Available at
www.ComputerScienceWeb.com
POWERED BY SCIENCE DIRECT®

ELSEVIER

Neurocomputing 55 (2003) 321 – 336

NEUROCOMPUTING

www.elsevier.com/locate/neucom

A comparison of PCA, KPCA and ICA for dimensionality reduction in support vector machine

L.J. Cao[a,*], K.S. Chua[b], W.K. Chong[c], H.P. Lee[a], Q.M. Gu[d]

[a]*Institute of High Performance Computing, 1 Science Park Road, #01-01 the Capricorn, Science Park II, Singapore 117528, Singapore*

[b]*Department of Mathematics, National University of Singapore, 10 Kent Ridge Crescent, Singapore 119260, Singapore*

[c]*Singapore-MIT Alliance, National University of Singapore, 10 Kent Ridge Crescent, Singapore 119260, Singapore*

[d]*Office of Nanjing Committee of People's Republic of China, Beijing East Road 210008, China*

Received 15 March 2002; accepted 8 January 2003

Abstract

Recently, support vector machine (SVM) has become a popular tool in time series forecasting. In developing a successful SVM forecastor, the first step is feature extraction. This paper proposes the applications of principal component analysis (PCA), kernel principal component analysis (KPCA) and independent component analysis (ICA) to SVM for feature extraction. PCA linearly transforms the original inputs into new uncorrelated features. KPCA is a nonlinear PCA developed by using the kernel method. In ICA, the original inputs are linearly transformed into features which are mutually statistically independent. By examining the sunspot data, Santa Fe data set A and five real futures contracts, the experiment shows that SVM by feature extraction using PCA, KPCA or ICA can perform better than that without feature extraction. Furthermore, among the three methods, there is the best performance in KPCA feature extraction, followed by ICA feature extraction.
© 2003 Elsevier B.V. All rights reserved.

Keywords: Support vector machines; Principal component analysis; Kernel principal component analysis; Independent component analysis

* Corresponding author.
E-mail address: caolj@ihpc.a-star.edu.sg (L.J. Cao).

1. Introduction

Recently, support vector machine (SVM) has become a popular tool in time series forecasting [18–20,26,27], due to its remarkable characteristics such as good generalization performance, the absence of local minima and the sparse representation of solution. Unlike most of the traditional methods which implement the *Empirical Risk Minimization Principal*, SVM implements the *Structural Risk Minimization Principal* which seeks to minimize an upper bound of the generalization error rather than minimize the training error [34]. This eventually results in better generalization performance in SVM than other traditional methods. As the training of SVM is equivalent to solving a linearly constrained convex quadratic programming problem, the solution of SVM is always global optimal and absent from local minima. Actually, the solution is determined by only support vectors which are a subset of training data points, so it can be represented sparsely.

In developing a SVM forecastor, the first important step is feature selection (new features are selected from the original inputs) or feature extraction (new features are transformed from the original inputs). In the modeling, all available indicators can be used as the inputs of SVM, but irrelevant features or correlated features could deteriorate the generalization performance of SVM. In the framework of SVM, several approaches for feature selection are also available. In [3], Bradley and Mangasarian find that SVM with 1-norm regularized term are an indirect approach of feature selection. In [9], the recursive feature elimination method is proposed to SVM for feature selection. Weston et al. [36] also propose the gradient descent method to SVM for feature selection. In our previous work [28,29], saliency analysis and genetic algorithm are also proved to be useful for selecting important features in SVM. In summary, all these approaches are in the domain of feature selection.

Principal component analysis (PCA) is a well-known method for feature extraction. By calculating the eigenvectors of the covariance matrix of the original inputs, PCA linearly transforms a high-dimensional input vector into a low-dimensional one whose components are uncorrelated. Nonlinear PCA has also been developed by using different algorithms [6]. Kernel principal component analysis (KPCA) is one type of nonlinear PCA developed by generalizing the kernel method into PCA [23]. The kernel method is originally used for SVM. Later, it has been generalized into many algorithms having the term of dot products such as PCA. Specifically, KPCA firstly maps the original inputs into a high-dimensional feature space using the kernel method and then calculates PCA in the high-dimensional feature space. The linear PCA in the high-dimensional feature space corresponds to a nonlinear PCA in the original input space. Recently, another linear transformation method called independent component analysis (ICA) is also developed [4]. Instead of transforming uncorrelated components, ICA attempts to achieve statistically independent components in the transformed vectors. ICA is originally developed for blind source separation. Later, it has been generalized for feature extraction [2,10,13].

The purpose of this paper is to compare the performance of PCA, KPCA and ICA for feature extraction in the context of SVM. By using one of the three methods, the original higher-dimensional inputs will be transformed into other lower-dimensional features. There are different features extracted in PCA, KPCA and ICA due to their different algorithms used. The new features are then used as the inputs of SVM to solve time series forecasting problems. By examining the sunspot data, Santa Fe data set A and five real futures contracts, the simulation shows that SVM by feature extraction using PCA, KPCA or ICA can always perform better than that without feature extraction. Furthermore, among the three methods, there is the best performance in KPCA feature extraction, followed by ICA feature extraction.

The rest of this paper is organized as follows. In Section 2, PCA, KPCA and ICA for feature extraction are described. In Section 3, the theory of SVM for regression estimation is presented. Section 4 gives the experimental results, followed by the conclusions in the last section.

2. Feature extraction methods

2.1. Principal component analysis

Given a set of centered input vectors x_t ($t=1,\dots,l$ and $\sum_{t=1}^{l} x_t=0$), each of which is of m dimension $x_t=(x_t(1),x_t(2),\dots,x_t(m))^{\mathrm{T}}$ (usually $m < l$), PCA linearly transforms each vector x_t into a new one s_t by

$$s_t = U^{\mathrm{T}} x_t, \tag{1}$$

where U is the $m \times m$ orthogonal matrix whose ith column u_i is the ith eigenvector of the sample covariance matrix $C=\frac{1}{l}\sum_{t=1}^{l} x_t x_t^{\mathrm{T}}$.

In other words, PCA firstly solves the eigenvalue problem (2).

$$\lambda_i u_i = C u_i, \quad i=1,\dots,m, \tag{2}$$

where λ_i is one of the eigenvalues of C. u_i is the corresponding eigenvector. Based on the estimated u_i, the components of s_t are then calculated as the orthogonal transformations of x_t:

$$s_t(i) = u_i^{\mathrm{T}} x_t, \quad i=1,\dots,m. \tag{3}$$

The new components are called principal components. By using only the first several eigenvectors sorted in descending order of the eigenvalues, the number of principal components in s_t can be reduced. So PCA has the dimensional reduction characteristic. The principal components of PCA also have the following properties [14].

(1) $s_t(i)$, $i=1,\dots,m$ are uncorrelated.
(2) $s_t(i)$, $i=1,\dots,m$ have sequentially maximum variances.
(3) The mean-squared approximation error in the representation of the original inputs by the first several principal components is minimal.

2.2. Kernel principal component analysis

KPCA is one approach of generalizing linear PCA into nonlinear case using the kernel method. The idea of KPCA is to firstly map the original input vectors x_t into a high-dimensional feature space $\phi(x_t)$ and then to calculate the linear PCA in $\phi(x_t)$. The linear PCA in $\phi(x_t)$ corresponds to a nonlinear PCA in x_t. By mapping x_t into $\phi(x_t)$ whose dimension is assumed to be larger than the number of training samples l, KPCA solves the eigenvalue problem (4).

$$\lambda_i u_i = \tilde{C} u_i, \quad i = 1, \ldots, l, \tag{4}$$

where $\tilde{C} = \frac{1}{l} \sum_{t=1}^{l} \phi(x_t)\phi(x_t)^{\mathrm{T}}$ is the sample covariance matrix of $\phi(x_t)$. λ_i is one of the non-zero eigenvalues of $\tilde{C}$. u_i is the corresponding eigenvector.

Eq. (4) can be transformed to the eigenvalue problem (5) [23].

$$\tilde{\lambda}_i \alpha_i = K \alpha_i, \quad i = 1, \ldots, l, \tag{5}$$

where K is the $l \times l$ kernel matrix. The value of each element of K is equal to the inner product of two vectors x_i and x_j in the high-dimensional feature space $\phi(x_i)$ and $\phi(x_j)$. That is, $K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$. As the calculations of the dot product $\phi(x_i) \cdot \phi(x_j)$ are all replaced with the kernel function $K(x_i, x_j)$, the mapping of $\phi(x_t)$ from x_t is implicit. The elegance of using K is that one can deal with $\phi(x_t)$ of arbitrary dimensionality without having to compute $\phi(x_t)$ explicitly. Any function satisfying Mercer's condition [17] can be used as K. $\tilde{\lambda}_i$ is one of the eigenvalues of K, satisfying $\tilde{\lambda}_i = l\lambda_i$. α_i is the corresponding eigenvector of K, satisfying $u_i = \sum_{j=1}^{l} \alpha_i(j)\phi(x_j)$ ($\alpha_i(j)$, $j = 1, \ldots, l$ are the components of α_i). Furthermore, for assuring the eigenvectors of $\phi(x_t)$ is of unit length $u_i \cdot u_i = 1$, each α_i must be normalized using the corresponding eigenvalue by $\tilde{\alpha}_i = \frac{\alpha_i}{\sqrt{\tilde{\lambda}_i}}$.

Finally, based on the estimated $\tilde{\alpha}_i$, the principal components for x_t is calculated by

$$s_t(i) = u_i^{\mathrm{T}} \phi(x_t) = \sum_{j=1}^{l} \tilde{\alpha}_i(j) K(x_j, x_t), \quad i = 1, \ldots, l \tag{6}$$

In addition, for making the sample input vectors in $\phi(x_t)$ centered $\sum_{t=1}^{l} \phi(x_t) = 0$, in Eq. (6), the kernel matrix on the training set K and on the testing set K_t are, respectively, modified by

$$\tilde{K} = \left(I - \frac{1}{l} 1_l 1_l^{\mathrm{T}}\right) K \left(I - \frac{1}{l} 1_l 1_l^{\mathrm{T}}\right), \tag{7}$$

$$\tilde{K}_t = \left(K_t - \frac{1}{l} 1_{l_t} 1_l^{\mathrm{T}} K\right) \left(I - \frac{1}{l} 1_l 1_l^{\mathrm{T}}\right), \tag{8}$$

where I is l-dimensional identity matrix. l_t is the number of testing data points. 1_l and 1_{l_t} represent the vectors whose elements are all ones, with length l and l_t, respectively. K_t represents the $l_t \times l$ kernel matrix for the testing data points.

From Eq. (6), it can be found that KPCA can extract more number of principal components than PCA as the maximal number of principal components in KPCA is l, instead of m. Same as PCA, the dimension of s_t can also be reduced in KPCA if only considering the first several eigenvectors. By using the kernel method to implement nonlinear PCA, the other properties of PCA as described in (1)–(3) are all retained in KPCA except these characteristics are considered in $\phi(x_t)$.

2.3. Independent component analysis (ICA)

ICA is originally developed for blind source separation whose goal is to recover mutually independent but unknown source signals from their linear mixtures without knowing the mixing coefficients. Let x_t denote the linear mixtures and s_t denote the original source signals, the goal of the ICA is to estimate s_t by

$$s_t = Ux_t, \tag{9}$$

so that the estimated components of s_t are statistically as independent as possible. U is called the $m \times m$ un-mixing matrix. “The statistical independence” implies that the joint probability density of the components of s_t equal to the product of the marginal densities of the individual components. Thus, the higher order information of the original inputs is required for estimating s_t, rather than the second-order information of the sample covariance as used in PCA. For the identification of Eq. (9), one fundamental requirement is that all the independent components of s_t, with the possible exception of one component, must be non-Gaussian.

A large amount of algorithms have been developed for performing ICA [1,8,15,21]. One of the best methods is the fixed-point-FastICA algorithm proposed by Giannakopoulos et al. [7]. In the FastICA algorithm, the mutual information is used as the criterion to estimate s_t as it is a natural measure of the independence between random variables. Minimizing the mutual information between the components corresponds to maximizing their negentropy. However, maximizing the negentropy cannot be performed directly as the involved probability densities of the components are unknown. In the FastICA algorithm, the negentropy is approximated by using the contrast function which has the following form:

$$J_G(u_i) = [E\{G(u_i^{\mathrm{T}} x_t)\} - E\{G(v)\}]^2 \tag{10}$$

where u_i is a m-dimensional vector, comprising one of the rows of the matrix U. v is a standardized Gaussian variable. G is a non-quadratic function. The commonly used functions for G are $G_1(s) = (1/\gamma_1)\log\cosh(\gamma_1 s)$, $G_2(s) = -(1/\gamma_2)\exp(-\gamma_2(s^2/2))$ and $G_3(s) = \frac{1}{4}s^4$, where γ_1 and γ_2 are parameters with $1 \leqslant \gamma_1 \leqslant 2$ and $\gamma_2 \approx 1$. Maximizing $J_G(u_i)$ leads to estimating u_i by

$$u_i^+ = E\{x_t g(u_i^{\mathrm{T}} x_t)\} - E\{g'(u_i x_t)\}u_i, \tag{11}$$

$$u_i^* = u_i^+ / \|u_i^+\|, \tag{12}$$

where u_i^* is a new estimated value of u_i. g and g' are, respectively, the first and second derivatives of G. Based on the maximal negentropy principal, the whole matrix U can be computed by maximizing the sum of one-unit contrast function and taking into account the constraint of decorrelation [11,12].

To simplify the FastICA algorithm, two preprocessing steps are applied to x_t. First, x_t is made zero mean by subtracting its mean. That is, $x_t = x_t - E\{x_t\}$. The second step is to whiten x_t so that the whitened variable $\tilde{x}_t$ has a correlation matrix of unity one. That is, $\tilde{x}_t = Vx_t$ and $E\{\tilde{x}_t\tilde{x}_t^{\mathrm{T}}\} = I$. V can be obtained by using PCA. The use of PCA whitening also has the property of reducing the dimension of $\tilde{x}_t$, eventually reducing the number of components of s_t. A more detailed description of the FastICA algorithm can be found in [11].

There are two distinct characteristics between PCA and ICA. Firstly, the components of ICA are statistically independent, not merely uncorrelated as in PCA. Secondly, the un-mixing matrix U of ICA is not orthogonal, unlike that of PCA whose components are represented on an orthonormal basis.

3. Theory of SVM for regression estimation

After feature extraction using PCA, KPCA or ICA, the training data points can be expressed as $(s_1, y_1), (s_2, y_2), \ldots, (s_l, y_l)$ ($s_i \in R^n, n \leqslant m$ is the transformed input vector, $y_i \in R$ is the target value), SVM approximates the function using the following form:

$$f(s) = w \cdot \phi(s) + b, \tag{13}$$

where $\phi(s)$ represents a high-dimensional feature space which is nonlinearly mapped from the input space s. The coefficients w and b are estimated by minimizing the regularized risk function (14).

$$\text{minimize} \quad \frac{1}{2}\|w\|^2 + C\frac{1}{l}\sum_{i=1}^{l} L_\varepsilon(y_i, f(\phi(s_i))) \tag{14}$$

$$L_\varepsilon(y_i, f(\phi(s_i))) = \begin{cases} |y_i - f(\phi(s_i))| - \varepsilon & |y_i - f(\phi(s_i))| \geqslant \varepsilon, \\ 0 & \text{otherwise.} \end{cases} \tag{15}$$

The first term $\|w\|^2$ is called the regularized term. Minimizing $\|w\|^2$ will make a function as flat as possible, thus playing the role of controlling the function capacity. The second term $1/l \sum_{i=1}^{l} L_\varepsilon(y_i, f(\phi(s_i)))$ is the empirical error measured by the ε-insensitive loss function (15). This loss function provides the advantage of using sparse data points to represent the designed function (13). C is referred to as the regularization constant. ε is the tube size of SVM. They are both user-prescribed parameters and determined empirically.

To get the estimations of w and b, Eq. (14) is transformed to the primal objective function (16) by introducing the positive slack variables $\xi_i^{(*)}$.

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}(\xi_i + \xi_i^*) \\ \text{subject to} \quad & y_i - w \cdot \phi(s_i) - b \leqslant \varepsilon + \xi_i, \\ & w \cdot \phi(s_i) + b - y_i \leqslant \varepsilon + \xi_i^*, \\ & \xi_i^{(*)} \geqslant 0, \quad i = 1, \ldots, l. \end{aligned} \tag{16}$$

Finally, by introducing Lagrange multipliers and exploiting the optimality constraints, the decision function (16) has the following explicit form [34]:

$$f(s) = \sum_{i=1}^{l} (a_i - a_i^*) K(s_i, s) + b. \tag{17}$$

In Eq. (17), K is the kernel function. The value is $K(s_i, s_j) = \phi(s_i) \cdot \phi(s_j)$. a_i and a_i^* are the so-called Lagrange multipliers. They satisfy the equalities $a_i a_i^* = 0$, $a_i \geqslant 0$ and $a_i^* \geqslant 0$ where $i = 1, \ldots, l$, and they are obtained by maximizing the dual function of (18), which has the following form:

$$\begin{aligned} W(a_i^{(*)}) = {} & \sum_{i=1}^{l} y_i (a_i - a_i^*) - \varepsilon \sum_{i=1}^{l} (a_i + a_i^*) \\ & - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} (a_i - a_i^*)(a_j - a_j^*) K(s_i, s_j) \end{aligned} \tag{18}$$

with the following constraints:

$$\sum_{i=1}^{l} (a_i - a_i^*) = 0,$$

$$0 \leqslant a_i^{(*)} \leqslant C, \quad i = 1, \ldots, l.$$

Based on the Karush–Kuhn–Tucker conditions [16], only a number of coefficients $(a_i - a_i^*)$ will assume nonzero values, and the corresponding training data points are referred to as support vectors. According to Eq. (17), it is evident that only support vectors are used to determine the decision function as the values of $(a_i - a_i^*)$ for the other training data points are equal to zero. As support vectors are usually only a small subset of the training data points, this characteristic is referred to as the sparsity of the solution.

4. Experimental results

4.1. Sunspot data

The sunspot data set is believed to be nonlinear and non-stationary. It has long served as a benchmark and been well studied in the previous literature [5,33,35]. The data set consists of a total of 280 yearly averaged sunspots recorded from 1700 to 1979. To make results comparable, this study uses the same experimental setup as used in [33,35]. That is, the data points from 1700 to 1955 are used as the training set, and the remaining data points from 1956 to 1979 are used as the test set. Twelve previous sunspots are used to predict the current sunspot. For this small data set, the 5-fold cross validation method is used to select the optimal free parameters of SVM, KPCA and ICA. That is, the whole training set is firstly sequentially partitioned into five parts. Four parts are used as the training set, and the remaining one part is used

Table 1
The parameters used and the converged NMSE in the sunspot data

Methods	$(\delta^2, C, \varepsilon, \delta^2(\gamma_2))^a$	n	NMSE
Original + SVM	(10, 100, 0.05)	—	0.1933
PCA + SVM	(100, 100, 0.01)	7	0.1853
KPCA + SVM	(1, 1000, 0.05, 0001)	131	0.1544
ICA + SVM	(1000, 1000, 0.05, 1.0)	10	0.1651
Benchmarks	0.28 [33]		
	0.35 [35]		

[a]The first δ^2, C, ε are the parameters of SVM. The second δ^2 is the Gaussian parameter of KPCA. γ_2 is the parameter of ICA. n is the number of principal components used.

as the validation set. This process is repeated until each of the partitioned parts is used as the validation set. The free parameters which produce the smallest sum of the validation errors are used in the experiment. The error is measured by using the normalized mean square error (NMSE) criterion. The NMSE is calculated as follows:

$$\mathrm{NMSE} = \frac{1}{\delta^2 l_t}\sum_{i=1}^{l_t}(y_i - \hat{y}_i)^2, \tag{19}$$

$$\delta^2 = \frac{1}{l_t - 1}\sum_{i=1}^{l_t}(y_i - \bar{y})^2, \tag{20}$$

where $\hat{y}$ represents the predicted value. $\bar{y}$ denotes the mean of the actual values.

The PCA program used is developed using Matlab. The KPCA algorithm used is also a Matlab program developed by Rosipal [22], which can be downloaded from http://www.um.sav.sk/dep_3/rosi/publications.html. The Gaussian function $\exp(-(x_i - x_j)^2/\delta^2)$ is used as the kernel function of KPCA because the Gaussian kernel tends to give good performance under general smoothness assumptions [24]. The best value of δ^2 is chosen based on the 5-fold cross validation method, as described in Table 1. The FastICA algorithm for implementing ICA is also a Matlab program developed by Hyvärinen [11], which can be downloaded from http://www.cis.hut.fi/projects/ica/fastica/1998. For the choice of G, the three commonly used functions described in Section 2.3 are all investigated. The function $-(1/\gamma_2)\exp(-\gamma_2(s^2/2))$ is used as it produces the smallest sum of validation errors. Based on the 5-fold cross validation method, the value of γ_2 is used as 1.0 as described in Table 1.

In the implementation of PCR, KPCR and ICA, one major problem is to search for the optimal number of principal components n in s_t. The following procedure is used in the experiment. Firstly, the principal component calculated using the eigenvector corresponding to the largest eigenvalue is used as the inputs of SVM to perform the prediction. The principal components are then increased one by one (corresponding to the eigenvectors sorted in a descending order of eigenvalues) at each step. So the number of principal components from the minimum value 1 to the maximum value

(12 in PCA and ICA; 209 in KPCA) are all investigated. The value which produces the smallest sum of cross validation errors is used, which is also given in Table 1.

For training SVM, the sequential minimal optimization algorithm solving the regression problem [24,25] is implemented and the program is developed using VC^{++} language. The Gaussian function is also used as the kernel function of SVM. The optimal values of the Gaussian parameter δ^2, C and ε are chosen based on the 5-fold cross validation method. As shown in Table 1, the values of δ^2, C and ε could vary before feature extraction and after feature extraction.

The best results obtained in SVM by using PCA feature extraction (PCA + SVM), KPCA feature extraction (KPCA+SVM), and ICA feature extraction (ICA+SVM) and the original inputs without feature extraction (original + SVM) are given in Table 1. The benchmarks reported in [33,35] are also included in the table for comparison. Table 1 shows that our result obtained by using SVM without feature extraction is much better than the benchmarks. Furthermore, by feature extraction using PCA, KPCA or ICA, SVM can always converge to a smaller NMSE on the test set than that without feature extraction. This demonstrates the fact that dimension reduction can improve the generalization performance of SVM. Among the three feature extraction methods, there is the smallest error in KPCA feature extraction, followed by ICA feature extraction. From Table 1, it can also be observed that there is a larger number of principal components used in KPCA in comparison with those of PCA and ICA.

4.2. Santa Fe data set A

The data set A in Santa Fe Competition is a laser time series recorded from a Far-Infrared-Laser in a chaotic state, which is approximately described by three coupled nonlinear ordinary differential equations. The data set contains 1000 data points, followed by a continued data set containing 9093 data points. In the experiment, 12 data points are used as inputs to predict the current data point. 100 data patterns randomly selected from the whole data set is used as the validation set. The remaining data patterns are used as the training set. The first 100 data points in the continued data set are used as the test set. So there are a total of 888 data patterns in the training set, 100 data patterns in the validation set, and 100 data patterns in the test set.

The Gaussian kernel is still used for KPCA and SVM. The function of G in ICA is also used as $-(1/\gamma_2)\exp(-\gamma_2 s^2/2)$ according to the validation set. The values of the Gaussian parameter δ^2, γ_2 and the free parameters of SVM that produce the smallest error NMSE on the validation set are used, which are described in Table 2. Moreover, the same procedure as used in the sunspot data set is used to select the optimal number of principal components in PCA, KPCA and ICA. The values used are described in Table 2.

The best results obtained in Santa Fe data set A are given in Table 2. Apparently, SVM by feature extraction using PCA, KPCA or ICA can always converge to a smaller NMSE on the test set than that without feature extraction. Among all the methods, there is the smallest error on KPCA feature extraction, followed by ICA feature extraction. Furthermore, there is also a larger number of principal components in KPCA compared to those of PCA and ICA.

Table 2
The parameters used and the converged NMSE in the Santa Fe data set A

Methods	$(\delta^2, C, \varepsilon, \delta^2(\gamma_2))$	n	NMSE
Original + SVM	$(10, 1000, 0.01)$	—	0.0197
PCA + SVM	$(10, 100, 0.05)$	8	0.0129
KPCA + SVM	$(0.01, 1000, 0.05, 0.0001)$	59	0.0111
ICA + SVM	$(0.01, 100, 0.01, 0.8)$	11	0.0121

Table 3
Five financial futures contracts and their used time periods

Futures	Time period
CME-SP	24/05/1989–08/10/1993
CBOT-US	23/05/1991–20/10/1995
CBOT-BO	23/05/1991–17/10/1995
EUREX-BUND	31/05/1991–25/10/1995
MATIF-CAC40	18/10/1993–09/04/1998

4.3. Financial data sets

Five real financial futures contracts collated from the Chicago Mercantile Market are also examined. They are the Standard& Poor 500 stock index futures (CME-SP), United Sates 30-year government bond (CBOT-US), United States 10-year government bond (CBOT-BO), German 10-year government bond (EUREX-BUND) and French government stock index futures (MATIF-CAC40). The time periods used in these futures are listed in Table 3. The daily closing prices are used as the data set.

The original closing price is transformed into a five-day relative difference in percentage of price (RDP). The advantages of applying this transformation are described in the previous paper [30]. The input variables are constructed from 3 lagged transformed closing prices (x_1, x_2, x_3) which is obtained by subtracting a 15-day exponential moving average from the closing price, 14 lagged RDP values based on 5-day periods $(x_4, \ldots, x_{17})$, and 3 technical indicators: the moving average convergence divergence (MACD), on balance volume (OBV), and volatility (x_{18}, x_{19}, x_{20}). The subtraction in the transformed closing prices is performed to eliminate the trend in price as there is a ratio of the maximum value to the minimum value about 2:1 in the data set. The optimal length of the moving average is not critical, but it should be longer than the forecasting horizon of 5 days [31]. The use of the transformed closing price is to maintain the information contained in the original closing price as much as possible, as the application of the RDP transform to the original closing price may remove some useful information. MACD is defined as the difference of two exponential moving averages, and it is commonly used to predict market trends in financial markets. OBV moves in the same direction as price. That is, as price increases, the OBV will gain magnitude. As OBV can relate the volume into price, it is also used as input here.

Table 4
Input and output variables

Indicator	Calculation
x_1	$P(i) - \overline{\mathrm{EMA}_{15}(i)}$
x_2	$P(i-1) - \overline{\mathrm{EMA}_{15}(i-1)}$
x_3	$P(i-2) - \overline{\mathrm{EMA}_{15}(i-2)}$
x_4	$(p(i) - p(i-5))/p(i-5) * 100$
x_5	$(p(i-1) - p(i-6))/p(i-6) * 100$
x_6	$(p(i-2) - p(i-7))/p(i-7) * 100$
x_7	$(p(i-3) - p(i-8))/p(i-8) * 100$
x_8	$(p(i-4) - p(i-9))/p(i-9) * 100$
x_9	$(p(i) - p(i-10))/p(i-10) * 100$
x_{10}	$(p(i-1) - p(i-11))/p(i-11) * 100$
x_{11}	$(p(i-2) - p(i-12))/p(i-12) * 100$
x_{12}	$(p(i) - p(i-15))/p(i-15) * 100$
x_{13}	$(p(i-1) - p(i-16))/p(i-16) * 100$
x_{14}	$(p(i-2) - p(i-17))/p(i-17) * 100$
x_{15}	$(p(i) - p(i-20))/p(i-20) * 100$
x_{16}	$(p(i-1) - p(i-21))/p(i-21) * 100$
x_{17}	$(p(i-2) - p(i-22))/p(i-22) * 100$
x_{18}	$\overline{\mathrm{EMA}_{10}(i)} - \overline{\mathrm{EMA}_{20}(i)}$
x_{19}	$\begin{cases} p(i) \geqslant p(i-1) & \text{obv} += \text{volume}(i) \\ p(i) \leqslant p(i-1) & \text{obv} -= \text{volume}(i) \end{cases}$
x_{20}	$k * \mathrm{sqrt}(1/n * \sum_{i=1}^{n} \log^2(h(i)/l(i)))\ (k = 80, n = 5)$
RDP + 5	$(\overline{p(i+5)} - \overline{p(i)})/\overline{p(i)} * 100$ $\overline{p(i)} = \overline{\mathrm{EMA}_3(i)}$

$\mathrm{EMA}_n(i)$ is the n-day exponential moving average of the ith day. $p(i)$, $h(i)$, $l(i)$ are the closing, highest and lowest price of the ith day.

Volatility denotes the range of the highest price and lowest price in one day, and it is often used as a measure of the market risk. The output variable RDP+5 is obtained by first smoothing the closing price with a 3-day exponential moving average, because the application of a smoothing transform to the dependent variable generally enhances the prediction performance [32]. The calculations for all the indicators are given in Table 4.

The data is further preprocessed by using the following two techniques. As outliers may make it difficult to arrive at an effective solution for SVM, RDP values beyond the limits of ± 2 standard deviations are selected as outliers. They are replaced with the closest marginal values. The other preprocessing technique is data scaling. All the data points are scaled into the range of $[-0.9,\ 0.9]$ as the data points include both positive values and negative values. All the five data sets are partitioned into the training set, the validation set, and the testing set according to the time sequence. There are a total of 907 data patterns in the training set, and 200 data patterns in both the validation set and the test set in each of the data sets.

In all the data sets, the Gaussian kernel is still used for KPCA and SVM. The function of G in ICA is used as $-(1/\gamma_2)\exp(-\gamma_2 s^2/2)$ according to the validation set.

Table 5
The parameters used in five financial futures contracts

Data sets	Original + SVM $(\delta^2, C, \varepsilon)$	PCA + SVM $(\delta^2, C, \varepsilon)$	KPCA + SVM $(\delta^2, C, \varepsilon, \delta^2)$	ICA + SVM $(\delta^2, C, \varepsilon, \gamma_2)$
CME-SP	(1000,100,0.05)	(10,10,0.01)	(100,100,0.05,0.01)	(10,1,0.05,1.0)
CBOT-US	(100,10,0.05)	(10,10,0.01)	(100,1,0.05,0.01)	(10,1,0.05, 0.5)
CBOT-BO	(1000,1000,0.05)	(10000,100,0.05)	(1000,1000,0.05,0.01)	(10000,10,0.05,0.5)
EUREX-BUND	(1000,1000,0.05)	(1000,1000,0.05)	(10,1,0.05,0.01)	(1000,100,0.05,1.0)
MATIF-CAC40	(100,100,0.05)	(1000,100,0.05)	(100,100,0.05,0.01)	(1000,1,0.05,1.0)

Table 6
The number of principal components used and the converged NMSE in five financial futures contracts

Data sets	Original + SVM	PCA + SVM		KPCA + SVM		ICA + SVM	
	NMSE	*n*	NMSE	*n*	NMSE	*n*	NMSE
CME-SP	0.8775	10	0.8681	250	0.6988	7	0.8451
CBOT-US	0.9582	9	0.9392	253	0.8642	7	0.8973
CBOT-BO	0.9715	10	0.9683	300	0.8855	9	0.9553
EUREX-BUND	0.8339	11	0.8264	207	0.7089	10	0.8221
MATIF-CAC40	0.9212	10	0.9166	211	0.7698	19	0.8714
t-value	—		3.1388		7.3078		3.6220
	3.1388		—		6.5363		3.1988
	7.3078		6.5363		—		4.8086

Note: $t_{0.025,4} = 2.776$, $t_{0.005,4} = 4.604$, $t_{0.0025,4} = 5.598$ and $t_{0.001,4} = 7.173$.

The optimal values of the free parameters of KPCA, ICA and SVM chosen based on the validation set are given in Table 5. In the implementation of PCA, KPCA and ICA for feature extraction, the same procedure as used in the previous experiments is used to choose the optimal number of principal components, as described in Table 6.

The best results obtained in all the data sets are given in Table 6. Obviously, there is the same conclusion as obtained in the previous experiments. That is, SVM by feature extraction using PCA, KPCA or ICA can always converge to a smaller NMSE on the test set than that without feature extraction. Among all the methods, there is the best performance in KPCA feature extraction, followed by KPCA feature extraction. Furthermore, there is also a larger number of principal components in KPCA compared to those of PCA and ICA.

A paired *t*-test is performed to determine whether there is significant difference between each feature set in terms of the converged NMSE on the test set. The calculated *t*-values are illustrated in Table 6. The result shows that PCA and ICA both outperform the original feature set without feature extraction with $a = 2.5\%$ significance level for a one-tailed test, and KPCA outperforms the original feature set with $a{=}0.1\%$ significance level. This indicates that PCA, KPCA and ICA are all effective in SVM for feature extraction. The result also shows that KPCA and ICA respectively outperforms PCA

Table 7
The CPU time used in PCA, KPCA and ICA

Data sets	PCA	KPCA	ICA
Sunspot	5	2568	25
Santa Fe A	35	234013	66
CME-SP	75	1625580	267
CBOT-US	67	1937101	129
CBOT-BO	66	2262209	792
EUREX-BUND	74	2939517	163
MATIF-CAC40	66	1991286	325

with $a = 0.25\%$ and 2.5% significance level. And KPCA also outperforms ICA with $a = 0.5\%$ significance level.

Finally, for all the data sets, the CPU time used in PCA, KPCA and ICA is also calculated and reported in Table 7. Apparently, the time spent in KPCA is much larger than those of PCA and ICA. The reason lies in that the number of training data points is much larger than the input dimension in all the data sets studied, therefore, KPCA needs to estimate a larger number of principal components (equal to l) than those of PCA and ICA (equal to m).

5. Conclusions

In this paper, PCA, KPCA and ICA are applied to SVM for feature extraction. PCA linearly transforms the original inputs into uncorrelated new features, while in ICA the linearly transformed features are statistically independent. KPCA is a nonlinear PCA by generalizing the kernel method into linear PCA. The sunspot data, Satan Fe data set A and five real futures contract are examined in the experiment. The simulation shows that SVM by feature extraction using PCA, KPCA or ICA can achieve better generalization performance than that without feature extraction. This demonstrates the fact that dimension reduction by PCA, KPCA or ICA can improve the generalization performance of SVM.

The experiment also shows that KPCA and ICA perform better than PCA on all the studied data sets, with the best performance in KPCA. The reason lies in the fact that KPCA and ICA can explore higher order information of the original inputs than PCA. Instead of the sample covariance matrix (the second-order information) as used in PCA, the negentropy in ICA could take into account the higher order information of the original inputs, as described in Eq. (10). By using the kernel method to generalize PCA into nonlinear, KPCA also implicitly takes into account the high order information of the original inputs. More number of principal components could also be extracted in KPCA, eventually resulting in the best generalization performance.

In this paper, the approach of choosing the principal components is not global optimal as only a subset of all the combinations of principal components are investigated. Future work needs to explore the approach of choosing the optimal principal components.

Furthermore, the high generalization performance of KPCA is obtained at the cost of a large amount of computation time. Future work also needs to explore ways of reducing the large computation cost involved in KPCA.

Acknowledgements

I would like to thank the Institute of High Performance Computing for the excellent facilities without which the present work would not have been possible. I also would like to thank all the reviewers for their precious comments.

References

[1] A. Bell, T.J. Sejnowski, An information maximization approach to blind separation and blind deconvolution, Neural Comput. 7 (1995) 1129–1159.

[2] A.J. Bell, T.J. Sejnowski, The ‘independent components’ of natural scenes are edge filters, Vision Res. 37 (23) (1997) 3327–3338.

[3] P.S. Bradley, O.L. Mangasarian, Feature selection via concave minimization and support vector machines, in: Proceedings of the 15th International Conference on Machine Learning, 1998, Madison, WI, USA, pp. 82–90.

[4] P. Comon, Independent component analysis, a new concept? Signal Process. 36 (3) (1994) 287–314.

[5] M.B. Cottrell, Y. Girard, M. Mangeas, C. Muller, Neural modeling for time series: a statistical stepwise method for weight elimination, IEEE Trans. Neural Networks 6 (6) (1995) 1355–1364.

[6] K.I. Diamantaras, S.Y. Kung, Principal Component Neural Networks, Wiley, New York, 1996.

[7] X. Giannakopoulos, J. Karhunen, E. Oja, An experimental comparison of neural algorithms for independent component analysis and blind separation, Int. J. Neural Syst. 9 (2) (1999) 99–114.

[8] M. Girolami, C. Fyfe, Generalized independent component analysis through unsupervised learning with emergent bussgang properties, Proceedings of the 1997 IEEE International Conference on Neural Networks, 1997, Houston, Texas, pp. 2147–2152.

[9] I. Guyon, J. Weston, S. Barnhill, V. Vapnik, Gene selection for cancer classification using support vector machines, Mach. Learning 46 (1/3) (2002) 389–422.

[10] P.O. Hoyer, A. Hyvärinen, Independent component analysis applied to feature extraction from colour and stereo images, Network: Comput. Neural Syst. 11 (3) (2000) 191–210.

[11] A. Hyvärinen, Fast and robust fixed-point algorithms for independent component analysis, IEEE Trans. Neural Networks 10 (3) (1999) 626–634.

[12] A. Hyvärinen, E. Oja, Independent component analysis: algorithms and applications, Neural Networks 13 (4–5) (2000) 411–430.

[13] G.J. Jang, S.J. Yun, Y. Hwan, Feature vector transformation using independent component analysis and its application to speaker identification, in: Proceedings of Eurospeech, 1999, Budapest Hungary, pp. 767–770.

[14] I.J. Jolliffe, Principal Component Analysis, Springer, New York, 1986.

[15] J. Karhunen, E. Oja, L. Wang, R. Vigario, J. Joutsensalo, A class of neural networks for independent component analysis, IEEE Trans. Neural Networks 8 (1997) 486–504.

[16] H.W. Kuhn, A.W. Tucker, Nonlinear programming, in: Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probabilisticsn, 1951, Berkeley, pp. 481–492.

[17] J. Mercer, Functions of positive and negative type and their connection with the theory of integral equations, Philos. Trans. Roy. Soc. London A 209 (1909) 415–446.

[18] S. Mukherjee, E. Osuna, F. Girosi, Nonlinear prediction of chaotic time series using support vector machines, in: NNSP ’97: Neural Networks for Signal Processing VII: Proceedings of the IEEE Signal Processing Society Workshop, 1997, Amelia Island, FL, USA, pp. 511–520.

[19] K.R. Muller, A.J. Smola, G. Ratsch, B. Scholkopf, J. Kohlmorgen, Using support vector machines for time series prediction, in: B. Scholkopf, C.J.C. Burges, A.J. Smola (Eds.), Advances in Kernel Methods—Support Vector Learning, MIT Press, Cambridge, MA, 1999, pp. 243–254.

[20] K.R. Muller, J.A. Smola, G. Ratsch, B. Scholkopf, J. Kohlmorgen, V.N. Vapnik, Predicting time series with support vector machines, in: ICANN '97: Proceedings of the Seventh International Conference on Artificial Neural Networks, 1997, Lausanne, Switzerland, pp. 999–1004.

[21] E. Oja, The nonlinear PCA learning rule and signal separation—mathematical analysis, Neurocomputing 17 (1997) 25–45.

[22] R. Rosipal, M. Girolami, L.J. Trejo, A. Cichocki, Kernel PCA for feature extraction and de-noising in non-linear regression, Neural Comput. Appl. 10 (3) (2001) 231–243.

[23] B. Scholkopf, A. Smola, K.R. Muller, Nonlinear component analysis as a Kernel eigenvalue problem, Neural Comput. 10 (1998) 1299–1319.

[24] A.J. Smola, Learning with Kernels, Ph.D. Thesis, GMD, Birlinghoven, Germany, 1998.

[25] A.J. Smola, B. Schölkopf, A Tutorial on support vector regression, NeuroCOLT Technical Report NC-TR-98-030, Royal Holloway College, University of London, UK, 1998.

[26] F.E.H. Tay, L.J. Cao, Application of support vector machines in financial time series forecasting, Omega 29 (4) (2001) 309–317.

[27] F.E.H. Tay, L.J. Cao, Improved financial time series forecasting by combining support vector machines with self-organizing feature map, Intell. Data Anal. 5 (2001) 1–16.

[28] F.E.H. Tay, L.J. Cao, Saliency analysis of support vector machines for feature selection, Neural Network World 2 (1) (2001) 153–166.

[29] F.E.H. Tay, L.J. Cao, A comparative study of saliency analysis and genetic algorithm for feature selection in support vector machines, Intell. Data Anal. 5 (3) (2001) 191–209.

[30] F.E.H. Tay, L.J. Cao, Modified support vector machines in financial time series forecasting, Neurocomputing 48 (2002) 847–861.

[31] M. Thomason, The practitioner methods and tool, J. Comput. Intell. Finance 7 (4) (1999) 35–45.

[32] M. Thomason, The practitioner methods and tool, J. Comput. Intell. Finance 7 (6) (1999) 35–45.

[33] H. Tong, K.S. Lim, Threshold autoregressive, limit cycles and cyclical data, J. Roy. Stat. Soc. Ser. B 42 (3) (1980) 245–292.

[34] V.N. Vapnik, The Nature of Statistical Learning Theory, Springer, New York, 1995.

[35] A.S. Weigend, B.A. Huberman, D.E. Rumelhart, Predicting the future: a connectionist approach, Int. J. Neural Syst. 1 (1990) 193–209.

[36] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, V.N. Vapnik, Feature selection for SVMs, Adv. Neural Inform. Process. Systems 13 (2001) 668–674.

Dr L.J. Cao received her Ph.D. in Mechanical Engineering from the National University of Singapore. She is currently working as a Post-doctoral Research Fellow in the Data Analysis and Data Intensive Computing Group in the Institute of High Performance Computing. Her research area centers on artificial intelligence methods such as neural networks, genetic algorithms, and support vector machines.

Dr K.S. Chua obtained his Ph.D. from the Department of Mathematics, Stanford University in 1988 under Robert Finn on capillary surfaces. He has taught at the National University of Singapore and the German Singapore Institute and was previously group leader in data mining in the Data Analysis and Data Intensive Computing group at the Institute of High Performance Computing, Singapore. His main research interest is the application of numeric and symbolic computations and mathematics to solve scientific and engineering problems. Recently, he has been working on data analysis and some parts of number theory.

Dr W.K. Chong is currently the Senior Manager (IT, Research and Industry) of the Singapore-MIT Alliance (SMA). He has held position as the Programme Manager for High-End Computing Programme in the Institute of High Performance Computing prior to the current job at SMA.

Dr Chong obtained his Ph.D. and B. Eng. from the Department of Mechanical Engineering, National University of Singapore. His research interests are in the areas of Large Scale Simulations, Parallel and Distributed Computing, Artificial Intelligence techniques in Modelling and Simulation, Visualisation, Fluid-Structure Interaction, Computational Fluid Dynamics, Boundary Layer Stability, Bubble Dynamics and Data Mining.

Dr H.P. Lee is currently the Deputy Executive Director (Research) of the Institute of High Performance Computing (IHPC) as well as an Associate Professor at the Department of Mechanical Engineering, National University of Singapore. He has held position as the Sub-dean for External Relations in the Faculty of Engineering, National University of Singapore prior to taking up the appoint at IHPC.

Dr Lee obtained his Ph.D. and M.S. from the Department of Mechanical Engineering, Stanford University, B.A. from the University of Cambridge. His research interests are in the areas of vibration and acoustics, simulation techniques, metal forming, MEMS and data mining.

Dr Q.M. Gu received his Ph.D. in the Institute of Automation from the Southeast University, People's Republic of China. He is currently working as Project Consultant in the Office of Nanjing Municipal Government, People's Republic of China. His research area centers on data mining, genetic algorithms, and support vector machines.