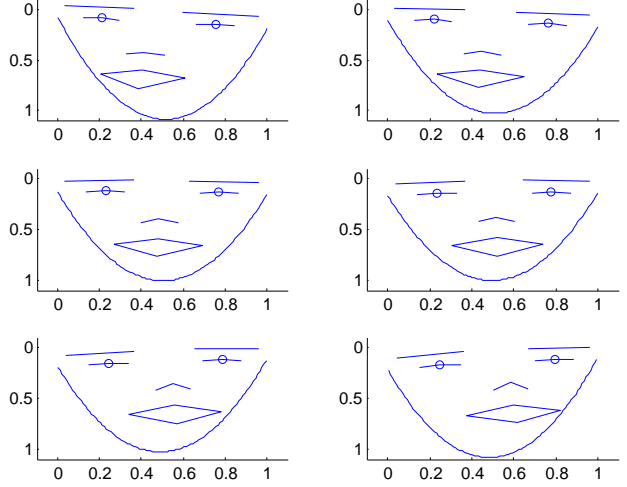

Figure 7. The effect of varying the first PCA feature for ASM.

feature extracted with kernel PCA, we compute the mean $\bar{y}_k$ and the standard deviation σ_{yk} of all the kernel PCA features. Each time, we uniformly select y_k in $[\bar{y}_k - c\sigma_{yk}, \bar{y}_k + c\sigma_{yk}]$ where $c > 0$ is a constant, and set $y_{k'} = \bar{y}_{k'}$ for all $k' \neq k$. The effect of varying the first Gaussian kernel PCA feature and the second Gaussian kernel PCA feature are shown in Figure 9 and Figure 10 respectively.

By observation, we can see that the first PCA feature affects the orientation of the human face, and the second PCA feature to some extent determines some microexpression from amazement to calmness of the human face. In contrast, the first Gaussian kernel PCA feature seems to be determining some microexpression from confidence to fear, while the second Gaussian kernel PCA feature contains both orientation information and some microexpression.

4. Discussion

In this section, we address two concerns: first, how to select the parameters for Gaussian kernel PCA; second, what is the intuitive explanation of Gaussian kernel PCA.

4.1. Parameter Selection

Parameter selection for kernel PCA directly determines the performance of the algorithm. For Gaussian kernel PCA, the most important parameter is the σ in the kernel function defined by (31). The Gaussian kernel relies on the distance $\|\mathbf{x} - \mathbf{y}\|$ between two vec-
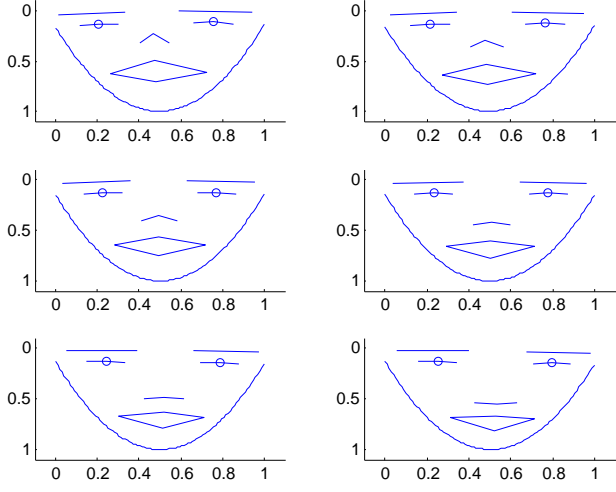
Figure 8. The effect of varying the second PCA feature for ASM.

Figure 9. The effect of varying the first Gaussian kernel PCA feature for ASM.

tors. If the distance between these two vectors is too large, the value of $k(\mathbf{x}, \mathbf{y})$ will be close to zero. Thus, σ is used to enhance the capture range of the kernel function. If σ is too small, all kernel values will be close to zero, and the kernel PCA will fail to extract any information of the structure of the data. We hope that for a given testing data $\mathbf{y}$ and the training data set $\{\mathbf{x}_n\}$, the kernel function gives at least several values of $k(\mathbf{x}_n, \mathbf{y})$ that are significantly larger than zero. Thus we are interested in $\min_n \|\mathbf{y} - \mathbf{x}_n\|$. However, the parameter must be determined in training, thus we define the median minimal distance value *MedMin* of the training data by

$$MedMin = \operatorname*{median}_{n} \min_{m \neq n} \|\mathbf{x}_n - \mathbf{x}_m\|. \qquad (40)$$

We use the median value instead of the maximal or mean value here to exclude some noisy data points that are far away from other data points. Thus for the parameter σ, we require it to be significantly larger than *MedMin*. In our work, we select σ by setting

$$\sigma = 10 \times MedMin, \qquad (41)$$

which turns out to have good performance for our synthetic data classification task and human face images classification task.

For the pre-image reconstruction of Gaussian kernel PCA, the initial guess $\mathbf{z}_0$ will determine whether the iterative algorithm (37) converges. We can simply use the mean of the training data as the initial guess:

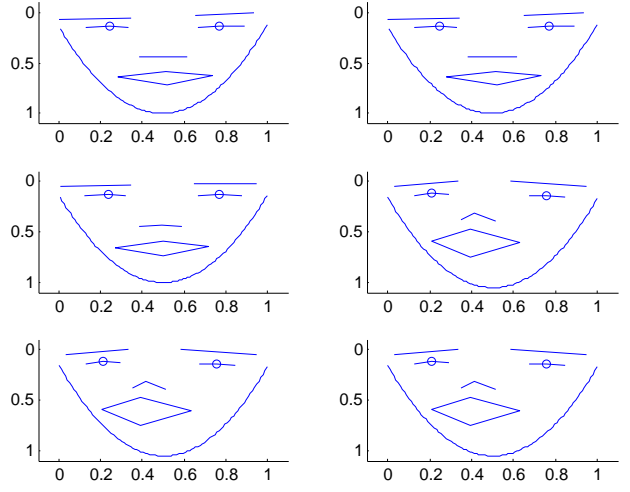$$\mathbf{z}_0 = \operatorname*{mean}_{n} \mathbf{x}_n. \qquad (42)$$

4.2. Intuitive Explanation of Gaussian Kernel PCA

We can see that in our synthetic data classification experiment, Gaussian kernel PCA with a properly selected parameter σ can perfectly separate the two classes in an unsupervised manner, which is not possible for traditional PCA. In the human face images classification experiment, Gaussian kernel PCA has a lower training error rate and a much lower testing error rate than traditional PCA. From these two experiments, we can see that Gaussian kernel PCA reveals more complex hidden structures of the data than traditional PCA. An intuitive understanding of the Gaussian kernel PCA is that it makes use of the distance between different training data points, which is like k-nearest neighbor or clustering methods. With a well selected σ, Gaussian kernel PCA will have a proper capture range, which will enhance the connection between the data points that are close to each other in the original feature space. Then by applying eigenvector analysis, the eigenvectors will describe the direction in a high-dimensional space in which the different clusters of data are scattered to the greatest extent.

In this paper we are mostly using Gaussian kernel for kernel PCA, this is because it is intuitive, easy to implement, and possible to reconstruct the pre-images. However, we indicate that there are techniques to find more powerful kernel matrices by learning (Weinberger et al., 2004; 2005).
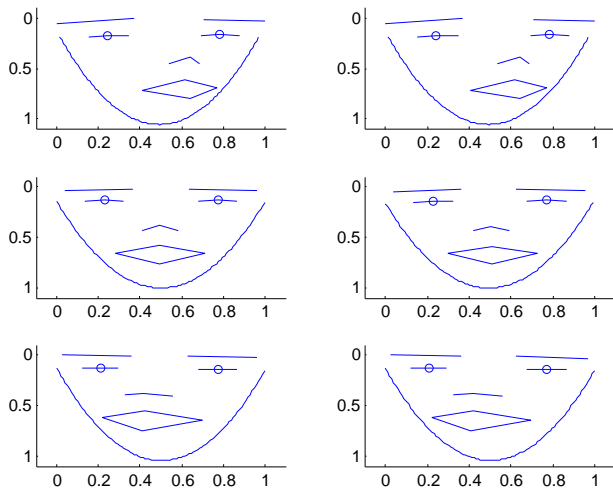
Figure 10. The effect of varying the second Gaussian kernel PCA feature for ASM.

5. Conclusion

In this paper, we discussed the theories of PCA, kernel PCA and ASMs. Then we focused on the pre-image reconstruction for Gaussian kernel PCA, and used this technique to design kernel PCA-based ASMs. We tested kernel PCA dimensionality reduction on synthetic data and human face images, and found that Gaussian kernel PCA succeeded in revealing more complicated structures of data than traditional PCA and achieving much lower classification error rates. We also implemented the Gaussian kernel PCA-based ASM and tested it on human face images. We found that Gaussian kernel PCA-based ASMs are promising in providing more deformation patterns than traditional ASMs. A potential application is that we could combine traditional ASMs and Gaussian kernel PCA-based ASMs for microexpression recognition on human face images. Besides, we proposed a parameter selection method to find the proper parameters for Gaussian kernel PCA, which works well in our experiments.

References

Bakr, Gkhan H., Weston, Jason, and Schlkopf, Bernhard. Learning to find pre-images. In *Advances in Neural Information Processing Systems*, pp. 449–456. MIT Press, 2004.

Bishop, Christopher M. (ed.). *Pattern Recognition and Machine Learning.* Springer, Cambridge, U.K., 2006.

Cootes, T. F. Taylor, C. J. Cooper D. H. Graham J. Active shape models – their training and application. *Computer Vision and Image Understanding*, 61(1):38, 1995.

Demers, David and Y, Garrison Cottrell. Non-linear dimensionality reduction. In *Advances in Neural Information Processing Systems 5*, pp. 580–587. Morgan Kaufmann, 1993.

Georghiades, A.S., Belhumeur, P.N., and Kriegman, D.J. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):643–660, 2001.

Mika, Sebastian, Schlkopf, Bernhard, Smola, Alex, Mller, Klaus-Robert, Scholz, Matthias, and Rtsch, Gunnar. Kernel pca and de-noising in feature spaces. In *Advances in Neural Information Processing Systems 11*, pp. 536–542. MIT Press, 1999.

Romdhani, Sami, Gong, Shaogang, Psarrou, Alexandra, and Y, Ra Psarrou. A multi-view nonlinear active shape model using kernel pca, 1999.

Schlkopf, Bernhard, Smola, Alexander, and Mller, Klaus-Robert. Nonlinear component analysis as a kernel eigenvalue problem, 1996.

Scholkopf, Bernhard, Smola, Alexander, and Mller, Klaus-Robert. Kernel principal component analysis. In *Advances in Kernel Methods – Support Vector Learning*, pp. 327–352. MIT Press, 1999.

Twining, C. J. and Taylor, C. J. Kernel principal component analysis and the construction of non-linear active shape models. In *Proceedings of British Machine Vision Conference*, pp. 23–32, 2001.

Weinberger, Kilian Q., Sha, Fei, and Saul, Lawrence K. Learning a kernel matrix for nonlinear dimensionality reduction. In *Proceedings of the 21st International Conference on Machine Learning (ICML-04)*, pp. 839–846. ACM Press, 2004.

Weinberger, Kilian Q., Packer, Benjamin D., and Saul, Lawrence K. Nonlinear dimensionality reduction by semidefinite programming and kernel matrix factorization. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, pp. 381–388, 2005.