Data Visualization via Kernel Machines

Yuan-chin Ivan Chang[1], Yuh-Jye Lee[2], Hsing-Kuo Pao[2], Mei-Hsien Lee[3], and Su-Yun Huang[1]

[1] Institute of Statistical Science, Academia Sinica, Taipei, Taiwan
ycchang@stat.sinica.edu.tw, syhuang@stat.sinica.edu.tw
[2] Computer Science & Information Engineering, National Taiwan University of Science and Technology, Taipei, Taiwan
yuh-jye@mail.ntust.edu.tw, pao@mail.ntust.edu.tw
[3] Division of Biostatistics, Institute of Epidemiology, School of Public Health, National Taiwan University, Taipei, Taiwan

1 Introduction

Due to the vast development of computer technology, we easily encounter with enormous amount of data collected from diverse sources. That has lead to a great demand for innovative analytic tools for complicated data, where traditional statistical methods can no longer be feasible. Similarly, modern data visualization techniques must face the same situation and provide adequate solutions accordingly.

High dimensionality is always an obstacle to the success of data visualization. Besides the problem of high dimensionality, exploration of information and structures hidden in complicated data can be very challenging. The parametric models, on one hand, are often inadequate for complicated data; on the other hand, the traditional nonparametric methods can be far too complex to have a stable and affordable implementation due to the "curse of dimensionality". Thus, developing new nonparametric methods for analyzing massive data sets is a highly demanding task. With the recent success in many machine learning topics, kernel methods (e.g., Vapnik, 1995) certainly provide us powerful tools for such analysis. Kernel machines facilitate a flexible and versatile nonlinear analysis of data in a very high dimensional (often infinite dimensional) reproducing kernel Hilbert space (RKHS). Reproducing kernel Hilbert spaces have rich mathematical theory as well as topological and geometric structures to allow probabilistic interpretation and statistical inference. They also provide a convenient environment suitable for massive computation.

For many classical approaches, statistical procedures are carried out directly on sample data in Euclidean space $\mathbb{R}^p$. By kernel methods, data are first mapped to a high dimensional Hilbert space, via a certain kernel or its spectrum and classical statistical procedures will act on these kernel-transformed data afterward. Kernel transformations provide us a new way of specifying "distance" or "similarity" metric between different elements.

After preparing the raw data in kernel form, standard statistical and/or mathematical softwares are ready to use for exploring nonlinear structures of data. For instance, we can do nonlinear dimension reduction by kernel principal component analysis (KPCA), which can be useful for constructing high quality classifiers as well as raising new angles of view for data visualization. That is, we are able to view the more complicated (highly nonlinear) structure of the massive data without suffering from the computational difficulties of building complex models. Many multivariate methods can also be extended to cover the highly nonlinear cases through kernel machine framework.

In this article, by combining the classical methods of multivariate analysis, such as PCA, canonical correlation analysis (CCA) and cluster analysis, with kernel machines, we introduce their kernelized counterparts for more versatile and flexible data visualization.

2 Kernel machines in the framework of an RKHS

The goal of this section is twofold. First, it serves as an introduction to some basic theory of RKHS relevant for kernel machines. Secondly, it provides a unified framework for kernelizing some classical linear methods such as PCA, CCA, support vector clustering (SVC), etc. to allow for nonlinear structure exploration. For further details, we refer the reader to Aronszajn (1950) for the theory of reproducing kernels and reproducing kernel Hilbert spaces and Berlinet and Thomas-Agnan (2004) for their usage in probability, statistics and machine learning. Listed below are some definitions and basic properties.

- Let $\mathcal{X} \subset \mathbb{R}^p$ be the sample space of data, and here it serves as an index set. A real symmetric function $\kappa : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ is said to be positive definite if for any positive integer m, any sequence of numbers $\{a_1, a_2, \ldots, a_m \in \mathbb{R}\}$ and points $\{x_1, x_2, \ldots, x_m \in \mathcal{X}\}$, we have $\sum_{i,j=1}^m a_i a_j \kappa(x_i, x_j) \geq 0$.
- An RKHS is a Hilbert space of real valued functions on $\mathcal{X}$ satisfying the property that all evaluation functionals are bounded linear functionals. Note that an RKHS is a Hilbert space of pointwise defined functions, where the $\mathcal{H}$-norm convergence implies pointwise convergence.
- To every positive definite kernel κ on $\mathcal{X} \times \mathcal{X}$ there corresponds a unique RKHS, denoted by $\mathcal{H}_\kappa$, of real valued functions on $\mathcal{X}$. Conversely, to every RKHS $\mathcal{H}$ there exists a unique positive-definite kernel κ such that $\langle f(\cdot), \kappa(x, \cdot)\rangle_{\mathcal{H}} = f(x), \forall f \in \mathcal{H}, \forall x \in \mathcal{X}$, which is known as the reproducing property. We say that this RKHS admits the kernel κ. A positive definite kernel is also named a reproducing kernel.
- For a reproducing kernel κ satisfying the condition $\int_{\mathcal{X} \times \mathcal{X}} \kappa^2(x, u) dx du < \infty$, it has a countable discrete spectrum given by

$$\kappa(x,u) = \sum_q \lambda_q \phi_q(x)\phi_q(u), \quad \text{or} \quad \kappa := \sum_q \lambda_q \phi_q \otimes \phi_q \quad \text{for short.} \tag{1}$$

The main idea of kernel machines is first to map the data in an Euclidean space $\mathcal{X} \subset \mathbb{R}^p$ into an infinite dimensional Hilbert space. Next, a certain classical statistical notion, e.g., say PCA, is carried out in this feature Hilbert space. Such a hybrid model of classical statistical notion and a kernel machine is nonparametric in nature, but its data fitting uses the underlying parametric notion, e.g., PCA finds some leading linear components. The extra effort involved is the preparation for kernel data before feeding them into some classical procedures. Below we will introduce two different but isomorphic maps to embed the underlying Euclidean sample space into a feature Hilbert space. Consider the transformation

$$\Phi : x \mapsto (\sqrt{\lambda_1}\phi_1(x), \sqrt{\lambda_2}\phi_2(x), \ldots, \sqrt{\lambda_q}\phi_q(x), \ldots)' \,. \tag{2}$$

Let $\mathcal{Z} := \Phi(\mathcal{X})$, named the feature space. The inner product in $\mathcal{Z}$ is given by

$$\Phi(x) \cdot \Phi(u) = \sum_q \lambda_q \phi_q(x)\phi_q(u) = \kappa(x,u) \,. \tag{3}$$

The kernel trick (3) of turning inner products in $\mathcal{Z}$ into kernel values allows us to carry out many linear methods in the spectrum-based feature space $\mathcal{Z}$ without explicitly knowing the spectrum Φ itself. Therefore, it makes possible to construct nonlinear (from the Euclidean space viewpoint) variants of linear methods. Consider another transformation

$$\gamma : \mathcal{X} \mapsto \mathcal{H}_\kappa \quad \text{given by} \quad \gamma(x) := \kappa(x,\cdot) \,, \tag{4}$$

which brings a point in $\mathcal{X}$ to an element in $\mathcal{H}_\kappa$. The original sample space $\mathcal{X}$ is thus embedded into a new sample space $\mathcal{H}_\kappa$. The map is called Aronszajn map in Hein and Bousquet (2004). We connect these two maps (2) and (4) via $\mathcal{J} : \Phi(\mathcal{X}) \mapsto \gamma(\mathcal{X})$ given by $\mathcal{J}(\Phi(x)) = \kappa(x,\cdot)$. Note that $\mathcal{J}$ is a one-to-one linear transformation satisfying

$$\|\Phi(x)\|^2_{\mathcal{Z}} = \kappa(x,x) = \|\kappa(x,\cdot)\|^2_{\mathcal{H}_\kappa} = \|\gamma(x)\|^2_{\mathcal{H}_\kappa} \,.$$

Thus, $\Phi(\mathcal{X})$ and $\gamma(\mathcal{X})$ are isometrically isomorphic, and these two feature representations (2) and (4) are equivalent in this sense. Since they are equivalent, mathematically there is no distinction between them. However, for data visualization, there does exist a difference. As the feature map (2) is not explicitly known, there is no way of visualizing the feature data in $\mathcal{Z}$. In this article, for data visualization purpose, data or extracted data features are put in the framework of $\mathcal{H}_\kappa$. We will use the feature map (4) for the later KPCA and kernel canonical correlation analysis (KCCA). As for the SVC, the data cluster will be visualized in the original sample space $\mathcal{X}$, thus, we will use the spectrum-based feature map (2) for ease.

Given data $\{x_1, \ldots, x_n\}$, let us write, for short, the corresponding new data in the feature space $\mathcal{H}_\kappa$ by

$$\gamma(x_j) := \gamma_j \ \ (\in \mathcal{H}_\kappa)\,. \tag{5}$$

As can be seen later, via this new data representations (4) and (5), statistical procedures can be solved in this kernel feature space $\mathcal{H}_\kappa$ in a parallel way using existing algorithms of classical procedures such as PCA and CCA. This is the key spirit of kernelization. The kernelization approach can be regarded, from the original sample space viewpoint, as a nonparametric method, since it adopts a model via kernel mixtures. Still it has the computational advantage of keeping the process analogous to a parametric method, as its implementation involves only solving a parametric-like problem in $\mathcal{H}_\kappa$. The resulting kernel algorithms can be interpreted as running the original parametric (often linear) algorithms on kernel feature space $\mathcal{H}_\kappa$. For the KPCA and KCCA in this article, we use existing PCA and CCA codes on kernel data. One may choose to use codes from Matlab, R, Splus or SAS. The extra programming effort involved is to prepare data in an appropriate kernel form.

Let us discuss another computational issue. Given a particular training data set of size n and by applying the idea of kernel trick in (3), we can generate an $n \times n$ kernel data matrix according to a chosen kernel independent of the statistical algorithms to be used. We then apply the classical algorithm of our interest, which only depends on the dot product, to the kernel matrix directly. Now the issue is, when the data size is huge, generating the full kernel matrix will become a stumbling stone due to the computational cost including CPU time and memory space. Moreover, the time complexity of the algorithm might depend on the size of this full kernel matrix. For example, the complexity of SVM is $O(n^3)$. To overcome these difficulties Lee and Mangasarian (2001a) proposed the "reduced kernel" idea. Instead of using the full square kernel matrix $\mathbb{K}$, they randomly chose only a small portion of columns from $\mathbb{K}$ to form a thin rectangular kernel matrix, called a reduced kernel. The use of partial columns corresponds to the use of partial kernel bases in $\mathcal{H}_\kappa$, while keeping the full rows means that all data points are used for model fitting. The idea of reduced kernel is applied to the smooth support vector machine (Lee and Mangasarian, 2001a, 2001b), and according to their numerical experiments, the reduced kernel method can dramatically cut down the computational load and memory usage without sacrificing much the prediction accuracy. The heuristic is that the reduced kernel method regularizes the model complexity through cutting down the number of kernel bases without sacrificing the number of data points to enter model fitting. This idea also has been successfully applied to the smooth ϵ-insensitive support vector regression (Lee, Hsieh and Huang, 2005), which is a penalized ϵ-insensitive least squares fit and results in an adaptive ridge-type support vector regression estimator. For theoretical study of the reduced kernel method we refer the reader to Lee and Huang (2006). Comparison study of the empirical behavior of eigenvalues and eigenvectors of the full kernel versus reduced kernel can also be found therein. Also note that the feature representation (4) conveniently allows the related optimization problems to be solved in its primal form as compared with in the dual form. The primal optimization has several advantages and they are especially prominent for large scale problems. The primal optimization directly optimizes the objective function and often the solution can be obtained in only a couple of gradient steps, and it can easily accommodate the reduced kernel approach or other low-rank approximation approach for large scale problems. Although, the kernel machine packages are conveniently available and included in many softwares, such as R, Matlab, etc., this reduced kernel method allows us to utilize the kernel machines with less computational efforts especially for large data sets. In this article, the reduced kernel is adopted in conjunction with the algorithms for KPCA and KCCA. Some reference papers and Matlab code are available at `http://dmlab1.csie.ntust.edu.tw/downloads`.

3 Kernel principal component analysis

To deal with high dimensional data, methods of projection pursuit play a key role. Among various approaches, PCA is probably the most basic and commonly used one for dimension reduction. As an unsupervised method, it looks for an r-dimensional linear subspace, with $r < p$ carrying as much information (in terms of data variability) as possible. Operationally, it sequentially finds a new coordinate axis each time, which assumes the "best" direction to view the data, and along which data can own the most variance. Combination of all the new coordinate axes, the so called principal components, forms the basis set for the r-dimensional subspace. It is often the case that a small number of principal components is sufficient to account for most of the relevant data structure and information. They are sometimes called factors or latent variables of the data. For the classical PCA, we try to find the leading eigenvectors by solving an eigenvalue problem in the original sample space. We refer the reader to Mardia, Kent and Bibby (1979) and Alpaydin (2004) for further details. Due to its nature, PCA can only find linear structures in data. If we are interested in not only linear features, but also in nonlinear ones, it is natural to ask what we can do and how we do it? Inspired by the success of kernel machines, Schölkopf, Smola and Müller (1998) and Schölkopf, Burges and Smola (1999) raised the idea of kernel principal component analysis. In their papers, they apply the idea of PCA to the feature data in $\mathcal{Z}$ via the feature map (2). Their method allows to analyze higher-order correlations between input variables. In practice, the transformation needs not be explicitly specified and the whole operation can be done by computing the dot products in (3). In this article, our formulation of KPCA is in terms of its equivalent variant in the framework of RKHS $\mathcal{H}_\kappa$ given by (4).

Actually, given any algorithm that can be expressed solely in terms of dot products (i.e., without explicit usage of the variables $\Phi(x)$ themselves), the kernel method enables us to extend to nonlinear versions of this given algorithm. See, e.g., Aizerman, Braverman and Rozonoer (1964) and Boser, Guyon and Vapnik (1992). This general fact is well known to the machine learning community, and is gradually gaining popularity in statistical society, too. Here we give some examples of applying this method in the domain of unsupervised learning, to obtain a nonlinear form of PCA. Some data sets from UCI Machine Learning Benchmark data archives are used for illustration.

3.1 Computation of KPCA

Before getting into KPCA, we briefly review the computational procedure of classical PCA. Let $X \in \mathbb{R}^p$ be a random vector with covariance matrix $\Sigma := \mathrm{Cov}(X)$. To find the first principal component is to find a unit vector $w \in \mathbb{R}^p$ such that the variance of projection of X along w is maximized, i.e.,

$$\max_{w} \; w'\Sigma w \quad \text{subject to} \quad \|w\| = 1\,. \tag{6}$$

This can be rewritten as a Lagrangian problem:

$$\max_{\alpha, w} \; w'\Sigma w - \alpha(w'w - 1)\,, \tag{7}$$

where α is the Lagrange multiplier. Taking derivative with respect to α and w and setting them to zero, we then solve for α and w. Denote the solution by α_1 and w_1. They must satisfy $\Sigma w_1 = \alpha_1 w_1$, and ${w_1}'w_1 = 1$. Therefore, w_1 is obtained by finding the eigenvector associated with the leading eigenvalue α_1. For the second principal component, we look for a unit vector w_2 which is orthogonal to w_1 and maximizes the variance of the projection of X along w_2. That is, in terms of a Lagrangian problem, we solve for α_2, w_2 and β in the following optimization formulation

$$\max_{\alpha_2, \beta, w_2} \; w_2'\Sigma w_2 - \alpha_2(w_2'w_2 - 1) - \beta(w_1'w_2)\,. \tag{8}$$

Using similar procedure, we are able to find the leading principal components sequentially.

Assume for simplicity that the data $\{x_1, \ldots, x_n\}$ are already centered to their mean, then the sample covariance matrix is given by $\Sigma_n = \sum_{j=1}^n x_j x_j'/n$. By applying the above sequential procedure to the sample covariance Σ_n, we can obtain the empirical principal components.

For KPCA using the feature representation (4), mapped data in the feature space $\mathcal{H}_\kappa$ are $\{\gamma_1, \ldots, \gamma_n\}$. The sample covariance (which is also known as a covariance operator in $\mathcal{H}_\kappa$) is given by

$$C_n := \frac{1}{n}\sum_{j=1}^{n}(\gamma_j - \bar{\gamma}) \otimes (\gamma_j - \bar{\gamma}), \tag{9}$$

where $f \otimes g$ is a linear operator defined by $(f \otimes g)(h) := \langle g, h \rangle_{\mathcal{H}_\kappa} f$ for $f, g, h \in \mathcal{H}_\kappa$. Applying similar arguments as before, we aim to find the leading eigen-components of C_n. That is to solve for h in the following optimization problem

$$\max_{h \in \mathcal{H}_\kappa} \ \langle h, C_n h \rangle_{\mathcal{H}_\kappa} \quad \text{subject to} \quad \|h\|_{\mathcal{H}_\kappa} = 1 \,. \tag{10}$$

It can be shown that the solution is of the form $h = \sum_{j=1}^n \beta_j \gamma_j \in \mathcal{H}_\kappa$, where β_j's are scalars. As

$$\langle h, C_n h \rangle_{\mathcal{H}_\kappa} = \sum_{i,j=1}^n \beta_i \beta_j \langle \gamma_i, C_n \gamma_j \rangle_{\mathcal{H}_\kappa} = \beta' \mathbb{K} \left(I_n - \frac{\mathbf{1}_n \mathbf{1}_n'}{n} \right) \mathbb{K} \beta / n \,,$$

and $\|h\|^2_{\mathcal{H}_\kappa} = \beta' \mathbb{K} \beta$, where $\mathbb{K} = [\kappa(x_i, x_j)]$ denotes the $n \times n$ kernel data matrix. The optimization problem can be reformulated as

$$\max_{\beta \in \mathbb{R}^n} \ \beta' \mathbb{K} \left(I_n - \frac{\mathbf{1}_n \mathbf{1}_n'}{n} \right) \mathbb{K} \beta / n \quad \text{subject to} \quad \beta' \mathbb{K} \beta = 1 \,. \tag{11}$$

The Lagrangian of the above optimization problem is

$$\max_{\alpha \in \mathbb{R}, \beta \in \mathbb{R}^n} \ \beta' \mathbb{K} \left(I_n - \frac{\mathbf{1}_n \mathbf{1}_n'}{n} \right) \mathbb{K} \beta / n - \alpha(\beta' \mathbb{K} \beta - 1) \,,$$

where α is the Lagrange multiplier. Taking derivatives with respect to β's and setting them to zero, we get

$$\mathbb{K} \left(I_n - \frac{\mathbf{1}_n \mathbf{1}_n'}{n} \right) \mathbb{K} \beta / n = \alpha \mathbb{K} \beta, \quad \text{or} \quad \left(I_n - \frac{\mathbf{1}_n \mathbf{1}_n'}{n} \right) \mathbb{K} \beta = n \alpha \beta \,. \tag{12}$$

This leads to the eigenvalues-eigenvectors problem for $\left(I_n - \frac{\mathbf{1}_n \mathbf{1}_n'}{n} \right) \mathbb{K}$. Denote its largest eigenvalue by α_1 (note that the multiplicative factor n is absorbed into the eigenvalue) and its associated eigenvector by β_1, then the corresponding first kernel principal component is given by $h_1 = \sum_{j=1}^n \beta_{1j} \gamma_j$ in the feature space $\mathcal{H}_\kappa$. We can sequentially find the second and the third principal components, etc. From (12) we have that β_k, $k = 1, 2, \ldots$, are orthogonal to $\mathbf{1}_n$ and then the normalization $\beta_k' \mathbb{K} \beta_k = 1$ is equivalent to $\beta_k' \left(I_n - \frac{\mathbf{1}_n \mathbf{1}_n'}{n} \right) \mathbb{K} \beta_k = 1$. Thus, β_k is normalized according to $\alpha_k \beta_k' \beta_k = 1$.

For an $x \in \mathbb{R}^p$ and its feature image $\gamma(x) \in \mathcal{H}_\kappa$, the projection of $\gamma(x)$ along the kth eigen-component of C_n is given by

$$\langle \gamma(x), h_k \rangle_{\mathcal{H}_\kappa} = \langle \gamma(x), \sum_{j=1}^n \beta_{kj} \gamma_j \rangle_{\mathcal{H}_\kappa}, = \sum_{j=1}^n \beta_{kj} \kappa(x_j, x) \,, \tag{13}$$

where β_k is the kth eigenvector of $\left(I_n - \frac{\mathbf{1}_n \mathbf{1}_n'}{n} \right) \mathbb{K}$. Therefore the projection of $\gamma(x)$ onto the dimension reduction linear subspace spanned by the leading r eigen-components of C_n is given by

$$\left(\sum_{j=1}^{n} \beta_{1j} \kappa(x_j, x), \ldots, \sum_{j=1}^{n} \beta_{rj} \kappa(x_j, x) \right) \in \mathbb{R}^r .$$

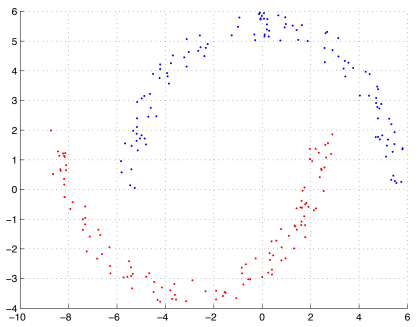
Let us demonstrate the idea of KPCA through a few examples. There are three data sets in this demonstration, the synthesized "two moon" data set, the "Pima Diabetes" data set, and the "Image segmentation" data set.

Example 1 First, we compare the PCA and KPCA using a synthesized "two moons" data set shown in Figure 1. The original data set is in (a) located in a 2-D space. We can observe that the two classes of data can not be well separated along any one-dimensional component. Therefore, by applying PCA, we are not going to see good separation along the first principal coordinate axis. In the histogram of (b1), the horizontal axis is the first principal coordinate from the PCA and vertical axis is the frequency. As we can see, it gives a big portion of overlapping between two classes. On the other hand, a kernelized projection can provide an alternative solution. In the histogram of (b2), the horizontal axis is the first principal coordinate from the KPCA (with polynomial kernel of degree 3) and the vertical axis is the frequency. Still, the KPCA does not give a good separation. However, in the histogram of (b3), the KPCA using radial basis function (RBF, also known as Gaussian kernel) with $\sigma = 1$ gives a good separation. If the PCA or KPCA will be used as a preprocessing step before a classification task, clearly, the KPCA using radial basis function with $\sigma = 1$ is the best choice among them. Their ROC curves are shown in (c), with the area under curve (AUC) reported as $\mathcal{A}_{\mathrm{PCA}} = 0.77$, $\mathcal{A}_{\mathrm{KPCA(Poly)}} = 0.76$ and $\mathcal{A}_{\mathrm{KPCA(RBF)}} = 0.91$. Obviously, the KPCA with RBF ($\sigma = 1$) has clear advantage on the separation between two groups over the classical PCA and the KPCA using polynomial kernel.

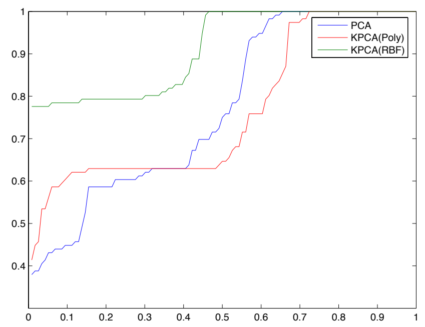
Example 2 In this example we use the Pima Diabetes data set from UCI Machine Learning data archives. The reduced kernel method is adopted by randomly sampling 20% of column vectors from the full kernel matrix. For reduced kernel PCA, assume $\tilde{\mathbb{K}}$ is the underlying reduced kernel data matrix of size $n \times m$, where n is the data size and m is the reduced set size (i.e., columns set size). The KPCA using reduced kernel is a singular value decomposition problem to extract the leading right and left singular vectors $\tilde{\beta}$ and β:

$$\left(I_n - \frac{\mathbf{1}_n \mathbf{1}_n'}{n} \right) \tilde{\mathbb{K}} \tilde{\beta} = \alpha \beta, \;\; \text{normalized to} \;\; \alpha \tilde{\beta}' \tilde{\beta} = 1 \text{ and } \alpha \beta' \beta = 1. \tag{14}$$

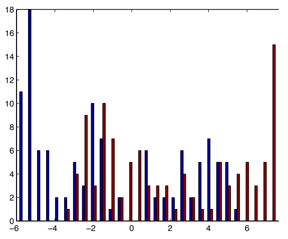
In this data set, there are nine variables including the Number of times pregnant, Plasma glucose concentration (glucose tolerance test), Diastolic blood pressure (mm Hg), Triceps skin fold thickness (mm), 2-Hour serum insulin (mu U/ml), Body mass index (weight in kg/(height in m)^2), Diabetes pedigree function, Age (years), and Class variable (test for diabetes) – positive and negative. For demonstration purpose, we use the first eight variables as input measurements and the last variable as the class variable. The PCA and



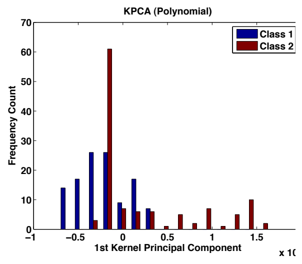
(a)



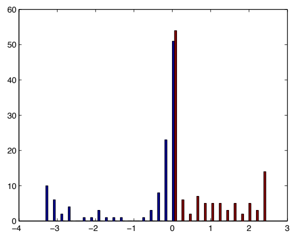
(c)



(b1)



(b2)



(b3)

Fig. 1. PCA and KPCA applied to a synthesized "2 moons" data set. (a) original data, (b1) PCA result, (b2) KPCA with polynomial kernel of degree 3, (b3) KPCA with Gaussian kernel, $\sigma = 1$. In each of these histograms (b1)-(b3), the horizontal axis is the principal coordinate from either the PCA or KPCA and the vertical axis is the frequency. (c) gives their ROC curves with Area Under Curve reported as $\mathcal{A}_{\text{PCA}} = 0.77$, $\mathcal{A}_{\text{KPCA(Poly)}} = 0.76$ and $\mathcal{A}_{\text{KPCA(RBF)}} = 0.91$.

KPCA using both the polynomial kernel and the Gaussian kernel are carried out on the entire input measurements. In Figures 2-4, the red and blue denote the positive and negative samples, respectively. Figure 2 shows the data scatter projected onto the subspace spanned by the first three principal components produced by the classical PCA. Similarly, Figure 3 are plots of data scatter projected onto the subspace spanned by the first three principal components obtained by the KPCA using a polynomial kernel with degree 3 and scale parameter 1. Figures 4(a)-4(c) are pictures of projections with principal components produced by Gaussian kernels with $\sigma^2 = 1/2, 1/6$ and $1/10$, respectively. Comparing Figure 2 obtained by the PCA with others, it is clearly that the KPCA provides some extra information of data, which can not be seen in the classical PCA.

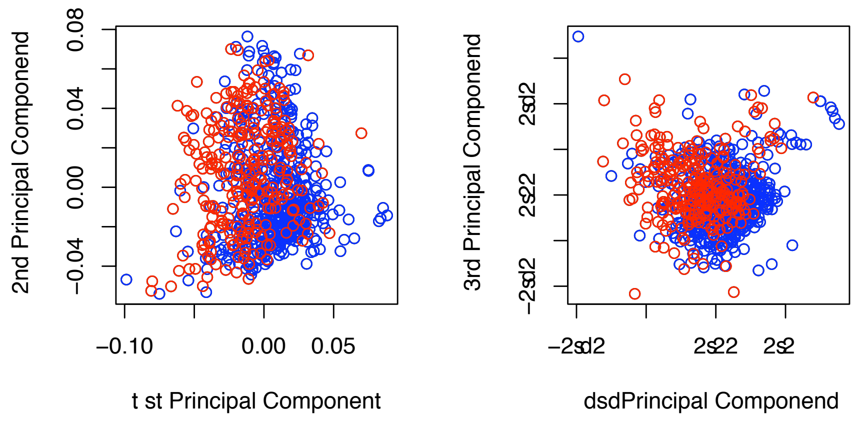


Fig. 2. PCA based on original input variables.

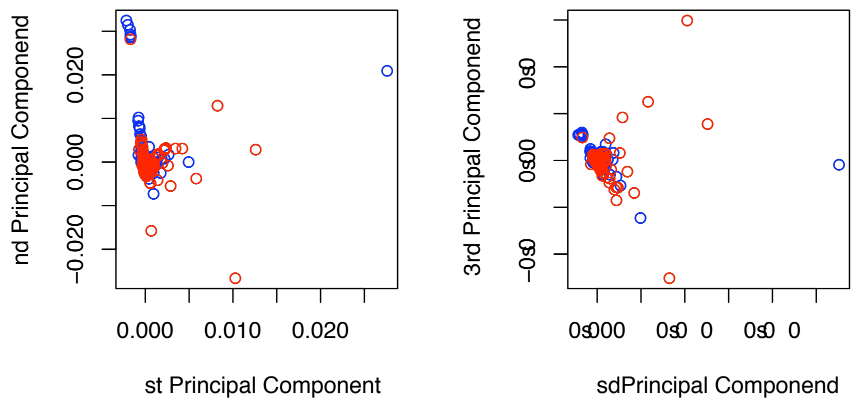
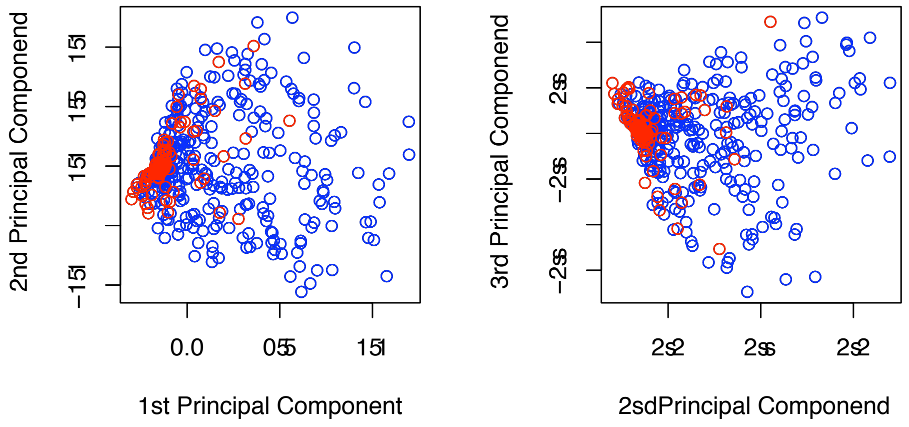
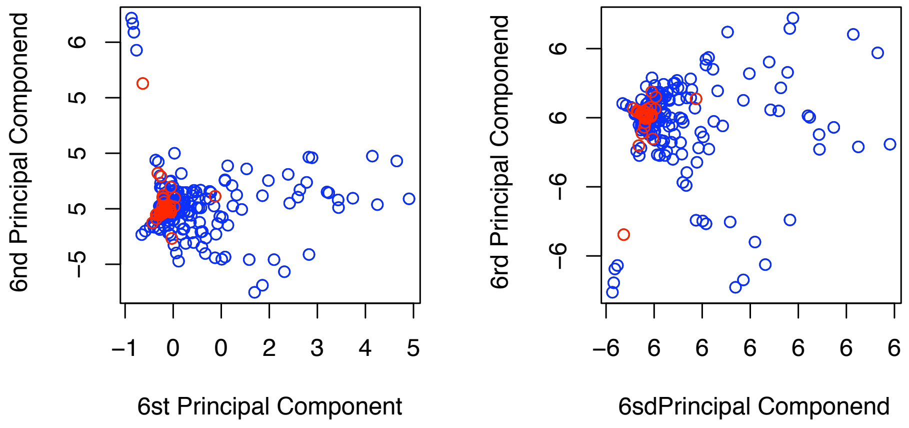


Fig. 3. KPCA with polynomial kernel of degree=3, scale=1.

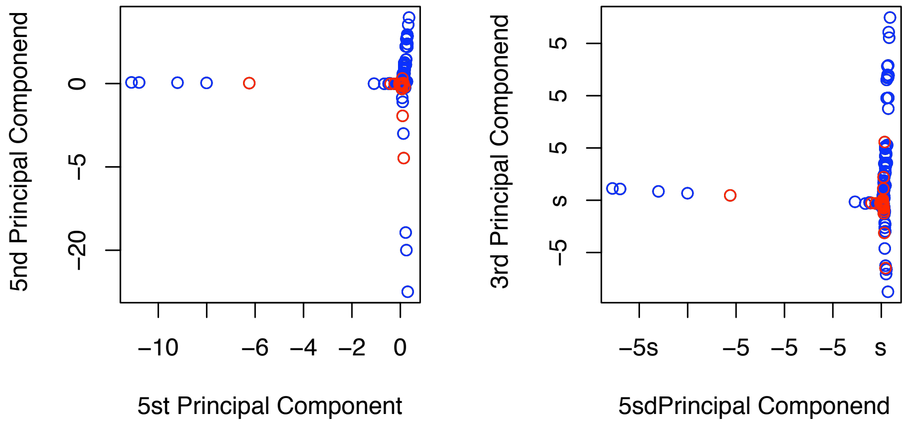
Example 3 In the third series (Figure 5), we apply the PCA and KPCA to the Image Segmentation data set (also from UCI Machine Learning data archives), which consists of 210 data points, each with 19 attributes and classified into 7 classes. The KPCA with RBF gives a better separation of classes than the PCA, as can be seen in (a1) & (b1). With all 7 classes plotted in one graph, it is hard to clearly see the effect of KPCA. Therefore, we further provide figures retaining only "brickface" and "path". We can clearly see the separation produced by the KPCA but not by the PCA.



(a) $\sigma^2 = 1/2$

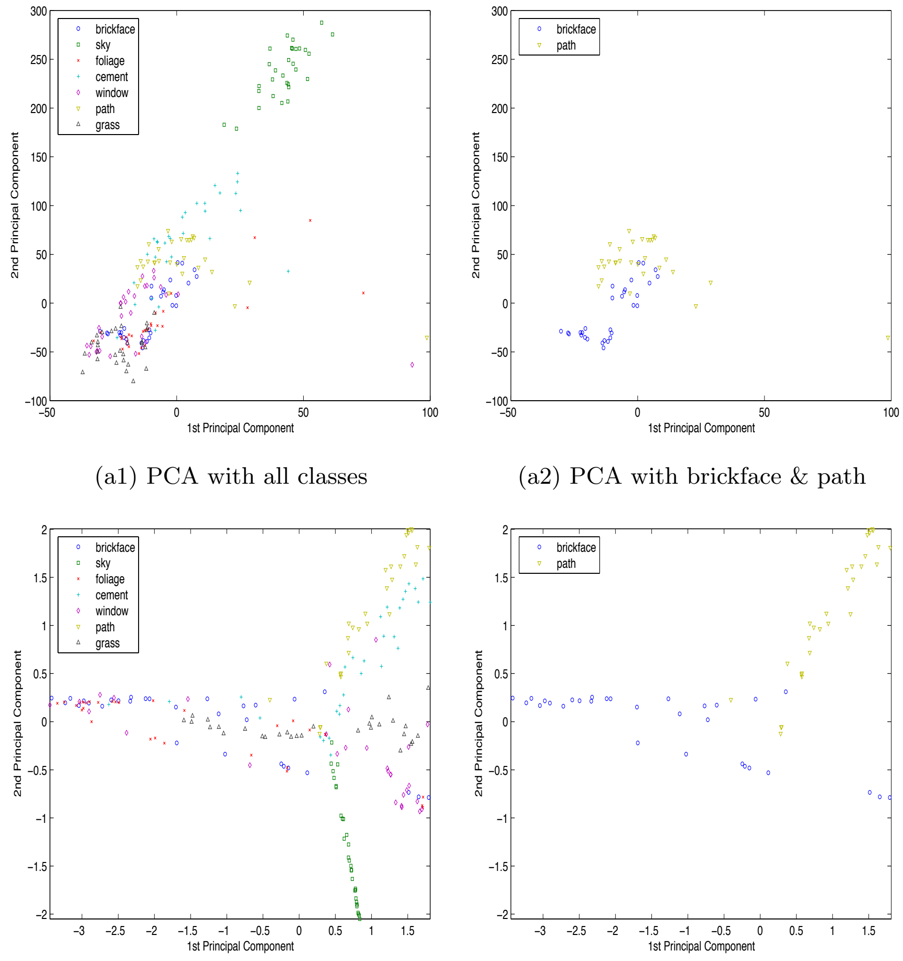


(b) $\sigma^2 = 1/6$



(c) $\sigma^2 = 1/10$

Fig. 4. KPCA with Gaussian kernels.



(a1) PCA with all classes

(a2) PCA with brickface & path

(b1) KPCA with all classes

(b2) KPCA with brickface & path

Fig. 5. PCA vs. KPCA for the "image segmentation" data set. Some limited number of outliers are omitted in the figures.

Remark 1. The choice of kernel and its window width are still an issue in general kernel methodology. There are some works on choice of kernel for classification and supervised learning problems, but it is still lack of guidelines in clustering and non-supervised learning problems. In this experimental study, we merely aim to show that the nonlinear information of data can be obtained through the kernel methods with only minor efforts. The kernel method can really help to dig out nonlinear information of the data, which can be otherwise difficult or impossible to see by the classical linear PCA in the original input space.

4 Kernel canonical correlation analysis

The description and classification of relation between two sets of variables have been a long interest to many researchers. Hotelling (1936) introduced the canonical correlation analysis to describe the linear relation between two sets of variables having a joint distribution. It defines a new coordinate system for each of the sets in a way that the new pair of coordinate systems are optimal in maximizing correlations. The new systems of coordinates are simply linear systems of the original ones. Thus, the classical CCA can only be used to describe linear relations. Via such linear relations it finds only linear dimension reduction subspace and linear discriminant subspace, too. Motivated from the active development and the popular and successful usage of various kernel machines, there has emerged a hybrid approach of the classical CCA with a kernel machine (Akaho, 2001; Bach and Jordan, 2002), named kernel canonical correlation analysis. The KCCA was also studied recently by Hardoon, Szedmak and Shawe-Taylor (2004) and others.

Suppose the random vector X of p components has a probability distribution P on $\mathcal{X} \subset \mathbb{R}^p$. We partition X into

$$X = \begin{bmatrix} X^{(1)} \\ X^{(2)} \end{bmatrix} ,$$

with p_1 and p_2 components, respectively. The corresponding partition of $\mathcal{X}$ is denoted by $\mathcal{X}_1 \oplus \mathcal{X}_2$. We are interested in finding relations between $X^{(1)}$ and $X^{(2)}$. The classical CCA is concerned with linear relations. It describes linear relations by reducing the correlation structure between these two sets of variables to the simplest possible form by means of linear transformations on $X^{(1)}$ and $X^{(2)}$. It finds pairs $(\alpha_i, \beta_i) \in \mathbb{R}^{p_1+p_2}$ in the following way. The first pair maximizes the correlation between $\alpha_1' X^{(1)}$ and $\beta_1' X^{(2)}$ subject to the unit variance constraints $\mathrm{Var}(\alpha_1' X^{(1)}) = \mathrm{Var}(\beta_1' X^{(2)}) = 1$, and the kth pair (α_k, β_k), which are uncorrelated with the first $k-1$ pairs, maximizes the correlation between $\alpha_k' X^{(1)}$ and $\beta_k' X^{(2)}$, and again subject to the unit variance constraints. The sequence of correlations between $\alpha_i' X^{(1)}$ and $\beta_i' X^{(2)}$ describes only the linear relations between $X^{(1)}$ and $X^{(2)}$. There are cases where linear correlations may not be adequate for describing the “associations” between $X^{(1)}$ and $X^{(2)}$. A natural alternative, therefore, is to explore for nonlinear relations. Kernel methods can provide a convenient way for nonlinear generalization. Let $\kappa_1(\cdot, \cdot)$ and $\kappa_2(\cdot, \cdot)$ be two positive definite kernels defined on $\mathcal{X}_1 \times \mathcal{X}_1$ and $\mathcal{X}_2 \times \mathcal{X}_2$, respectively. Let $\mathbb{X}$ denote the data matrix given by

$$\mathbb{X} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}_{n\times p}.$$

Each data point (as a row vector) $x_j = (x_j^{(1)}, x_j^{(2)})$ in the data matrix is transformed into a kernel representation:

$$x_j \mapsto \gamma_j = (\gamma_j^{(1)}, \gamma_j^{(2)})\,, \tag{15}$$

where

$$\gamma_j^{(i)} = (\kappa_i(x_j^{(i)}, x_1^{(i)}), \ldots, \kappa_i(x_j^{(i)}, x_n^{(i)})), \;\; j = 1, \ldots, n, \;\text{ and }\; i = 1, 2\,.$$

Or, by matrix notation, the kernel data matrix is given by

$$\mathbb{K} = [\ \mathbb{K}_1\ \mathbb{K}_2\ ] = \begin{bmatrix} \gamma_1^{(1)} & \gamma_1^{(2)} \\ \vdots & \vdots \\ \gamma_n^{(1)} & \gamma_n^{(2)} \end{bmatrix}_{n\times 2n}, \tag{16}$$

where $\mathbb{K}_i = [\kappa_i(x_j^{(i)}, x_{j'}^{(i)})]_{j,j'=1}^n$, $i = 1, 2$, are the full kernel matrices for data $\{x_j^{(i)}\}_{j=1}^n$. The representation of x_j by $\gamma_j = (\gamma_j^{(1)}, \gamma_j^{(2)}) \in \mathbb{R}^{2n}$ can be regarded as an alternative way of recording data measurements with high inputs.

The KCCA procedure consists of two major steps:

(a) Transform the data points to a kernel representation as in (15) or (16) in matrix notation.

(b) The classical CCA procedure is acting on the kernel data matrix $\mathbb{K}$. Note that some sort of regularization is necessary here to solve the associated spectrum problem of extracting leading canonical variates and correlation coefficients. Here we use the reduced kernel concept stated in the RKHS section and in Example 2. Only partial columns are computed to form reduced kernel matrices, denoted by $\tilde{\mathbb{K}}_1$ and $\tilde{\mathbb{K}}_2$. The classical CCA procedure is acting on the reduced kernel matrix $[\tilde{\mathbb{K}}_1\ \tilde{\mathbb{K}}_2]$.

As the KCCA is simply the classical CCA acting on kernel data, existing code from standard statistical packages are ready for use. In the example below we use Matlab m-file “canoncorr”, which implements the classical CCA, on kernel data.

Example 4 We use the data set “pen-based recognition of hand-written digits” from UCI Machine Learning data archives for visual demonstration of nonlinear discriminant using KCCA. We use the 7494 training instances for explanatory purpose. For each instance there are 16 input measurements (i.e., x_j is 16-dimensional) and a corresponding group label y_j from $\{0, 1, 2, \ldots, 9\}$. A Gaussian kernel with the window width $(\sqrt{10S_1}, \ldots, \sqrt{10S_{16}})$ is used to prepare the kernel data, where S_i's are the coordinate-wise sample covariances. A reduced kernel of size 300 equally stratified over 10 digit groups is used and serves as the $\tilde{\mathbb{K}}_1$ in step (b) of the KCCA procedure. We use y_j, the group labels, as our $\mathbb{K}_2$ (no kernel transformation involved). Precisely,

$$\mathbb{K}_2 = \begin{bmatrix} Y_1' \\ \vdots \\ Y_n' \end{bmatrix}_{n \times 10}, \quad Y_j' = (0, \dots, 1, 0, \dots),$$

where Y_j is a dummy variable for group membership. If $y_j = i$, $i = 0, 1, \dots, 9$, then Y_j has the entry 1 in the $(i+1)$th place and 0 elsewhere. Now we want to explore for the relations between the input measurements and their associated group labels using CCA and KCCA. The training data are used to find the leading CCA- and KCCA-found variates. Next 20 test samples from each digit-group are drawn randomly from the test set [4]. Scatter plots of test data projected along the leading CCA-found variates (Figure 6) and the leading KCCA-found variates (Figure 7) are given below. Different groups are labeled with distinct digits. It is clear that the CCA-found variates are not informative in group labels, while the KCCA-found variates are.

5 Kernel cluster analysis

Cluster analysis is categorized as unsupervised learning method, which tries to find the group structure in an unlabeled data set. A cluster is a collection of data points which are "similar" to points in the same cluster, according to certain criterion, and are "dissimilar" to points belonging to other clusters. The simplest clustering method is probably the k-means (can hybrid with kernel machine, or stands alone). Given a predetermined number of clusters k, the k-means algorithm will proceed to group data points into k clusters by (1) placing k initial centroids in the space, (2) assigning each data point to the cluster of its closest centroid, (3) updating the centroid positions and repeat the steps (1) and (2) until some stopping criterion is reached (see MacQueen, 1967). Despite its simplicity, the k-means algorithm has some disadvantages in certain ways. First, a predetermined k is necessary for the algorithm input, and different k can lead to dramatically different results. Secondly, suboptimal results can occur for certain initial choices of centroid seeds. Thirdly, algorithm may not be appropriate for some data distribution, where the metric is not uniformly defined, i.e., the idea of "distance" has different meanings in different regions or for data belonging to different labels.

Here we address these issues by introducing a different clustering approach, namely, the support vector clustering, which allows hierarchical clusters with

[4] The test set has 3498 instances in total with average around 350 instances for each digit. For clarity of plots and to avoid excess ink, we use only 20 test points per digit.

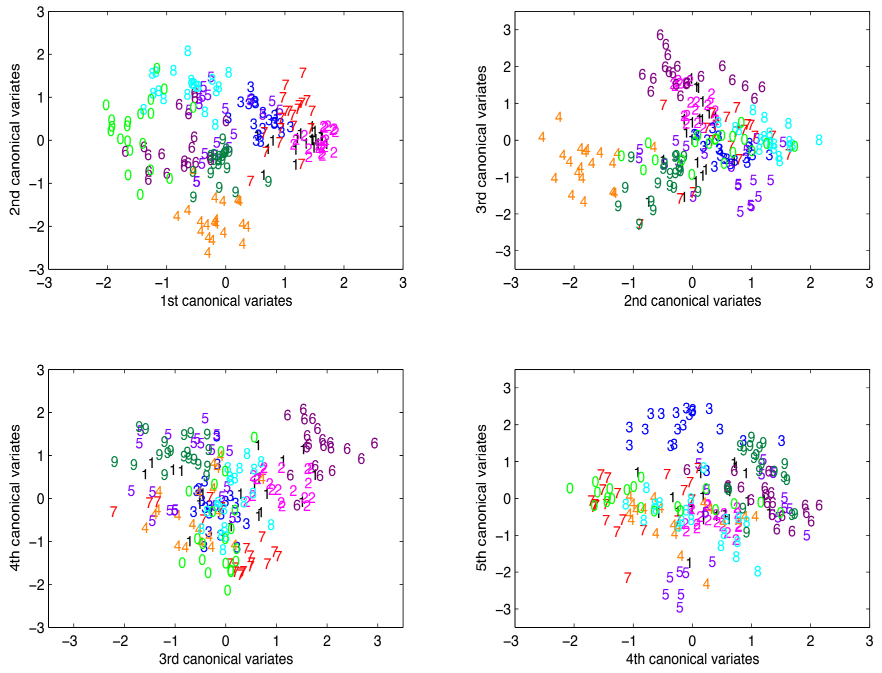


Fig. 6. Scatter plot of pen-digits over CCA-found variates.

versatile clustering boundaries. Below we briefly describe the idea of SVC by Ben-Hur, Horn, Siegelmann and Vapnik (2001). The SVC is inspired from support vector machines and kernel methods. In SVC, data points are mapped from the data space $\mathcal{X}$ to a high dimensional feature space $\mathcal{Z}$ by a nonlinear transformation (2). This nonlinear transformation is defined implicitly by a Gaussian kernel with $\langle \Phi(x), \Phi(u) \rangle_{\mathcal{Z}} = \kappa(x, u)$. The key idea of SVC is to find the smallest sphere in the feature space, which encloses the data images $\{\Phi(x_1), \ldots, \Phi(x_n)\}$. That is, we aim to solve the minimization problem:

$$\min_{\mathbf{a} \in \mathcal{Z}, R} R^2, \quad \text{subject to } \|\Phi(x_j) - \mathbf{a}\|_{\mathcal{Z}}^2 \le R^2, \forall j \,, \tag{17}$$

where R is the radius of an enclosing sphere in $\mathcal{Z}$. To solve the above optimization problem the Lagrangian is introduced. Let

$$L := R^2 - \sum_{j=1}^{n} (R^2 - \|\Phi(x_j) - \mathbf{a}\|^2) \beta_j \,, \tag{18}$$

where $\beta_j \ge 0$ are the Lagrange multipliers. By differentiating L with respect to the primal variables R and $\mathbf{a}$ respectively and setting the derivatives equal

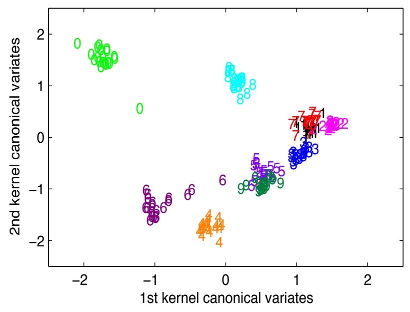
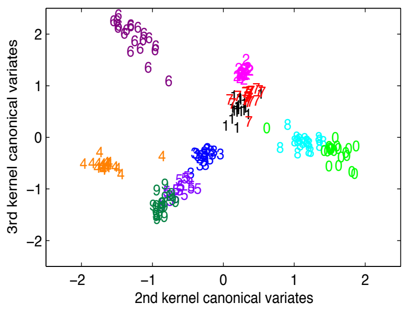
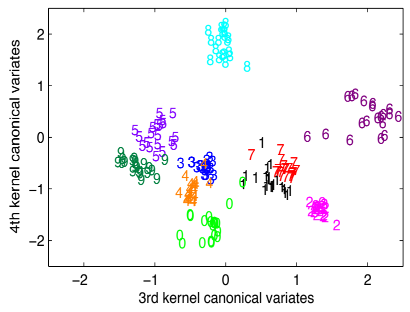
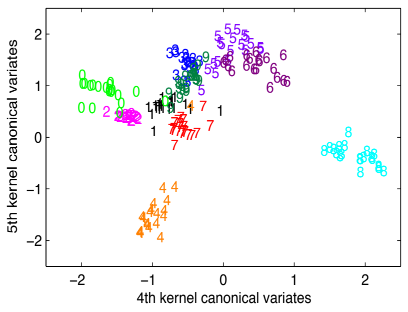





Fig. 7. Scatter plots of pen-digits over KCCA-found variates.

to zero, we have

$$\sum_{j}^{n} \beta_j = 1 \quad \text{and} \quad \mathbf{a} = \sum_{j}^{n} \beta_j \Phi(x_j)\,. \tag{19}$$

Moreover, the corresponding Karush-Kuhn-Tucker complementarity conditions are

$$(R^2 - \|\Phi(x_j) - \mathbf{a}\|^2)\beta_j = 0, \forall j\,. \tag{20}$$

Combining (19) and (20), we can eliminate the primal variables R and $\mathbf{a}$ and get the following dual problem:

$$\begin{aligned} &\max_{\beta} W(\beta) := \sum_{j=1}^{n} \beta_j \|\Phi(x_j) - \bar{\Phi}\|_{\mathcal{Z}}^2 \\ &s.t. \;\; \bar{\Phi} := \textstyle\sum_j \beta_j \Phi(x_j), \quad \sum_j \beta_j = 1 \;\; \text{and} \;\; \beta_j \geq 0\,. \end{aligned} \tag{21}$$

That is, the SVC algorithm aims to find a weighting scheme β so that the weighted data spread $W(\beta)$ appears as far apart as possible.

As R is the radius of the enclosing sphere, corresponding pre-images of the enclosing sphere consist of points $\mathcal{C} := \{x : \|\Phi(x) - \bar{\Phi}\|_{\mathcal{Z}}^2 = R^2\}$. For $x \in \mathcal{C}$ we have

$$\|\Phi(x) - \bar{\Phi}\|_{\mathcal{Z}}^2 = \kappa(x,x) - 2\sum_{j=1}^{n} \beta_j \kappa(x_j, x) + \sum_{j,j'=1}^{n} \beta_j \beta_{j'} \kappa(x_j, x_{j'}) = R^2 .$$

Or equivalently

$$\mathcal{C} := \{x : \sum_{j=1}^{n} \beta_j \kappa(x_j, x) = \rho\} , \tag{22}$$

where $\rho = (\kappa(0,0) + \sum_{j,j'=1}^{n} \beta_j \beta_{j'} \kappa(x_j, x_{j'}) - R^2)/2$. When the enclosing sphere is mapped back to the data space $\mathcal{X}$, it forms a set of probability contours. These contours are used as cluster boundaries and data points inside each contour are assigned into the same cluster. The SVC forms contours by kernel mixture (22) with mixing coefficients β_j being solved from (21). Note that $\bar{\Phi}$ is a weighted centroid in the feature space and $\sum_{j=1}^{n} \beta_j \|\Phi(x_j) - \bar{\Phi}\|_{\mathcal{Z}}^2$ can be regarded as a weighted measure of data dispersion in the feature space. In other words, the SVC algorithm finds mixing coefficients to make data dispersion measure as large as possible in the feature space $\mathcal{Z}$, while it draws the kernel mixture contours in the original data space $\mathcal{X}$ to form clusters. The set $\mathcal{C}$ defines the cluster boundaries. Data points lying on the boundaries are called support vectors. Note that the nonlinear transformation Φ is implicitly defined by a Gaussian kernel, $\kappa(x_j, x_{j'}) = e^{-q\|x_j - x_{j'}\|^2}, q > 0$. (Normalizing constant for κ is not relevant for cluster analysis and is dropped for simplicity.) A larger value of q corresponds to a smaller window width and leads to more resulting clusters in the analysis.

Unlike k-means algorithm, where the number of clusters k has to be prescribed by users, the window width in SVC can vary continuously and results in hierarchical clusters. The number of clusters depends on the window width of the Gaussian kernel. Decreasing the width leads to an increasing number of clusters. Also, different from the procedure of k-means, no initial centroids are required as the algorithm input. Therefore, a deterministic result, independent from initial condition can be expected. The SVC also has the ability to deal with outliers by employing the slack variables. It allows some data points stay outside the enclosing sphere in the feature space. This is same as the "soft margin" idea in support vector machines. With the introduction of slack variables, the optimization problem becomes

$$\min_{\mathbf{a},R,\xi} R^2 + C\sum_{j=1}^{n} \xi_j, \quad \text{subject to } \|\Phi(x_j) - \mathbf{a}\|_{\mathcal{Z}}^2 \le R^2 + \xi_j,\ \xi_j \ge 0, \forall j . \tag{23}$$

It is straightforward to derive the corresponding dual problem:

$$\begin{aligned} &\max_{\beta} W(\beta) := \sum_{j=1}^{n} \beta_j \|\Phi(x_j) - \bar{\Phi}\|_{\mathcal{Z}}^2 \\ &s.t.\ \ \bar{\Phi} := \textstyle\sum_j \beta_j \Phi(x_j), \quad \sum_i \beta_j = 1 \ \text{ and } \ 0 \le \beta_j \le C\,. \end{aligned} \tag{24}$$

As $\langle \Phi(x_j), \Phi(x_{j'}) \rangle_{\mathcal{Z}} = \kappa(x_j, x_{j'})$, the dual problem can be rewritten as a simple quadratic programming problem:

$$\begin{aligned} &\max_{\beta} W(\beta) := -\textstyle\sum_j \sum_{j'} \beta_j \beta_{j'} \kappa(x_j, x_{j'}) \\ &s.t.\ \textstyle\sum_j \beta_j = 1 \ \text{ and } \ 0 \le \beta_j \le C\,. \end{aligned} \tag{25}$$

Solutions for β's are not unique, unless the kernel matrix $\mathbb{K} = [\kappa(x_j, x_{j'})]$ is of full rank. At an optimal solution to problem (25), if $0 < \beta_j < C$, then $\xi_j = 0$ and the corresponding data point x_j and its image $\Phi(x_j)$ lie, respectively, on the cluster boundaries and the surface of the sphere in $\mathcal{Z}$. Such a point is called a *support vector* (SV). The image of a point x_j with $\xi_j > 0$ lies outside the sphere. This will imply the corresponding $\beta_j = C$. Such an x_j is called a *bounded support vector* (BSV). Data points can be classified into three types: SVs lie on the cluster boundaries, BSVs are outside the boundaries, and the rest points lie inside clusters. Since $0 \le \beta_j \le C$ and $\sum_{j=1}^{n} \beta_j = 1$ there is no BSV when $C \ge 1$. Moreover, $1/(nC)$ is an upper bound on the fraction of BSVs.

In a 3-dimensional space, we can easily visualize the clusters once the boundaries are drawn. But in a data space of higher dimension, it is hard to picture and determine which data points are inside a specific cluster. Thus, we need an algorithm for cluster assignment. Ben-Hur et al. (2001) introduce the *adjacency matrix* for cluster assignment. Let $R(y) := \|\Phi(y) - \bar{\Phi}\|_{\mathcal{Z}}$ be the feature distance of $\Phi(y)$ to the data centroid $\bar{\Phi}$. Denote the adjacency matrix by $A = [A_{jj'}]$, where $A_{jj'} = 1$ stands for the pair (j, j') being in the same cluster and $A_{jj'} = 0$ for otherwise. We need only the upper triangular part of the A matrix. For $j < j'$

$$A_{jj'} = \begin{cases} 1 & : \ \text{if } R(y) \le R, \forall y \text{ on the line segment connecting } x_j \text{ and } x_{j'}, \\ 0 & : \ \text{otherwise.} \end{cases}$$

This definition is based on a geometric observation by Ben-Hur et al. For a given pair of data points, say x_j and $x_{j'}$, belonging to different clusters, any path that connects them must exit from the enclosing sphere. Therefore, such a path contains a segment of points y such that $R(y) > R$. Checking all points on a line segment is impossible. In practice, Ben-Hur et al. suggest to use 10 to 20 uniformly distributed points for checking whether $R(y) > R$ or not. Once the adjacency matrix A is formed, the clusters can be defined as the connected components of the graph induced by A. This procedure will leave the BSVs unclustered as outliers. One may either assign them to the nearest clusters or leave them alone. We will illustrate an example to show how the SVC works and the effect of the window width of the Gaussian kernel.

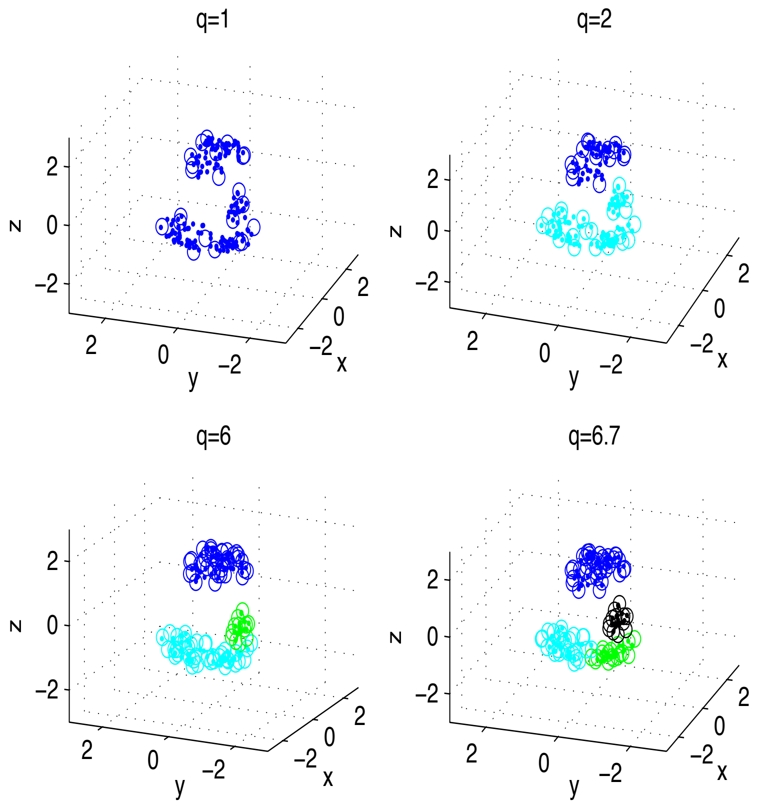


Fig. 8. 200 points in a 3-dimensional space.

Example 5 We synthesize 200 points in R^3 space, among which 80 points are in the upper half sphere of the ball with radius 1.6 and center at the origin. The rest 120 points are generated from three areas of XY plane and then mapped into the lower half sphere of the ball. We vary the parameter q from 1 to 7 and the number of resulting clusters changes from 1 to 4. When $q = 1$, all 200 points are in one cluster, and when $q = 6.7$, we have 4 clusters which is consistent with the way data being generated. The results are depicted in Figure 8. The cluster membership is represented in different colors and "∘" indicates support vectors.

References

1. Aizerman, M.A., Braverman, E.M. and Rozoner, L.I. (1964). Theoretical foundations of the potential function method in pattern recognition learning. *Au-*

tomation and Remote Control, 25: 821–837.

2. Akaho, S. (2001). A kernel method for canonical correlation analysis. *International Meeting of Psychometric Society (IMPS2001).*
3. Alpaydin, E. (2004). *Introduction to Machine Learning.* MIT Press, Cambridge.
4. Aronszajn, N. (1950). Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68: 337–404.
5. Bach, F. and Jordan, M.I. (2002). Kernel independent component analysis. *Journal of Machine Learning Research*, 3: 1–48.
6. Ben-Hur, A., Horn, D., Siegelmann, H.T. and Vapnik, V. (2001). Support vector clustering. *Journal of Machine Learning Research*, 2: 125–137.
7. Berlinet, A. and Thomas-Agnan, C. (2004). *Reproducing Kernel Hilbert Spaces in Probability and Statistics.* Kluwer Academic Publishers, Boston.
8. Boser, B.E., Guyon, I.M. and Vapnik, V.N. (1992). A training algorithm for optimal margin classifiers. *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, 5: 144–152, Pittsburgh, ACM Press.
9. Hardoon, D.R., Szedmak, S. and Shawe-Taylor, J. (2004). Canonical correlation analysis: An overview with application to learning methods. *Neural Computation*, 16(12): 2639–2664.
10. Hein, M. and Bousquet, O. (2004). Kernels, associated structures and generalizations. Technical report, Max Planck Institute for Biological Cybernetics, Germany. `http://www.kyb.tuebingen.mpg.de/techreports.html.`
11. Hotelling, H. (1936). Relations between two sets of variates. *Biometrika*, 28: 321–377.
12. Lee, Y.J., Hsieh, W.F. and Huang, C.M. (2005). ϵ-SSVR: A smooth support vector machine for ϵ-insensitive regression. *IEEE Transactions on Knowledge and Data Engineering*, 17(5): 678–685.
13. Lee, Y.J. and Huang, S.Y. (2006). Reduced support vector machines: a statistical theory. *IEEE Transactions on Neural Networks*, to appear.
14. Lee, Y.J. and Mangasarian, O.L. (2001a). RSVM: Reduced support vector machines. *Proceedings of the First SIAM International Conference on Data Mining.*
15. Lee, Y.J. and Mangasarian, O.L. (2001b). SSVM: A smooth support vector machine for classification. *Computational Optimization and Applications*, 20(1): 5–22.
16. MacQueen, J.B. (1967). Some methods for classification and analysis of multivariate observations. *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability*, Berkeley, University of California Press, 1: 281–297.
17. Mardia, K.V., Kent, J.T. and Bibby, J.M. (1979). *Multivariate Analysis.* Probability and Mathematical Statistics: A Series of Monographs and Textbooks. Academic Press, New York.
18. Schölkopf, B., Burges, C. and Smola, A. (1999). Kernel principal component analysis. *Advances in Kernel Methods - Support Vector Learning*, 327–352. MIT Press, Cambridge.
19. Schölkopf, B., Smola, A. and Müller, K. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5): 1299-1319.
20. Vapnik, V.N. (1995). *The Nature of Statistical Learning Theory.* Springer-Verlag, New York.