Learning Linear and Nonlinear PCA with Linear Programming

Neural Processing Letters

ISSN 1370-4621
Volume 33
Number 2

Neural Process Lett (2011)
33:151-170
DOI 10.1007/
s11063-011-9170-4

Springer

Neural Process Lett (2011) 33:151–170
DOI 10.1007/s11063-011-9170-4

Learning Linear and Nonlinear PCA with Linear Programming

Rui Zhang · Wenjian Wang

Published online: 23 January 2011

Abstract An SVM-like framework provides a novel way to learn linear principal component analysis (PCA). Actually it is a weighted PCA and leads to a semi-definite optimization problem (SDP). In this paper, we learn linear and nonlinear PCA with linear programming problems, which are easy to be solved and can obtain the unique global solution. Moreover, two algorithms for learning linear and nonlinear PCA are constructed, and all principal components can be obtained. To verify the performance of the proposed method, a series of experiments on artificial datasets and UCI benchmark datasets are accomplished. Simulation results demonstrate that the proposed method can compete with or outperform the standard PCA and kernel PCA (KPCA) in generalization ability but with much less memory and time consuming.

Keywords Principal component analysis · Linear programming · Support vector machine

1 Introduction

Principal component analysis is a powerful technique for extracting structure from possibly high-dimensional data sets, and it has received much more attentions in many literatures [2,5,6,9,10]. From the view point of mathematics, PCA is an orthogonal transformation of the coordinate system in which the data are described. The new coordinate values, by which the data are represented, are called principal components. It is often the case that a small number of principal components are sufficient to account for most of the structure embedding in data.

R. Zhang
School of Science, Shandong University of Technology, Zibo 255049, P. R. China

W. Wang (✉)
Key Laboratory of Computational Intelligence & Chinese Information Processing of Ministry of Education, School of Computer and Information Technology, Shanxi University, Taiyuan 30006, P. R. China
e-mail: wjwang@sxu.edu.cn

Support vector machine (SVM) introduced by Vapnik [16,17] is an elegant tool for solving pattern recognition and regression problems, and has been demonstrated to be very valuable for real-world applications [4,11,12,15]. Over the past few years, some researchers try to apply SVM frame to other fields, such as PCA and one class problem [3,13,14] etc. As far as the relationship between SVM and PCA is concerned, a kernel method for directly performing a nonlinear form of PCA was developed by Schölkopf et al. [1] in 1998. A simple and straightforward primal-dual SVM formulation to PCA in dual variables, which is similar to least-squares SVM (LS-SVM, see Ref. [7]), was presented by Suykens et al. [8] in 2003. Tao et al. [13] provided a novel way to learn linear PCA in 2007. They established an SVM-like framework for PCA called SVPCA, where new expected risk, linear separability and margin were defined. The concept of PCA margin may lead to a new convex semi-definite optimization problem and as a result, the generalization bound of SVPCA can be analyzed under the frame of statistical learning theory. The robustness of SVPCA follows from the soft idea in SVMs as well. Since the SDP problem is convex, the corresponding learning algorithms have no local minima. In fact, Tao et al. only solved the problem about the first principal component line in linear PCA, as for the other principal components and nonlinear PCA were not discussed.

In this paper, we learn linear and nonlinear PCA with linear programming problems, which are easy to be solved and can obtain the unique global solution. Moreover, we construct algorithms to obtain all principal components for PCA and KPCA. As we know, for the standard PCA and KPCA, if the data size is large, it would need a very large memory to store kernel matrix and a lot of time to calculate eigenvalues and corresponding eigenvectors. It should be emphasized that the proposed method essentially attempts to use partial data points to determine all principal components. To verify the performance of the proposed method, a series of experiments on artificial datasets and UCI benchmark datasets are carried out. Simulation results demonstrate that the proposed method can compete with or outperform the standard PCA and KPCA in generalization ability but with much less memory and time consuming.

The paper is organized as follows. In Sect. 2, the standard PCA, KPCA and the SVM-like framework of linear SVPCA [13] are described briefly. In Sect. 3, the approach for learning linear PCA with a linear programming problem is introduced, and the determining of the first principal component by using partial points is explained in detail. Meanwhile, the corresponding constructed algorithm is provided. Similarly, the approach and corresponding algorithm for KPCA by using partial data points are illustrated in Sect. 4. Simulation experiments and discussions are presented in Sect. 5. The last section concludes the proposed works.

2 Preliminary Knowledge

For the sake of readability of the following sections, we briefly introduce the standard PCA, KPCA and the main results of linear SVPCA [13].

2.1 The Standard PCA and Kernel PCA

2.1.1 The Standard PCA

Principal components analysis takes an initial subset of the principal axes of the training data and projects the data into the space spanned by the set of eigenvectors. A set of data are projected into the subspace spanned by the first k eigenvectors of the covariance matrix of

the training set for some $k < l$ (l is the number of training data). The new coordinates are known as the principal coordinates with the eigenvectors referred to the principal axes.

The primal principal components analysis algorithm performs the following computation.

Input:
- a data set $S = \{x_i\}_{i=1}^{l}$, dimension k.

Process:
- $\mu = \frac{1}{l}\sum_{i=1}^{l} \mathbf{x}_i$
- $C = \frac{1}{l}\sum_{i=1}^{l}(\mathbf{x}_i - \mu)(\mathbf{x}_i - \mu)'$
- $[U, \bigwedge] = eig(lC)$
- $\tilde{x}_i = U_k' x_i, i = 1, \ldots, l$

Output:
- Transformed data $\tilde{S} = \{\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_l)$

2.1.2 The Standard KPCA

Kernel PCA is the application of PCA in a kernel—defined feature space making use of the dual representation. The kernel PCA algorithm performs the following computation.

Input:
- a data set $S = \{x_i\}_{i=1}^{l}$, dimension k.

Process:
- $K_{ij} = k(x_i, x_j), i, j = 1, \ldots, l$
- $K = K - \frac{1}{l}\mathbf{j}\mathbf{j}'K - \frac{1}{l}K\mathbf{j}\mathbf{j}' + \frac{1}{l^2}(\mathbf{j}'K\mathbf{j})\mathbf{j}\mathbf{j}'$, where $\mathbf{j}$ is the all 1s vector.
- $[V, \bigwedge] = eig(K)$
- $\alpha^j = \lambda_j^{-1/2} v_j, j = 1, \ldots, k$
- $\tilde{x} = \left(\sum_{i=1}^{l} \alpha_i^j k(x_i, x)\right)_{j=1}^{k}$

Output:
- Transformed data $\tilde{S} = \{\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_l)$

2.2 Brief Introduction of Linear SVPCA

In Ref. [13], a complete SVM-like framework to learn the direction of linear PCA is presented, where new expected risk, linear separability and margin are defined.

Consider a given training data set $S = \{\mathbf{x}_i\}_{i=1}^{l}$, where $\mathbf{x}_i \in R^n$ is independently drawn from a distribution with density $p(\mathbf{x})$. Define $f : R \to R^n$, $f(t) = \mathbf{x}_0 + t\mathbf{e}$ where $t \in R, x_0, e \in R^n$ and $\|\mathbf{e}\| = 1$. Let the hypothesis space H be a set of all such functions and $\triangle(\mathbf{x}, f) = \{min\|\mathbf{x} - \mathbf{x}_0 - t\mathbf{e}\|^2 : t \in R\}$

Definition 2.1 (Loss function, expected risk and empirical risk). Let $f \in H_0 = \{f(t) = \mathbf{x}_0 + t\mathbf{e} : \mathbf{x}_0 = \frac{1}{l}\sum_{i=1}^{l}\mathbf{x}_i, \|\mathbf{e}\| = 1, f : R \to R^n\}$. L is defined as follows:

$$L(\eta, \mathbf{x}, f) = \begin{cases} 0, & \text{if } \triangle(\mathbf{x}, f) \le \eta \\ 1, & \text{otherwise} \end{cases}$$

L is called the loss function of f about an $\eta - PCA$ problem, and

$$errD(f,\eta) = \int L(\eta, \mathbf{x}, f)p(\mathbf{x})d\mathbf{x}$$

$$err_{emp}(f,\eta) = \frac{1}{l}\sum_{i=1}^{l} L(\eta, \mathbf{x}_i, f)$$

are called the expected and empirical risks of f about the $\eta - PCA$ problem, respectively.

Definition 2.2 (The optimal component). If there exists a $f_0 \in H_0$ such that

$$err(f_0,\eta) = min\left\{\int L(\eta, \mathbf{x}, f)p(\mathbf{x})d\mathbf{x}, f \in H_0\right\}$$

Then f_0 is called the optimal component of the $\eta - PCA$ problem.

Definition 2.3 (Margin, separable and maximum margin component). Let $f \in H_0$. The margin of a sample $\mathbf{x}_i (i = 1, 2, \ldots, l)$ is defined as $m(f, \mathbf{x}_i) = \eta - \{min\|\mathbf{x}_i - \mathbf{x}_0 - t\mathbf{e}\|^2 : t \in R\}$. The margin of a hypothesis f is defined as $m(f, S) = min\{m(f, \mathbf{x}_i), i = 1, 2, \ldots, l\}$. If there exists a $f \in H_0$ such that $m(f, S) \geq 0$, then the $\eta - PCA$ problem is called separable. For a separable problem, if $f_0 \in H_0$ satisfies $m(f_0, S) = max\{m(f, S), f \in H_0\}$, then f_0 is called the maximum margin component.

In linearly separable cases, finding the maximal margin component can be described as solving the following optimization problem:

$$\begin{aligned} &\max_{\mathbf{e}\in R^n} \ \gamma \\ &s.t. \quad \mathbf{e}^T\mathbf{e} = 1, \\ &\qquad \eta - \{min\|\mathbf{x}_i - \mathbf{x}_0 - t\mathbf{e}\|^2 : t \in R\} \geq \gamma, i = 1, 2 \ldots, l. \end{aligned}$$

Let $r^2 = \eta - \gamma$, the above optimization problem can be reformulated as

$$\begin{aligned} &\max_{r,\mathbf{e}} \ r^2 \\ &s.t. \quad \mathbf{e}^T\mathbf{e} = 1 \\ &\qquad \mathbf{e}^T S_i \mathbf{e} \geq r_i^2 - r^2, i = 1, 2 \ldots, l \end{aligned} \tag{1}$$

where $S_i = (\mathbf{x}_i - \mathbf{x}_0)(\mathbf{x}_i - \mathbf{x}_0)^T, r_i = \| \mathbf{x}_i - \mathbf{x}_0 \|, \mathbf{x}_0 = \frac{1}{l}\sum_{i=1}^{l} \mathbf{x}_i$. Note for optimization problem (1), it is in fact looking for a specific $\mathbf{e}$ with the minimal margin.

Problem (1) can be changed into a 'dual' optimization problem, and stated as the following theorem.

Theorem 1 *Let* $\mathbf{e}_0$ *be the solution of problem (1). Then* $\mathbf{e}_0$ *must be an eigenvector of* $\sum_{i=1}^{l} \alpha_i^0 S_i$ *corresponding to the largest eigenvalue* β^0*, where* $\{\alpha_i^0, 1 \leq i \leq l\}$ *and* β^0 *is a solution of the following problem:*

$$\begin{aligned} \max_{\alpha,\beta} \ & \sum_{i=1}^{l} \alpha_i r_i^2 - \beta \\ s.t. \ & \sum_{i=1}^{l} \alpha_i S_i \mathbf{e}_0 = \beta \mathbf{e}_0 \ \ and\ \beta\ is\ the\ largest\ eigenvalue\ of\ \sum_{i=1}^{l} \alpha_i S_i \\ & \sum_{i=1}^{l} \alpha_i = 1, \alpha_i \geq 0, 1 \leq i \leq l \end{aligned} \tag{2}$$

where $\alpha = (\alpha_1, \ldots, \alpha_l)^T$.

Problem (2) is equivalent to the following optimization problem with linear matrix inequalities constraint:

$$\begin{aligned} \max_{\alpha,\beta} \ & \sum_{i=1}^{l} \alpha_i r_i^2 - \beta \\ s.t. \ & \sum_{i=1}^{l} \alpha_i S_i - \beta I \leq 0 \\ & \sum_{i=1}^{l} \alpha_i = 1, \alpha_i \geq 0, 1 \leq i \leq l \end{aligned}$$

where $\sum_{i=1}^{l} \alpha_i S_i - \beta I \leq 0$ represents that $\sum_{i=1}^{l} \alpha_i S_i - \beta I$ is negative semi-definite. This ia a standard SDP.

If there is no f in H_0 such that $m(f, S) \leq \eta$, then the η–PCA problem is not linearly separable. Similarly to SVM, by introducing the slack variables in constraints, for linearly inseparable PCA, they formulated the following optimization problem:[1]

$$\begin{aligned} \max_{e \in R^n, \xi} \ & \gamma - C \sum_{i=1}^{l} \xi_i \\ s.t. \ & \mathbf{e}^T \mathbf{e} = 1 \\ & \eta - \{min \parallel \mathbf{x}_i - \mathbf{x}_0 - t\mathbf{e} \parallel^2 : t \in R\} \geq \gamma - \xi_i \\ & i = 1, 2 \ldots, l; \xi \geq 0 \end{aligned} \tag{3}$$

Let $r^2 = \eta - \gamma$, the above optimization problem can be reformulated as

$$\begin{aligned} \min_{\{r^2, \mathbf{e}\}} \ & r^2 + C \sum_{i=1}^{l} \xi_i \\ s.t. \ & \mathbf{e}^T \mathbf{e} = 1 \\ & \mathbf{e}^T S_i \mathbf{e} \geq r_i^2 - r^2 - \xi_i, \ \xi_i \geq 0, \ i = 1, 2 \ldots, l \end{aligned} \tag{4}$$

where C is a predefined positive real number.

They obtained the following theorem.

[1] The original objective function of the optimization problem is wrong, we reformulated the objective function.

Theorem 2 *Let* $\mathbf{e}_0$ *be the solution of problem (3).*[2] *Then* $\mathbf{e}_0$ *must be an eigenvector of* $\sum_{i=1}^{l} \alpha_i^0 S_i$ *corresponding to the largest eigenvalue* β^0*, where* $\{\alpha_i^0, 1 \leq i \leq l\}$ *and* β^0 *is a solution of the following problem:*

$$\begin{aligned} &\max_{\{\alpha,\beta\}} \sum_{i=1}^{l} \alpha_i r_i^2 - \beta \\ &s.t. \sum_{i=1}^{l} \alpha_i S_i \mathbf{e}_0 = \beta \mathbf{e}_0 \quad \text{and} \quad \beta \text{ is the largest eigenvalue of } \sum_{i=1}^{l} \alpha_i S_i \\ &\sum_{i=1}^{l} \alpha_i = 1, 0 \leq \alpha_i \leq C, 1 \leq i \leq l \\ &\mathbf{e}_0^T \mathbf{e}_0 = 1 \end{aligned} \tag{5}$$

where $\alpha = (\alpha_1, \ldots, \alpha_l)^T$.

Note for optimization problem (3) and (5), it is in fact looking for a specific $\mathbf{e}_0$ which is the first principal component.

3 Learning Linear PCA with Linear Programming

In this section, we solve the optimization problem (5) with a linear programming problem, which is easy to be solved and can get the unique global solution.

Considering constrained optimization problem (5), let X_0 and X_1 be solution set and feasible solution set of (5), respectively. It is clear that $X_0 \subseteq X_1$. If X_1 contains only one point, then the point is the solution and $X_0 = X_1$.

The constraint conditions of (5) is as follows:

$$\begin{aligned} &(a) \quad \sum_{i=1}^{l} \alpha_i S_i \mathbf{e} = \beta \mathbf{e} \quad \text{and } \beta \text{ is the largest eigenvalue of } \sum_{i=1}^{l} \alpha_i S_i \\ &(b) \quad \sum_{i=1}^{l} \alpha_i = 1, 0 \leq \alpha_i \leq C, 1 \leq i \leq l \\ &(c) \quad \mathbf{e}^T \mathbf{e} = 1 \end{aligned} \tag{6}$$

Due to

$$\beta = \mathbf{e}^T \beta \mathbf{e} = \mathbf{e}^T \sum_{i=1}^{l} \alpha_i S_i \mathbf{e} = \sum_{i=1}^{l} \alpha_i \mathbf{e}^T S_i \mathbf{e}$$

the constraint conditions (6) can be equivalently transformed into the following optimization problem:

[2] $\mathbf{e}_0$ is an unit vector, we add $\mathbf{e}_0^T \mathbf{e}_0 = 1$ to constraint conditions of the optimization problem in the Theorem 2.

$$\begin{aligned} \max_{\{\alpha\}} \ & \sum_{i=1}^{l} \alpha_i \mathbf{e}^T S_i \mathbf{e} \\ s.t. \ & \sum_{i=1}^{l} \alpha_i = 1, \quad 0 \le \alpha_i \le C, 1 \le i \le l \\ & \mathbf{e}^T \mathbf{e} = 1 \end{aligned} \tag{7}$$

Maximizing $\sum_{i=1}^{l} \alpha_i \mathbf{e}^T S_i \mathbf{e}$ is equivalent to maximize $\mathbf{e}^T S_i \mathbf{e}$ for each nonnegative α_i [13]. Because $S_i = (\mathbf{x}_i - \mathbf{x}_0)(\mathbf{x}_i - \mathbf{x}_0)^T$ is a symmetric, semi-positive definite matrix and its rank is only one, it has only one non-zero eigenvalue c_i, and $c_i > 0$. In fact, $c_i = (\mathbf{x}_i - \mathbf{x}_0)^T (\mathbf{x}_i - \mathbf{x}_0) = \|\mathbf{x}_i - \mathbf{x}_0\|^2$ because $(\mathbf{x}_i - \mathbf{x}_0)(\mathbf{x}_i - \mathbf{x}_0)^T$ and $(\mathbf{x}_i - \mathbf{x}_0)^T (\mathbf{x}_i - \mathbf{x}_0)$ have the same non-zero eigenvalues. Therefore, the optimization problem (7) can be equivalently transformed into the following problem:

$$\begin{aligned} \max_{\{\alpha\}} \ & \sum_{i=1}^{l} c_i \alpha_i \\ s.t. \ & \sum_{i=1}^{l} \alpha_i = 1, \quad 0 \le \alpha_i \le C, 1 \le i \le l \end{aligned} \tag{8}$$

where $\alpha = (\alpha_1, \ldots, \alpha_l)^T$.

This is a linear programming problem. When $C < \frac{1}{l}$, Both problem (5) and problem (8) have no solution. When $C \ge \frac{1}{l}$, problem (8) has an unique global solution α^*. Hence, α^* is the solution of the optimization problem (5) also. If β^* is the largest eigenvalue of $\sum_{i=1}^{l} \alpha_i^* S_i$, $\mathbf{e}_0$ must be an unit eigenvector of β^*, just same as stated in Theorem 2.

Based on above discussions, one of our main results is expressed in Theorem 3.

Theorem 3 *Let $\alpha^* = (\alpha_1^*, \ldots, \alpha_l^*)^T$ be the solution of the linear programming problem (8), then the first principal component e_0 is the eigenvector corresponding to the maximum eigenvalue about*

$$\sum_{i=1}^{l} \alpha_i^* S_i = \sum_{\mathbf{x}_i \in SVs} \alpha_i^* S_i \tag{9}$$

where $S_i = (\mathbf{x}_i - \mathbf{x}_0)(\mathbf{x}_i - \mathbf{x}_0)^T$ and $SVs = \{\mathbf{x}_i | \alpha_i^* > 0, i = 1, 2, \ldots, l\}$.

Remark

- When $C \ge \frac{1}{l}$, problem (8) has an unique global solution. We prove this result. For simplicity, we assume that $C = \frac{1}{m}$ and m is an integer and smaller than l, and we assume that $c_1 \ge c2 \ge \cdots \ge c_m \ge c_{m+1} \ge c_l$, then $\alpha_1 = \alpha_2 = \cdots = \alpha_m = \frac{1}{m}$, and $\alpha_{m+1} = \alpha_{m+2} = \cdots = \alpha_l = 0$. This means that m farthest data points to mean center are selected.
- Suppose $\alpha^* = (\alpha_1^*, \ldots, \alpha_l^*)^T$ is the solution of linear problem (8). The linear problem (8) means that we select $\lfloor \frac{1}{C} \rfloor$ data points which are the farthest points away from the mean center x_0 and one can determine the first principal component by using these points, where $\lfloor x \rfloor$ means the minimum integer which is equal to x or bigger than x. So the proposed simplified principal components analysis (SPCA) is an approximating method. When all data points are considered, the proposed SPCA is the same as the classical PCA.

- In standard PCA , all data points are used. For a large scale data set, it needs a very large memory to store kernel matrix and a lot of time to calculated eigenvalues and corresponding eigenvectors. For the algorithms of PCA, we can see that different data points play the different roles. In general, the closer the data points to the mean center, the less important the data points contribute to PCA because their projections to an arbitrary direction are very small. So we can discard those data points which are close to the mean center and use the rest data points to approximate the stand PCA.
- The essence of the original optimization problem (5) is in fact looking for an unit eigenvector e_0 which is the first principal component. Either $\alpha_1 = \alpha_2 = \cdots = \alpha_m = \frac{1}{m}$ or $\alpha_1 = \alpha_2 = \cdots = \alpha_m = 1 (\alpha_{m+1} = \alpha_{m+2} = \cdots = \alpha_l = 0)$, we have the same unit eigenvector e_0 corresponding to the maximum eigenvalue of $\sum_{i=1}^{l} \alpha_i S_i$. So instead of solving linear programming problem (8), we select a given number of points which are the farthest points away from the mean center among all given data and If $x_i \in SVs$, we let $\alpha_i = 1$, otherwise, we let $\alpha_j = 0$.

The proposed SPCA algorithm is summarized as follows:

Input:
- A data set $S = \{\mathbf{x}_i\}_{i=1}^{l}$, a given number m, and contribution ratio γ.

Process:
- $\mu = \frac{1}{l} \sum_{i=1}^{l} \mathbf{x}_i$
- Calculating $c_i = (\mathbf{x}_i - \mu)^T (\mathbf{x}_i - \mu)$
- Selecting m data points which are the farthest points away from the mean center. The set of SVs consists of these selected data.
- $B = \sum_{x_i \in SVs} (\mathbf{x}_i - \mathbf{x}_0)(\mathbf{x}_i - \mathbf{x}_0)^T$.
- [U, $\bigwedge$]=eig(B)
- Sorting descensively the eigenvalues $\{\lambda_i\}_{i=1}^{l}$ and rearranging their corresponding eigenvectors. For simplicity, we use the same symbols U and $\bigwedge$.
- Determining the dimension k according to the following formula (10) for a given contribution ratio γ.
- $x_i^* = U_k^T \mathbf{x}_i, i = 1, \cdots, l$

Output:
- $S^* = \{\mathbf{x}_1^*, \mathbf{x}_2^*, \cdots, \mathbf{x}_l^*\}$

Where [U, $\bigwedge$] = eig(B) is the characteristic decomposition of B, and U_k is a matrix composed of the first k columns of U. The dimension k can be determined according to the following inequality:

$$\frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{l} \lambda_i} \geq \gamma \tag{10}$$

where $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_l$ are eigenvalues of B, and γ is called contribution ratio, which is generally selected as 0.70–0.95.

4 Learning Nonlinear PCA with Linear Programming

In this section, we discuss kernel PCA through solving a linear programming problem.

For the input domain $\mathcal{X}$, a map

$$\Phi : \mathcal{X} \to \mathcal{H}, \mathbf{x} \mapsto \Phi(\mathbf{x}) \tag{11}$$

is usually nonlinear, and the feature space $\mathcal{H}$ could have an arbitrarily large (even infinite) dimension. Again, we assume that we are dealing with centered data, $\{\Phi(x_i) - \Phi_0\}_{i=1}^{l}$, where $\Phi_0 = \frac{1}{l}\sum_{i=1}^{l}\Phi(x_i)$ is mean center in feature space.

In this situation, we rewrite the linear programming problem (8) as follows:

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^{l} c_i \alpha_i \\ s.t. \quad & \sum_{i=1}^{l} \alpha_i = 1, C \geq \alpha_i \geq 0, 1 \leq i \leq l \end{aligned} \tag{12}$$

where $\alpha = (\alpha_1, \ldots, \alpha_l)^T$, c_i is the unique positive eigenvalue of

$$S_i = (\Phi(\mathbf{x}_i) - \Phi_0)(\Phi(\mathbf{x}_i) - \Phi_0)^T$$

and

$$\begin{aligned} c_i &= (\Phi(\mathbf{x}_i) - \Phi_0)^T (\Phi(\mathbf{x}_i) - \Phi_0) \\ &= K(\mathbf{x}_i, \mathbf{x}_i) - \frac{2}{l}\sum_{j=1}^{l} K(\mathbf{x}_i, \mathbf{x}_j) + \frac{1}{l^2}\sum_{i,j=1}^{l} K(\mathbf{x}_i, \mathbf{x}_j) \end{aligned} \tag{13}$$

where $K(\mathbf{x}, \mathbf{y}) = \Phi(\mathbf{x})^T \Phi(\mathbf{y})$ is a kernel function.

Let $\alpha^* = (\alpha_1^*, \ldots, \alpha_l^*)^T$ be the solution of (12). Without loss of generality, suppose $\alpha_1^* > 0, \ldots, \alpha_k^* > 0$, and $\alpha_{k+1}^* = \alpha_{k+2}^* = \cdots = \alpha_l^* = 0$.

By a transformation

$$\widetilde{\Phi} : \mathbf{x}_i \mapsto \sqrt{\alpha_i^*}(\Phi(\mathbf{x}_i) - \Phi_0) \quad i = 1, \ldots, k \tag{14}$$

the covariance matrix takes the form

$$\begin{aligned} \sum_{i=1}^{l} \alpha_i^* S_i &= \sum_{i=1}^{k} \alpha_i^* S_i = \sum_{i=1}^{k} \alpha_i^* (\Phi(\mathbf{x}_i) - \Phi_0)(\Phi(\mathbf{x}_i) - \Phi_0)^T \\ &= \sum_{i=1}^{k} \widetilde{\Phi}(\mathbf{x}_i)\widetilde{\Phi}^T(\mathbf{x}_i) \end{aligned} \tag{15}$$

The corresponding kernel matrix $\widetilde{K}$ is defined as:

$$\begin{aligned} \widetilde{K}_{ij} &= \widetilde{\Phi}^T(\mathbf{x}_i)\widetilde{\Phi}(\mathbf{x}_j) = \sqrt{\alpha_i^*}(\Phi(\mathbf{x}_i) - \Phi_0)^T \sqrt{\alpha_j^*}(\Phi(\mathbf{x}_j) - \Phi_0) \\ &= \sqrt{\alpha_i^* \alpha_j^*}\{K(\mathbf{x}_i, \mathbf{x}_j) - \frac{1}{l}\sum_{j=1}^{l} K(\mathbf{x}_i, \mathbf{x}_j) - \frac{1}{l}\sum_{i=1}^{l} K(\mathbf{x}_i, \mathbf{x}_j) \\ &\quad + \frac{1}{l^2}\sum_{i,j=1}^{l} K(\mathbf{x}_i, \mathbf{x}_j)\}, \quad for\ i, j = 1, \ldots, k \\ \widetilde{K}_{ij} &= 0, \quad for\ i, j = k+1, \ldots, l. \end{aligned} \tag{16}$$

Thus the kernel matrix is sparse, only k^2 entries in kernel matrix $\widetilde{K}$ need to be stored.

The proposed algorithm, namely the simplified kernel PCA (SKPCA), is summarized as follows.

Input:

- A data set $S = \{\mathbf{x}_i\}_{i=1}^{l}$, a given number m and contribution ratio γ.

Process:

- Calculating c_i according to (13)
- Selecting m data points which are the farthest points away from the mean center. The set of SVs consists of these selected data. If $x_i \in SVs$, we let $\alpha_i = 1$, otherwise, we let $\alpha_j = 0$.
- If $\mathbf{x}_i, \mathbf{x}_j \in SVs$, calculating $\widetilde{K}_{ij}$ according to (16)
- $[\bigvee, \bigwedge] = eig(\widetilde{K})$
- Sorting descensively the eigenvalues $\{\lambda_i\}_{i=1}^{l}$ and rearranging their corresponding eigenvalues. For simplicity, we use the same symbols $\bigvee$ and $\bigwedge$.
- Determining the dimension k according to the formula (10) for a given contribution ratio γ.
- $\beta^j = \frac{1}{\sqrt{\lambda_j}}\mathbf{v}_j,\ j = 1, \ldots, k$
- $\mathbf{x}_i^* = \{\sum_{\mathbf{x}_i \in SVs} \beta_i^j \widetilde{K}(\mathbf{x}_i, \mathbf{x})\}_{j=1}^{k}$

Output:

- $S^* = \{\mathbf{x}_1^*, \mathbf{x}_2^*, \ldots, \mathbf{x}_l^*\}$

5 Experiments and Discussions

To verify the proposed approaches, three examples are provided. The first one is used to illustrate the difference of the first principal component between SPCA and the standard PCA on two artificial and ten UCI benchmark datasets. $\|\mathbf{e}_0 - \mathbf{e}_1\|$ denotes that difference, where $\mathbf{e}_0$ and $\mathbf{e}_1$ represent two normal vectors corresponding to the first principal components of PCA and SPCA, respectively. To further testify the generalization performance and dimension reduction ability of SPCA and SKPCA, in the second and third examples, some comparison experiments are implemented on UCI data sets for classification problem. All UCI benchmark datasets used in the experiments are listed in Table 1. All experiments are carried out by using Matlab 7.01, and on a 2.0 GHz Intel(R) Pentium(R) D CPU machine with 1KMB RAM.

Example 1 We use two synthetic and ten UCI benchmark datasets to illustrate the difference between SPCA and the standard PCA. Two synthetic data sets are

(1) 100 data points with two attributes are generated from an uniform distribution within the rectangle $[-0.5, 0.5] \times [-0.5, 0.5]$.
(2) 200 data points with five attributes are generated from a normal distribution with mean zero and standard deviation one.

One, ten, twenty, to hundred percent of data are respectively used to determine the first principal component and the norm $err \triangleq \|\mathbf{e}_0 - \mathbf{e}_1\|$. The smaller the err, the closer the first principal component lines between SPCA and the standard PCA. The corresponding err results are listed in Table 2.

From Table 2, it can be observed that the difference $err \triangleq \|\mathbf{e}_0 - \mathbf{e}_1\|$ descends with increment of the percent of the data for almost all data sets. When the percent reaches 100, the

proposed SPCA is the same as the classical PCA, and $\mathbf{e}_0$ is equal to $\mathbf{e}_1$. For synthetic datasets (1) and (2), with only 10 and 40% of data the err can be lower than 0.1, which means the first principal component lines by SPCA and PCA are very close. For UCI benchmark datasets, in the case of $err < 0.1$, 1% for banana dataset, 10% for waveform, 20% for thyroid, 50% for image, 60% for titanic, 70% for breast-cancer and diabetic, 80% for heart, 90% for splice and German are needed. Therefore, only based on partial data, one can determine the first principal component line very well.

The first principal component lines determined by the proposed SPCA and the standard PCA are showed in Fig. 1. Here, synthetic data (1) and banana data set are selected.

From Fig. 1, it can be seen that the first principal component lines determined by the standard PCA and the proposed SPCA are close very well, but the latter only uses one, ten and twenty percents of data points among the whole data sets, respectively. This means the proposed SPCA method is very effective and practical.

Example 2 We apply the standard PCA and the proposed SPCA to classification problems. The accuracy of classification, dimension reduction, and CPU time of feature decomposition are verified on seven datasets from the UCI benchmark repository. In this example, contribution ratio γ is respectively selected as 0.7, 0.8 , 0.9. Gaussian kernel $k(x, y) = e^{\frac{-\|x-y\|^2}{2\sigma^2}}$ is used with $\sigma = 0.5$. The comparison results are listed in Tables 3, 4, 5 where mean and std represent the mean value and standard deviation, respectively.

From Tables 3, 4, 5, it can be seen that with different contribution ratios, the SPCA behaves very well for classification performance, dimension reduction and training time. When $\gamma = 0.7$, the experiment results on three datasets (splice, waveform and thyroid) by SPCA exceed that by the classical PCA, and there are slight difference on three datasets (German, heart and breast-cancer). Only on one dataset (image), the performance of SPCA is inferior to that of PCA. When $\gamma = 0.8$, comparing with classical PCA, the numbers of superior, equivalent and inferior performance on datasets by SPCA are respective four (splice, waveform, heart and breast-cancer), two (German, and thyroid) and one (image). When $\gamma = 0.9$, only the performance on image by SPCA is worse than that by PCA.

In general, one can select the contribution ratio $\gamma = 0.8$ for PCA by experience. The experiments also support that the choice of $\gamma = 0.8$ is appropriate. In this situation, except for image and German data sets, the proposed SPCA only needs 10% of data to determine

Table 1 The used benchmark data sets from UCI benchmark repository

Dataset	No. of attributes	No. of training set	No. of testing set	No. of examples
Banana	2	400	4900	1, 3
Titanic	3	150	2051	1, 3
Thyroid	5	140	75	1, 2, 3
Waveform	21	400	4600	1, 2, 3
Image	18	1300	1010	1, 2
Splice	60	1000	2175	1, 2, 3
Breast-cancer	9	200	77	1, 2 , 3
German	20	700	300	1, 2, 3
Heart	13	170	100	1, 2, 3
Diabetic	8	468	300	1, 3

Table 2 Differences between the standard PCA and SPCA (err)

Percent(%)	1	10	20	30	40	50	60	70	80	90	100
Synthetic (1)	0.1480	0.0550	0.0415	0.2358	0.3243	0.0069	0.1231	0.1191	0.0188	0.0379	$1.8073e-15$
Synthetic (2)	0.4890	0.4864	0.3828	0.2360	0.0806	0.0941	0.0518	0.0220	0.0327	0.0242	$2.3954e-15$
Banana	0.0540	0.0491	0.0230	0.0173	0.0550	0.0698	0.0285	0.0262	0.0146	0.0020	$2.2204e-16$
Thyroid	0.9032	0.1345	0.0412	0.0281	0.0196	0.0144	0.0082	0.0028	0.0011	$7.1541e-4$	$6.1815e-16$
Waveform	0.4405	0.0941	0.0635	0.0430	0.0219	0.0242	0.0184	0.0170	0.0106	0.0045	$4.6091e-16$
Image	1.3281	0.2834	0.2285	0.1707	0.1466	0.0932	0.0579	0.0307	0.0129	0.0038	$9.8952e-16$
Titanic	0.2745	0.3694	0.2934	0.3100	0.2644	0.1837	0.0986	0.0274	0.0285	0.0142	$3.0017e-15$
Splice	1.3818	1.1446	0.6586	0.4624	0.3718	0.2665	0.1981	0.1571	0.1013	0.0591	$3.6486e-15$
Breast-cancer	0.7778	0.4828	0.2505	0.1743	0.1438	0.1389	0.1423	0.0532	0.0543	0.0500	$3.6388e-15$
German	1.0562	0.8224	0.5240	0.4149	0.3620	0.2827	0.2487	0.1950	0.1047	0.0575	$2.9830e-15$
Diabetic	0.7506	0.4059	0.3133	0.2899	0.2022	0.1883	0.1403	0.0988	0.0445	0.0196	$2.5385e-15$
Heart	0.6452	0.6085	0.4398	0.3573	0.3182	0.2223	0.1684	0.1259	0.0843	0.0399	$8.1377e-16$

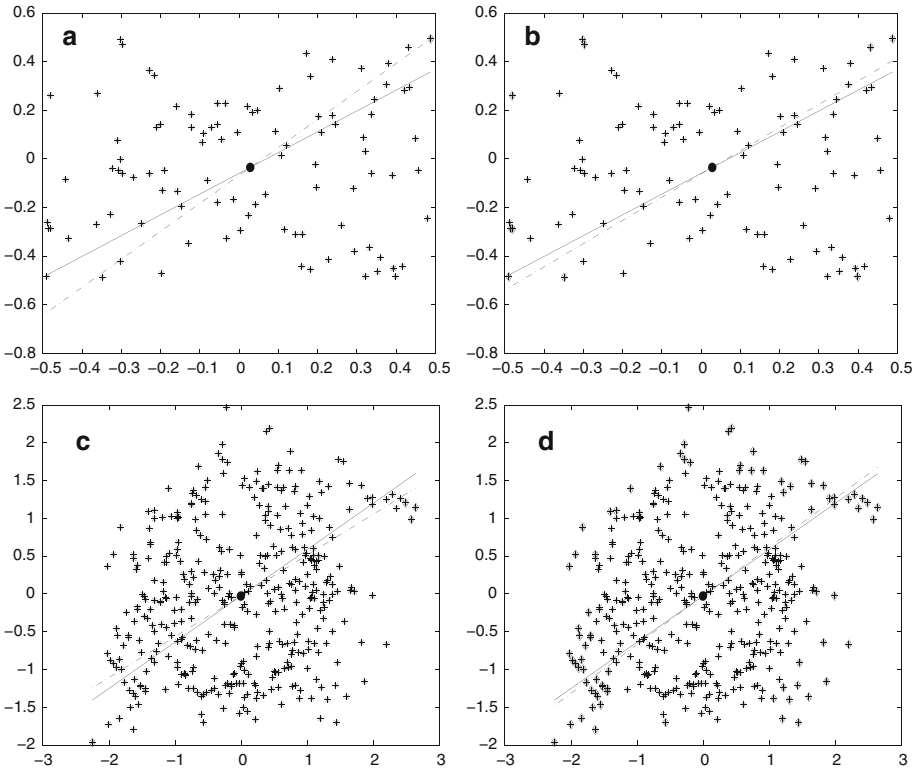

Fig. 1 The first principal component lines determined by SPCA and PCA. **a**, **b** are for synthetic data (1), and **c**, **d** are for Banana dataset. The *dashed line* and the *solid line* are the first principal component lines determined by SPCA and the standard PCA, respectively. The *solid point* is mean center. The *diamond points* are selected to determine the first principal component by SPCA. 1% of data points are used by SPCA in **a** (one data point) and **c** (four data points). 10 and 20% of data points are used by SPCA in **b** (ten data points) and **d** (eighty data points), respectively

all the principal components, meanwhile, one can obtain more higher accuracy, much less CPU time and more lower dimension than the PCA method does. For German data set, 40% of data are needed to determine all principal components and similar result can be obtained. For image data set, 80% of data are needed, and the accuracy is slight lower than that of PCA. The error between PCA and SPCA is 0.3%. All the experiment results in this section show that the proposed SPCA method could compete with or outperform the standard PCA in generalization performance but with much less memory and time consuming.

Example 3 We apply the standard KPCA and the proposed SKPCA to classification problems. Similarly, the classification performance, dimension reduction, and the CPU time are testified.

The contribution ratio γ is just selected as 0.8. In KPCA and SKPCA, Gaussian kernel $k(x, y) = e^{\frac{-\|x-y\|^2}{2\sigma^2}}$ is still used with $\sigma = 0.5$. The comparison results are listed in Tables 6.

From Table 6, it can be seen that the SKPCA behaves very well for classification performance, dimension reduction and training time similar to the SPCA does. When $\gamma = 0.8$, only on two datasets (banana and heart), the classification accuracy of the SKPCA is a little

Table 3 The comparison results between the standard PCA and SPCA with contribution ratio $\gamma = 0.7$

Dataset	Percent(%)	PCA	SPCA					
		100	10	20	40	60	80	Mean ± std
Splice	Dimension	34	1	1	1	1	1	1 ± 0.0000
	Accuracy	0.5200	0.5821	0.5830	0.5825	0.5834	0.5830	0.5828 ± 0.5050e-3
	CPU time	0.0080	0.0035	0.0035	0.0036	0.0035	0.0036	0.0035 ± 0.0548e-3
Waveform	Dimension	6	2	3	4	5	5	3.8000 ± 1.3038
	Accuracy	0.7159	0.8933	0.8624	0.8361	0.7798	0.7798	0.8303 ± 0.0503
	CPU time	$6.3966e-4$	$2.9623e-4$	$3.0551e-4$	$3.0039e-4$	$3.0653e-4$	$3.1397e-4$	0.0003 ± 0.0000
Image	Dimension	4	2	2	3	3	4	2.8000 ± 0.8367
	Accuracy	0.9139	0.7366	0.7287	0.8743	0.8842	0.9069	0.8261 ± 0.0862
	CPU time	$9.5011e-4$	$1.7931e-4$	$1.9139e-4$	$2.1304e-4$	$1.9757e-4$	$2.3161e-4$	0.0002 ± 0.0000
German	Dimension	11	6	8	10	10	11	9.0000 ± 2.0000
	accuracy	0.7233	0.6833	0.7200	0.7300	0.7267	0.7233	0.7167 ± 0.0190
	CPU time	0.0010	$2.7429e-4$	$2.8127e-4$	$3.0602e-4$	$3.0936e-4$	$3.5373e-4$	0.0003 ± 0.0000
Heart	Dimension	7	4	5	6	6	7	5.6000 ± 1.1402
	Accuracy	0.6500	0.6400	0.6300	0.6900	0.6300	0.6300	0.6440 ± 0.0261
	CPU time	$2.1616e-4$	$1.3418e-4$	$1.3812e-4$	$1.5044e-4$	$1.4287e-4$	$1.4674e-4$	0.0001 ± 0.0000
Breast-cancer	Dimension	5	3	3	4	4	5	3.8000 ± 0.8367
	Accuracy	0.7013	0.6883	0.6364	0.6623	0.6234	0.6623	0.6545 ± 0.0253
	CPU time	$4.5969e-4$	$8.6340e-5$	$8.9738e-5$	$8.7207e-5$	$1.1263e-4$	$9.1182e-5$	0.0001 ± 0.0000
Thyroid	Dimension	2	2	2	2	2	2	2 ± 0.0000
	Accuracy	0.9733	0.9733	0.9733	0.9733	0.9733	0.9733	0.9733 ± 0.0000
	CPU time	$7.6179e-5$	$5.2761e-5$	$5.3568e-5$	$5.2566e-5$	$5.3433e-5$	$5.3523e-5$	5.3e-5 ± 4.7e-7

Table 4 The comparison results between the standard PCA and SPCA with contribution ratio $\gamma = 0.8$

Dataset	Percent(%)	PCA	SPCA					
		100	10	20	40	60	80	Mean ± std
Splice	Dimension	42	1	1	1	1	1	1 ± 0.0000
	Accuracy	0.5200	0.5821	0.5830	0.5825	0.5834	0.5830	0.5828 ± 0.5050e-3
	CPU time	0.0077	0.0034	0.0036	0.0036	0.0036	0.0035	0.0035 ± 0.0894e-3
Waveform	Dimension	9	4	5	7	8	8	6.4000 ± 1.8166
	Accuracy	0.6724	0.8387	0.7820	0.6835	0.6735	0.6730	0.7301 ± 0.0760
	CPU time	$6.6701e-4$	$2.9743e-4$	$3.1551e-4$	$3.0560e-4$	$3.0954e-4$	$3.1448e-4$	0.0003 ± 0.0000
Image	Dimension	6	3	3	4	5	5	4.0000 ± 1.0000
	Accuracy	0.9297	0.8149	0.8089	0.8941	0.9178	0.9267	0.8725 ± 0.0566
	CPU time	0.0010	$1.7734e-4$	$1.8932e-4$	$2.3992e-4$	$1.9658e-4$	$2.1560e-4$	0.0002 ± 0.0000
German	Dimension	13	8	10	12	13	13	11.200 ± 2.1679
	Accuracy	0.7233	0.7033	0.7200	0.7267	0.7233	0.7233	0.7193 ± 0.0093
	CPU time	$8.4081e-4$	$2.7520e-4$	$2.8428e-4$	$3.0597e-4$	$3.1130e-4$	$5.1827e-4$	0.0003 ± 0.0001
Heart	Dimension	8	5	6	8	8	8	7.0000 ± 1.4142
	Accuracy	0.6100	0.7000	0.6800	0.6100	0.6100	0.6100	0.6420 ± 0.0444
	CPU time	$2.1322e-4$	$1.3434e-4$	$1.3873e-4$	$1.5002e-4$	$1.4307e-4$	$1.4882e-4$	0.0001 ± 0.0000
Breast-cancer	Dimension	6	4	4	5	6	6	5.0000 ± 1.0000
	Accuracy	0.6623	0.6753	0.6883	0.6883	0.6883	0.6753	0.6831 ± 0.0071
	CPU time	$4.6830e-4$	$8.5929e-5$	$8.9638e-5$	$8.9748e-5$	$9.1718e-5$	$9.2134e-5$	0.0001 ± 0.0000
Thyroid	Dimension	3	2	2	3	3	3	2.6000 ± 0.5477
	Accuracy	0.96	0.9733	0.9733	0.9467	0.9467	0.9467	0.9573 ± 0.0146
	CPU time	$7.5803e-5$	$5.1708e-5$	$5.1588e-5$	$5.3643e-5$	$5.3744e-5$	$5.3583e-5$	0.0001 ± 0.0000

Table 5 The comparison results between the standard PCA and SPCA with contribution ratio $\gamma = 0.9$

Dataset	Percent(%)	PCA	SPCA					
		100	10	20	40	60	80	Mean ± std
Splice	Dimension	50	5	8	10	12	14	9.8000 ± 3.4928
	Accuracy	0.5200	0.6676	0.6110	0.6000	0.5972	0.5926	0.6137 ± 0.0309
	CPU time	0.0078	0.0033	0.0034	0.0036	0.0035	0.0035	0.0035 ± 0.0001
Waveform	Dimension	14	8	10	12	13	14	11.4000 ± 2.4083
	Accuracy	0.6707	0.6754	0.6707	0.6707	0.6707	0.6707	0.6716 ± 0.0021
	CPU time	$6.5053e-4$	$3.0395e-4$	$3.0080e-4$	$2.9364e-4$	$3.0611e-4$	$3.0838e-4$	0.0003 ± 0.0000
Image	Dimension	8	5	5	7	7	8	6.4000 ± 1.3416
	Accuracy	0.9574	0.8554	0.8871	0.9089	0.9228	0.9307	0.9010 ± 0.0304
	CPU time	$9.5313e-4$	$1.8369e-4$	$2.0254e-4$	$1.9352e-4$	$1.9641e-4$	$1.9996e-4$	0.0002 ± 0.0000
German	Dimension	16	12	14	15	16	16	14.6000 ± 1.6733
	Accuracy	0.7200	0.7200	0.7233	0.7233	0.7200	0.7200	0.7213 ± 0.0018
	CPU time	$8.3575e-4$	$2.7957e-4$	$2.7855e-4$	$3.0671e-4$	$3.0177e-4$	$3.0201e-4$	0.0003 ± 0.0000
Heart	Dimension	10	7	8	10	10	10	9.0000 ± 1.4142
	Accuracy	0.6100	0.6400	0.6300	0.6100	0.6100	0.6100	0.6200 ± 0.0141
	CPU time	$2.1197e-4$	$1.3519e-4$	$1.3900e-4$	$1.4957e-4$	$1.4101e-4$	$1.4424e-4$	0.0001 ± 0.0000
Breast-cancer	Dimension	7	5	6	7	7	7	6.4000 ± 0.8944
	Accuracy	0.6753	0.6234	0.6883	0.6883	0.6753	0.6753	0.6701 ± 0.0269
	CPU time	$1.3630e-4$	$8.6039e-5$	$8.7051e-5$	$8.5888e-5$	$9.2545e-5$	$8.9262e-5$	0.0001 ± 0.0000
Thyroid	Dimension	4	3	3	4	4	4	3.6000 ± 0.5477
	Accuracy	0.9733	1.0000	0.9867	0.9733	0.9733	0.9733	0.9813 ± 0.0119
	CPU time	$7.7708e-5$	$5.2691e-5$	$5.2240e-5$	$5.3668e-5$	$5.2641e-5$	$5.3638e-5$	0.0001 ± 0.0000

Table 6 The comparison results between the standard KPCA and SKPCA with contribution ratio $\gamma = 0.8$

Dataset	Percent(%)	PCA	SPCA				
		100	10	20	40	60	80	Mean ± std
Banana	Dimension	16	8	11	14	15	16	12.8000 ± 3.2711
	Accuracy	0.8835	0.5853	0.8122	0.8757	0.8796	0.8763	0.8058 ± 0.1265
	CPU time	1.5594	0.0725	0.0948	0.2414	0.5129	0.9627	0.3769 ± 0.3716
Titanic	Dimension	4	5	5	5	5	4	4.8000 ± 0.4472
	Accuracy	0.7411	0.7640	0.7660	0.7660	0.7552	0.7411	0.7585 ± 0.0107
	CPU time	0.0531	0.0047	0.0053	0.0089	0.0172	0.0303	0.0133 ± 0.0107
Diabetic	Dimension	335	38	76	149	219	283	153.0000 ± 100.5311
	Accuracy	0.6667	0.6733	0.6733	0.6700	0.6700	0.6700	0.6713 ± 0.0018
	CPU time	1.9266	0.1334	0.1361	0.2954	0.6301	1.1477	0.4685 ± 0.4302
Splice	Dimension	778	80	160	320	480	640	336.0000 ± 229.0851
	Accuracy	0.5499	0.5237	0.5260	0.5389	0.5448	0.5517	0.5370 ± 0.0120
	CPU time	5.4807	1.7209	1.7443	1.8110	1.8252	1.9040	1.8011 ± 0.0723
Waveform	Dimension	320	32	64	128	192	256	134.4000 ± 91.6341
	Accuracy	0.6707	0.6754	0.6707	0.6707	0.6707	0.6707	0.6716 ± 0.0021
	CPU time	0.4734	0.2271	0.3098	0.3504	0.3004	0.5426	0.3461 ± 0.1185
German	Dimension	556	56	112	224	336	448	235.2000 ± 160.3596
	Accuracy	0.7233	0.7233	0.7233	0.7233	0.7233	0.7233	0.7233 ± 0
	CPU time	4.7145	1.2193	1.6720	0.8996	1.5495	2.7169	1.6115 ± 0.6874
Heart	Dimension	134	14	28	55	82	109	57.6000 ± 38.7982
	Accuracy	0.5900	0.5600	0.5600	0.5600	0.5600	0.5600	0.5600 ± 0
	CPU time	0.0661	0.0217	0.0068	0.0136	0.0258	0.0440	0.0224 ± 0.0141

Table 6 cotinued

Dataset	Percent(%)	PCA	SPCA				
		100	10	20	40	60	80	Mean ± std
Breast-cancer	Dimension	133	16	32	64	94	119	65.0000 ± 42.5676
	Accuracy	0.7013	0.7013	0.7013	0.7013	0.7013	0.7013	0.7013 ± 0
	CPU time	0.1177	0.0095	0.0109	0.0224	0.0387	0.0773	0.0318 ± 0.0280
Thyroid	Dimension	45	11	21	35	43	46	31.2000 ± 14.8728
	Accuracy	0.9467	0.6800	0.6933	0.7867	0.9467	0.9333	0.8080 ± 0.1274
	CPU time	0.0741	0.0114	0.0044	0.0090	0.0201	0.0370	0.0164 ± 0.0129

lower than that of the KPCA. In these cases, the used data are not exceeded to 40% of the whole data, and satisfactory results can be achieved. Therefore, taking one with another, the proposed SKPCA is effective and practical.

6 Conclusions

The contribution of this paper is that we learn linear PCA with linear programming, and extend this method to kernel PCA. Comparing with solving a semi-definite programming problem, the proposed method could learn linear and nonlinear PCA easily, and guarantee obtaining the unique global solution. Moreover, the proposed method can use partial data points to determine all the principal components, and so the store memory and time consumption can be saved consumedly.

To validate the performance of the proposed method, three tests are carried out. The results of Test 1 show that the first principal component lines determined by the standard PCA and the proposed SPCA are close to each other very well, but the latter only use a few data points. Test 2 and 3 illustrate that the proposed SPCA and SKPCA can compete with or outperform the standard PCA and KPCA in generalization ability, but they only need much less memory and time consuming. Therefore, the proposed methods are very efficient and practical. As for how to select partial data points and how much data points play important roles in determining the principal components will be reported in the further studies.

Acknowledgements The work described in this paper was partially supported by the National Natural Science Foundation of China (Nos. 60673095, 60975035, 71031006), Doctoral Fund of Ministry of Education of China (No. 20091401110003), Doctoral Fund of Shandong University of Technology (No. 4041-410002), Key Project of Science Technology Research of Ministry of Education (No. 208021). Program for New Century Excellent Talents in University (NCET-07-0525), Program for the Top Young Academic Leaders of Higher Learning Institutions (TYAL), Key Project of Natural Science Foundation of Shanxi Province (No. 2009011017-2), Foundation for Returness of Shanxi Province (No. 2008-14).

References

1. Barzilay O, Brailovsky V (1999) On domain knowledge and feature selection using a support vector machine. Pattern Recognition Lett 20:475–484
2. Brown M, Grundy W, Lin D, Cristianini N, Sugnet C, Furey T, Ares M, Haussler D (2000) Knowledge-based analysis of microarray gene expression data by using support vector machines. Proc Nat Acad Sci USA 97:262–267
3. Croux C, Haesbroeck G (2000) Principal component analysis based on robust estimators of the covariance or correlation matrix: influence functions and efficiencies. Biometrika 87:603–618
4. Diamantaras KI, Kung SY (1996) Principal component neural networks. Wiley, New York
5. Drucker H, Wu D, Vapnik V (1999) Support vector machines for spam categorization. IEEE Trans Neural Networks 10:1048–1054
6. Higuchi I, Eguchi S (2004) Robust principal component analysis with adaptive selection for tuning parameters. J Mach Learn Res 5:453–471
7. Joachims T (1998) Text categorization with support vector machines: learning with many relevant features. In: Proceedings of the European Conference on Machine Learning. Springer, Berlin, pp 137–142
8. Jolliffe IT (1986) Principal component analysis. Springer, New York
9. Shawe-Taylor J, Cristianini Nello (2005) Kernel methods for pattern analysis. China Machine press, Beijing
10. Schölkopf B, Smolar AJ, Muller KR (1998) Nonlinear component analysis as a kernel eigenvalue problem. Neural Comput 10:1299–1319
11. Suykens JAK, Vandewalle J (1999) Least squares support vector machine classifiers. Neural process Lett 9(3):293–300

12. Suykens JAK, Van Gestel T, Vandewalle J, De Moor B (2003) A support vector machine formulation to PCA analysis and its kernel version. IEEE Trans Neural Networks 14(2):447–450
13. Tao Q, Wu G, Wang J (2005) A new maximum margin algorithm for one–class problems and its boosting implementation. Pattern Recognition 38:1071–1077
14. Tao Q, Wu G, Wang J (2007) Learning linear PCA with convex semi-definite programming. Pattern Recognition 40(10):2633–2640
15. Tax D, Duin R (2004) Support vector data description. Mach Learn 54:45–66
16. Vapnik V (1995) The nature of statistical learning theory. Springer, New York
17. Vapnik V (1998) Statistical learning theory. Wiley, New York