Available online at www.sciencedirect.com
ScienceDirect
Journal of Multivariate Analysis 99 (2008) 1015–1034
Journal of Multivariate Analysis
www.elsevier.com/locate/jmva

Sparse principal component analysis via regularized low rank matrix approximation

Haipeng Shen[a,*], Jianhua Z. Huang[b]

[a] *Department of Statistics and Operations Research, University of North Carolina at Chapel Hill, Chapel Hill, NC 27599, USA*

[b] *Department of Statistics, Texas A&M University, College Station, TX 77843, USA*

Received 25 July 2006
Available online 27 June 2007

Abstract

Principal component analysis (PCA) is a widely used tool for data analysis and dimension reduction in applications throughout science and engineering. However, the principal components (PCs) can sometimes be difficult to interpret, because they are linear combinations of all the original variables. To facilitate interpretation, sparse PCA produces modified PCs with sparse loadings, i.e. loadings with very few non-zero elements. In this paper, we propose a new sparse PCA method, namely *sparse PCA via regularized SVD* (sPCA-rSVD). We use the connection of PCA with singular value decomposition (SVD) of the data matrix and extract the PCs through solving a low rank matrix approximation problem. Regularization penalties are introduced to the corresponding minimization problem to promote sparsity in PC loadings. An efficient iterative algorithm is proposed for computation. Two tuning parameter selection methods are discussed. Some theoretical results are established to justify the use of sPCA-rSVD when only the data covariance matrix is available. In addition, we give a modified definition of variance explained by the sparse PCs. The sPCA-rSVD provides a uniform treatment of both classical multivariate data and high-dimension-low-sample-size (HDLSS) data. Further understanding of sPCA-rSVD and some existing alternatives is gained through simulation studies and real data examples, which suggests that sPCA-rSVD provides competitive results.

AMS 1991 subject classification: 62H20; 62H25; 62H30

Keywords: Dimension reduction; High-dimension-low-sample-size; Regularization; Singular value decomposition; Thresholding

* Corresponding author.
E-mail addresses: haipeng@email.unc.edu (H. Shen), jianhua@stat.tamu.edu (J.Z. Huang).

0047-259X/$ - see front matter © 2007 Elsevier Inc. All rights reserved.
doi:10.1016/j.jmva.2007.06.007

1. Introduction

Principal component analysis (PCA) has been widely used in many applications as a feature extraction and dimension reduction tool as well illustrated in Jolliffe [11]. Suppose $\mathbf{X}$ is an $n \times p$ data matrix with $\text{rank}(\mathbf{X}) = r$, which records p variables on n observations. PCA sequentially finds unit vectors $\mathbf{v}_1, \ldots, \mathbf{v}_r$ that maximize the variance of $\mathbf{Xv}$ under the constraint that $\mathbf{v}_{i+1}$ is orthogonal to $\mathbf{v}_1, \ldots, \mathbf{v}_i$. Such vectors are called principal component (PC) loadings, and the $\mathbf{Xv}_i$'s are the corresponding PCs. The idea of dimension reduction using PCA is that the first *few* PCs might retain most of the variation in the data. Interpretation of PCs is useful, especially when the variables have physical meanings, for example, in microarray data each variable corresponds to a specific gene. However, PCs are usually linear combinations of *all* the original variables and their loadings are typically non-zero. This often makes it difficult to interpret the PCs without using subjective judgment, especially when p is large as frequently encountered in modern statistical applications.

To ease this drawback of PCA, various proposals have been introduced in the literature. Jolliffe [10] described several rotation techniques that are helpful for interpreting PCs. Jolliffe and Uddin [13] proposed SCoT to successively find linear combinations that maximize a criterion which balances variance and some simplicity measure. Vines [17] considered *simple components*, whose loadings are restricted to only integers such as 0, 1 and -1. Another group of methods, referred to as sparsePCA methods, aims at finding loading vectors with many zero components, thus increasing interpretability of PCs by reducing the number of explicitly used variables. Cadima and Jolliffe [2] described a *simple thresholding* approach, which artificially sets regular PC loadings to zero if their absolute values are below a certain threshold. Jolliffe et al. [12] proposed SCoTLASS, which applies the *lasso* penalty [16] on the loadings in a PCA optimization problem. More recently, Zou et al. [20] reformulated PCA as a regression-type problem, and proposed SPCA which achieves sparseness by imposing the lasso penalty on the regression coefficients.

In this paper, we provide a new approach to achieve sparse PCA, making use of the close connection between PCA and singular value decomposition (SVD) that PCA can be computed via the SVD of the data matrix $\mathbf{X}$. Without loss of generality, assume the columns of $\mathbf{X}$ are centered. Suppose $\text{rank}(\mathbf{X}) = r$ and let the SVD of $\mathbf{X}$ be

$$\mathbf{X} = \mathbf{UDV}^T,$$

where $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_r]$, $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_r]$ and $\mathbf{D} = \text{diag}\{d_1, \ldots, d_r\}$. The columns of $\mathbf{U}$ are orthonormal, so are the columns of $\mathbf{V}$. The singular values are assumed to be ordered so that $d_1 \geqslant d_2 \geqslant \cdots \geqslant d_r > 0$. Then, the columns of $\mathbf{Z} = \mathbf{UD}$ are the PCs, and the columns of $\mathbf{V}$ are the corresponding loadings.

To motivate our approach, we need to look at SVD from the viewpoint of low rank approximation of matrices. For an integer $l \leqslant r$, define

$$\mathbf{X}^{(l)} \equiv \sum_{k=1}^{l} d_k \mathbf{u}_k \mathbf{v}_k^T.$$

Then, $\mathbf{X}^{(l)}$ is the closest rank-l matrix approximation to $\mathbf{X}$ [4]. Here the term "closest" simply means that $\mathbf{X}^{(l)}$ minimizes the squared Frobenius norm between $\mathbf{X}$ and an arbitrary rank-l matrix

$\mathbf{X}^*$, where the Frobenius norm is defined as

$$\|\mathbf{X} - \mathbf{X}^*\|_F^2 = \mathrm{tr}\{(\mathbf{X} - \mathbf{X}^*)(\mathbf{X} - \mathbf{X}^*)^T\}.$$

Suppose, for example, we seek the best rank-one matrix approximation of $\mathbf{X}$ under the Frobenius norm. Note that any $n \times p$ rank-one matrix can be written as $\widetilde{\mathbf{u}}\widetilde{\mathbf{v}}^T$, where $\widetilde{\mathbf{u}}$ is a norm-1 n-vector and $\widetilde{\mathbf{v}}$ is a p-vector. The problem can be formulated as the following optimization problem:

$$\min_{\widetilde{\mathbf{u}},\widetilde{\mathbf{v}}} \|\mathbf{X} - \widetilde{\mathbf{u}}\widetilde{\mathbf{v}}^T\|_F^2. \tag{1}$$

Then the low rank approximation property of SVD implies that the solution is

$$\widetilde{\mathbf{u}} = \mathbf{u}_1, \quad \widetilde{\mathbf{v}} = d_1\mathbf{v}_1.$$

The subsequent pairs $(\mathbf{u}_k, d_k\mathbf{v}_k)$, $k > 1$, provide best rank one approximations of the corresponding residual matrices. For example, $d_2\mathbf{u}_2\mathbf{v}_2^T$ is the best rank one approximation of $\mathbf{X} - d_1\mathbf{u}_1\mathbf{v}_1^T$.

To obtain sparse loadings, we impose regularization penalties on $\widetilde{\mathbf{v}}$ in the optimization problem (1), and refer to our approach as *sparse PCA via regularized SVD*, or sPCA-rSVD for short. The key of our proposal is the observation that the optimization problem (1) is connected to least squares regressions. For a fixed $\widetilde{\mathbf{u}}$, the optimal $\widetilde{\mathbf{v}}$ is the least squares coefficient vector of regressing the columns of $\mathbf{X}$ on $\widetilde{\mathbf{u}}$. Introducing sparsity on $\widetilde{\mathbf{v}}$ in such a context is a familiar variable selection problem in regression. Thus many existing variable selection techniques using regularization penalties [16,3,5] are readily applicable.

The benefit of imposing sparsity-inducing penalties on $\widetilde{\mathbf{v}}$ in the optimization (1) is two-fold. First, the resulting PC loading vector is made sparse so that those *negligible* variables will not appear in the corresponding PC, therefore the PCs obtained are more interpretable. Meanwhile, since the left-out variables are *negligible*, the sparse PC won't suffer much in terms of the variance it explains. Second, when a covariance matrix has sparse eigenvectors, by using the sparsity-inducing penalties, sPCA-rSVD is statistically more efficient than the standard PCA in extracting the PCs, as illustrated using simulated examples in Sections 3.2 and 3.3. This is similar to regression problems with irrelevant regressors, where variable selection improves statistical efficiency.

We propose an iterative algorithm for computation of sPCA-rSVD. The algorithm only involves simple linear regression and componentwise thresholding rules; hence it enjoys nice properties such as easy implementation and efficient computation. We define the *degree of sparsity* of a PC as the number of zero components in the corresponding loading vector. The degree of sparsity naturally serves as the tuning parameter of the method. Two approaches are proposed to select the "optimal" degree of sparsity, one cross validation approach, and one ad hoc approach that is useful when the sample size is small or only the data covariance matrix is available. Different degree of sparsity is allowed for different loading vectors in our framework.

In addition to proposing sPCA-rSVD, we give a new definition of variance explained by the sparse PCs. This is necessary since for sparse PCA, the loading vectors need not be orthogonal and the PCs need not be uncorrelated, which makes the conventional definition too optimistic. We fix the problem of the conventional definition by using the viewpoint of dimension reduction. Our definition of the variance explained by the sparse PCs can be used to select the tuning parameters specifying sparsity, or to select the number of important PCs.

In this paper, we also prove that our sPCA-rSVD procedure still applies when only the covariance matrix is available, because it depends on the data through the Gram matrix $\mathbf{X}^T\mathbf{X}$. The sPCA-rSVD handles both "long" data matrices where $n \geqslant p$ and "fat" matrices where $n < p$ or even $n \ll p$ in a unified way. Standard PCA in classical multivariate analysis usually deals with

the "long" data matrices. The "fat" matrices are *high-dimension-low-sample-size* (HDLSS) data objects [15]. HDLSS has rapidly become a common feature of data encountered in many diverse fields such as text categorization, medical imaging and microarray gene expression analysis, and is outside the domain of classical multivariate analysis.

The rest of the paper is organized as follows. We present the methodological details of our sPCA-rSVD procedure in Section 2. Section 2.1 gives the optimization criterion that leads to sparse PCA; Section 2.2 describes an efficient iterative algorithm for computation; Section 2.3 proposes a modified definition of the variance explained by the sparse PCs; Section 2.4 discusses extraction of sparse PC loadings when only the covariance matrix of the data is available; Section 2.5 provides methods for tuning parameter selection. Further understanding of the proposed method, mainly through simulation studies, is provided in Section 3. Sections 4 and 5 compare the proposed procedure with SPCA and simple thresholding, respectively. Section 6 illustrates the proposed method using some real data examples. We end the paper with some discussion in Section 7 and technical proofs in Appendix A.

2. Sparse PCA via regularized SVD

This section describes in detail our sPCA-rSVD procedure. We focus our presentation on the procedure extracting the first sparse PC loading vector. Subsequent loading vectors can be obtained by applying the same procedure to the residual matrices of the sequential matrix approximations.

2.1. A penalized sum-of-squares criterion

Suppose $\mathbf{u}\mathbf{v}^T$ with $\|\mathbf{v}\| = 1$ is the best rank-one approximation of the data matrix $\mathbf{X}$. Then $\mathbf{u}$ is the first PC and $\mathbf{v}$ is the corresponding loading vector. For a given $\mathbf{u}$, elements of $\mathbf{v}$ are the regression coefficients by regressing the columns of $\mathbf{X}$ on $\mathbf{u}$. To achieve sparseness on $\mathbf{v}$, we propose to employ some regularization penalties in these regressions that promote shrinkage and sparsity on the regression coefficients.

However, the loading vector $\mathbf{v}$ is typically constrained to have unit length to make the representation unique. This constraint makes direct application of a penalty on $\mathbf{v}$ inappropriate. To overcome this difficulty, we rewrite $\mathbf{u}\mathbf{v}^T = \widetilde{\mathbf{u}}\widetilde{\mathbf{v}}^T$, where $\widetilde{\mathbf{u}}$ and $\widetilde{\mathbf{v}}$ are re-scaled versions of $\mathbf{u}$ and $\mathbf{v}$ such that $\widetilde{\mathbf{u}}$ has unit length and $\widetilde{\mathbf{v}}$ is free of any scale constraint, and then perform shrinkage on $\widetilde{\mathbf{v}}$ through some regularization penalty. After a sparse $\widetilde{\mathbf{v}}$ is obtained, we define the corresponding sparse loading vector as $\mathbf{v} = \widetilde{\mathbf{v}}/\|\widetilde{\mathbf{v}}\|$.

The precise formulation of our idea is the following. For a given $n \times p$ data matrix $\mathbf{X}$, we find an n-vector $\widetilde{\mathbf{u}}$ with $\|\widetilde{\mathbf{u}}\| = 1$ and a p-vector $\widetilde{\mathbf{v}}$ that minimize the following penalized sum-of-squares criterion,

$$\|\mathbf{X} - \widetilde{\mathbf{u}}\widetilde{\mathbf{v}}^T\|_F^2 + P_\lambda(\widetilde{\mathbf{v}}), \tag{2}$$

where $\|\mathbf{X} - \widetilde{\mathbf{u}}\widetilde{\mathbf{v}}^T\|_F^2 = \mathrm{tr}\{(\mathbf{X} - \widetilde{\mathbf{u}}\widetilde{\mathbf{v}}^T)(\mathbf{X} - \widetilde{\mathbf{u}}\widetilde{\mathbf{v}}^T)^T\} = \sum_{i=1}^n \sum_{j=1}^p (x_{ij} - \tilde{u}_i\tilde{v}_j)^2$ is the squared Frobenius norm, $P_\lambda(\widetilde{\mathbf{v}}) = \sum_{j=1}^p p_\lambda(|\tilde{v}_j|)$ is a penalty function and $\lambda \geqslant 0$ is a tuning parameter. Denote the solution of the optimization problem as $\mathbf{u}^*$ and $\mathbf{v}^*$. Then the unit length PC loading vector is $\mathbf{v} = \mathbf{v}^*/\|\mathbf{v}^*\|$.

Here, for simplicity, the penalty function remains the same for different components of $\widetilde{\mathbf{v}}$. It is a straightforward extension to allow different components of $\widetilde{\mathbf{v}}$ to use different penalty functions. We shall consider the soft thresholding (or L_1 or *lasso*) penalty [16], the hard thresholding penalty [3],

and the smoothly clipped absolute deviation (SCAD) penalty [5]. Selection of the tuning parameter λ is an important question in practice, and will be addressed later in Section 2.5.

2.2. An iterative algorithm

This subsection provides an iterative algorithm to minimize (2) with respect to $\widetilde{\mathbf{u}}$ and $\widetilde{\mathbf{v}}$ under the constraint $\|\widetilde{\mathbf{u}}\| = 1$. First consider the problem of optimizing over $\widetilde{\mathbf{u}}$ for a fixed $\widetilde{\mathbf{v}}$. The minimizing $\widetilde{\mathbf{u}}$ can be obtained according to Lemma A.1.

Lemma 1. *For a fixed* $\widetilde{\mathbf{v}}$, *the* $\widetilde{\mathbf{u}}$ *that minimizes* (2) *and satisfies* $\|\widetilde{\mathbf{u}}\| = 1$ *is* $\widetilde{\mathbf{u}} = \mathbf{X}\widetilde{\mathbf{v}}/\|\mathbf{X}\widetilde{\mathbf{v}}\|$.

Next we discuss optimization over $\widetilde{\mathbf{v}}$ for a fixed $\widetilde{\mathbf{u}}$. Since $P_\lambda(\widetilde{\mathbf{v}}) = \sum_j p_\lambda(|\tilde{v}_j|)$, the minimization criterion (2) can be rewritten as

$$\sum_i \sum_j (x_{ij} - \tilde{u}_i \tilde{v}_j)^2 + \sum_j p_\lambda(|\tilde{v}_j|) = \sum_j \left\{ \sum_i (x_{ij} - \tilde{u}_i \tilde{v}_j)^2 + p_\lambda(|\tilde{v}_j|) \right\}. \tag{3}$$

Therefore, we can optimize over individual components of $\widetilde{\mathbf{v}}$ separately. Expanding the squares and observing that $\sum_i \tilde{u}_i^2 = 1$, we obtain

$$\sum_i (x_{ij} - \tilde{u}_i \tilde{v}_j)^2 = \sum_i x_{ij}^2 - 2\sum_i x_{ij}\tilde{u}_i\tilde{v}_j + \sum_i \tilde{u}_i^2\tilde{v}_j^2 = \sum_i x_{ij}^2 - 2(\mathbf{X}^T\widetilde{\mathbf{u}})_j\tilde{v}_j + \tilde{v}_j^2.$$

Hence, the optimal $\tilde{v}_j$ minimizes $\tilde{v}_j^2 - 2(\mathbf{X}^T\widetilde{\mathbf{u}})_j\tilde{v}_j + p_\lambda(|\tilde{v}_j|)$ and depends on the form of $p_\lambda(\cdot)$. For the three penalties mentioned in Section 2.1, the expression of the optimal $\tilde{v}_j$ can be obtained by repeatedly applying Lemma 2. The proof of the lemma is easy and thus omitted.

Lemma 2. *Let* $\hat{\beta}$ *be the minimizer of* $\beta^2 - 2y\beta + p_\lambda(|\beta|)$.

1. *For the soft thresholding penalty* $p_\lambda(|\theta|) = 2\lambda|\theta|$,

$$\hat{\beta} = h_\lambda^{\text{soft}}(y) = \text{sign}(y)(|y| - \lambda)_+;$$

2. *For the hard thresholding penalty* $p_\lambda(|\theta|) = \lambda^2 I(|\theta| \neq 0)$,

$$\hat{\beta} = h_\lambda^{\text{hard}}(y) = I(|y| > \lambda)y;$$

3. *For the SCAD penalty*

$$p_\lambda(|\theta|) = 2\lambda|\theta| I(|\theta| \leqslant \lambda) - \frac{\theta^2 - 2a\lambda|\theta| + \lambda^2}{(a-1)} I(\lambda < |\theta| \leqslant a\lambda) + (a+1)\lambda^2 I(|\theta| > a\lambda),$$

$$\hat{\beta} = h_\lambda^{\text{SCAD}}(y) = \begin{cases} \text{sign}(y)(|y| - \lambda)_+ & \text{for } |y| \leqslant 2\lambda; \\ \{(a-1)y - \text{sign}(y)a\lambda\}/(a-2) & \text{for } 2\lambda < |y| \leqslant a\lambda; \\ y & \text{for } |y| > a\lambda, \end{cases}$$

where $a > 2$ *is another tuning parameter. We fix* $a = 3.7$ *following the recommendation in Fan and Li [5].*

According to Lemma 2, the minimizer of (3) is obtained by applying a thresholding rule h_λ to the vector $\mathbf{X}^T\widetilde{\mathbf{u}}$ componentwise. We shall denote $\widetilde{\mathbf{v}} = h_\lambda(\mathbf{X}^T\widetilde{\mathbf{u}})$ with the understanding that the rule $h_\lambda(\cdot)$ is applied componentwise.

The above discussion leads to an iterative procedure for minimizing (2).

Algorithm 1. sPCA-rSVD Algorithm

1. Initialize: Apply the standard SVD to $\mathbf{X}$ and obtain the best rank-one approximation of $\mathbf{X}$ as $s\mathbf{u}^*\mathbf{v}^{*T}$ where $\mathbf{u}^*$ and $\mathbf{v}^*$ are unit vectors. Set $\widetilde{\mathbf{v}}_{\text{old}} = s\mathbf{v}^*$ and $\widetilde{\mathbf{u}}_{\text{old}} = \mathbf{u}^*$.
2. Update:
   (a) $\widetilde{\mathbf{v}}_{\text{new}} = h_\lambda(\mathbf{X}^T\widetilde{\mathbf{u}}_{\text{old}})$;
   (b) $\widetilde{\mathbf{u}}_{\text{new}} = \mathbf{X}\widetilde{\mathbf{v}}_{\text{new}}/\|\mathbf{X}\widetilde{\mathbf{v}}_{\text{new}}\|$.
3. Repeat Step 2 replacing $\widetilde{\mathbf{u}}_{\text{old}}$ and $\widetilde{\mathbf{v}}_{\text{old}}$ by $\widetilde{\mathbf{u}}_{\text{new}}$ and $\widetilde{\mathbf{v}}_{\text{new}}$ until convergence.
4. Standardize the final $\widetilde{\mathbf{v}}_{\text{new}}$ as $\mathbf{v} = \widetilde{\mathbf{v}}_{\text{new}}/\|\widetilde{\mathbf{v}}_{\text{new}}\|$, the desired sparse loading vector.

Setting $\lambda = 0$ in the above algorithm, Step 2a reduces to $\widetilde{\mathbf{v}}_{\text{new}} = \mathbf{X}^T\widetilde{\mathbf{u}}_{\text{old}}$ and the algorithm becomes the well-known alternating least-squares algorithm for calculating SVD [7]. Penalty functions other than the three discussed above can also be used under the current framework, where we only need to modify the thresholding rule in Step 2a of Algorithm 1 accordingly. The computation cost of each iteration of our algorithm is $O(np)$.

The algorithm is developed for fixed λ. We could introduce tuning parameter selection in Step 2a using, for example, the cross validation criterion in Section 2.5. However, we prefer to use the degree of sparsity of the loading vector as the tuning parameter for two reasons. First, the interpretation is easy. More importantly, the parameter selection can then be performed outside the iteration loop, which is a major computational advantage. Note that, in Step 2a, setting the degree of sparsity to be j $(1 \leqslant j \leqslant p-1)$ is equivalent to setting $\lambda \in \left[|\mathbf{X}^T\widetilde{\mathbf{u}}_{\text{old}}|_{(j)}, |\mathbf{X}^T\widetilde{\mathbf{u}}_{\text{old}}|_{(j+1)}\right)$, where $|\mathbf{X}^T\widetilde{\mathbf{u}}_{\text{old}}|_{(j)}$ is the jth order statistic of $|\mathbf{X}^T\widetilde{\mathbf{u}}_{\text{old}}|$.

The iterative procedure of the sPCA-rSVD algorithm is defined for one-dimensional vectors $\widetilde{\mathbf{u}}$ and $\widetilde{\mathbf{v}}$, and can be used to obtain the first sparse loading vector $\mathbf{v}_1$. Subsequent sparse loading vectors $\mathbf{v}_i$ $(i > 1)$ can be obtained sequentially via rank-one approximation of residual matrices. Our framework allows different degree of sparsity for different $\mathbf{v}_i$'s by using different tuning parameters. However, the orthogonality among the $\mathbf{v}_i$'s is lost, a nice property enjoyed by standard PCA. Several other sparse PCA procedures lose this property as well, which is the price one pays for easy interpretation of the results.

We want to comment that it is possible to extract the first k PCs together using the best rank-k approximation formulation of the penalized least squares criterion. Tuning parameter selection would be much more involved, however. We leave this extension for future research.

2.3. Adjusted variance explained by PCs

In standard PCA, the PCs are uncorrelated and their loadings are orthogonal. These properties are lost in sparse PCA [10,13,12]. In this subsection we provide a modified definition of the variance explained by the PCs in response to the loss of these properties. An earlier proposal of the adjusted variance by Zou et al. [20] successfully takes into account the possible correlation between the sparse PCs, but the lack of orthogonality of the loadings is not addressed.

Let $\mathbf{V}_k = [\mathbf{v}_1, \ldots, \mathbf{v}_k]$ be the matrix of the first k sparse loading vectors. As in standard PCA, define the ith PC as $\mathbf{u}_i = \mathbf{X}\mathbf{v}_i$ and the variance it accounts for is then defined as $\|\mathbf{u}_i\|^2$. Denote the matrix of the first k PCs as $\mathbf{U}_k = [\mathbf{u}_1, \ldots, \mathbf{u}_k]$. Standard PCA calculates the total variance explained by the first k PCs as $\text{tr}\left(\mathbf{U}_k^T\mathbf{U}_k\right) = \sum_i \|\mathbf{u}_i\|^2$. Application of these concepts in

sparse PCA has two problems. First, the $\mathbf{v}_i$'s may not be orthogonal, which means that information contents of these $\mathbf{v}_i$'s may overlap. Second, calculation of the total variance as the sum of individual variance is too generous if the PCs are correlated. Below we offer a new definition of variance explained by the PCs from the viewpoint of dimension reduction.

The ith PC $\mathbf{u}_i$ is the projection of the data matrix onto the ith loading vector $\mathbf{v}_i$. When the loading vectors are not orthogonal, we should not consider separate projection of the data matrix onto each of the first k loading vectors. Instead, we consider the projection of $\mathbf{X}$ onto the k-dimensional subspace spanned by the k loading vectors as $\mathbf{X}_k = \mathbf{X}\mathbf{V}_k\left(\mathbf{V}_k^T\mathbf{V}_k\right)^{-1}\mathbf{V}_k^T$. We generally define the *total variance explained* by the first k PCs as $\mathrm{tr}\left(\mathbf{X}_k^T\mathbf{X}_k\right)$, which can be easily calculated using the SVD of $\mathbf{X}_k$. If the loading vectors are orthogonal as in the standard PCA, the above definitions reduce to the conventional definitions: in particular, $\mathbf{X}_k = \mathbf{U}_k\mathbf{V}_k^T$, and the total variance explained simplifies to $\mathrm{tr}\left(\mathbf{U}_k^T\mathbf{U}_k\right)$.

The above discussion suggests that the PC loading vectors might be better named *PC basis* vectors. These basis vectors span a sequence of nested subspaces that the data matrix can be projected onto. The following theorem suggests that the newly defined variance increases as additional basis vectors are added, and is bounded above by the total variance in the data matrix $\mathbf{X}$, which is calculated as $\mathrm{tr}\left(\mathbf{X}^T\mathbf{X}\right)$.

Theorem 1. $\mathrm{tr}\left(\mathbf{X}_k^T\mathbf{X}_k\right) \leqslant \mathrm{tr}\left(\mathbf{X}_{k+1}^T\mathbf{X}_{k+1}\right) \leqslant \mathrm{tr}\left(\mathbf{X}^T\mathbf{X}\right)$.

To deal with the correlation among PCs, in calculating the added variance explained by an additional PC, the variance accountable by the previous PCs should be adjusted for. Define the *adjusted variance* of the kth PC as $\mathrm{tr}\left(\mathbf{X}_k^T\mathbf{X}_k\right) - \mathrm{tr}\left(\mathbf{X}_{k-1}^T\mathbf{X}_{k-1}\right)$. According to Theorem A.1, the adjusted variance is always non-negative. We also define the *cumulative percentage of explained variance* (CPEV) by the first k PCs as $\mathrm{tr}\left(\mathbf{X}_k^T\mathbf{X}_k\right)/\mathrm{tr}\left(\mathbf{X}^T\mathbf{X}\right)$. It is valued between 0 and 1 as a consequence of Theorem A.1. Below in Section 2.5.2, the CPEV is used in an ad hoc procedure for selecting the degree of PC sparsity. The CPEV can also be used in a screeplot to determine the number of important PCs.

2.4. Sparse PCs and the sample covariance matrix

This subsection discusses computation of sparse PCA when only the sample covariance matrix is available. We show that our sparse loading vectors depend on the data matrix $\mathbf{X}$ only through the Gram matrix $\mathbf{X}^T\mathbf{X}$, and so is the total variance explained by the first k PCs. Since the Gram matrix is the sample covariance matrix up to a scaling constant, an immediate conclusion of this subsection is that our sparse PCA is well-defined using only the sample covariance matrix.

Lemma 3. *Suppose* $\widetilde{\mathbf{u}}_1$ *and* $\widetilde{\mathbf{v}}_1$ *minimizes* (2) *with* $\|\widetilde{\mathbf{u}}\| = 1$. *Then* $\widetilde{\mathbf{v}}_1$ *minimizes*

$$-2\|\mathbf{X}\widetilde{\mathbf{v}}\| + \|\widetilde{\mathbf{v}}\|^2 + P_\lambda(\widetilde{\mathbf{v}}) \tag{4}$$

and $\widetilde{\mathbf{u}}_1 = \mathbf{X}\widetilde{\mathbf{v}}_1/\|\mathbf{X}\widetilde{\mathbf{v}}_1\|$.

Lemma 3 suggests that $\widetilde{\mathbf{v}}_1$ depends on the data matrix $\mathbf{X}$ only through $\mathbf{X}^T\mathbf{X}$, or the corresponding sample covariance matrix; hence the first sparse loading vector $\mathbf{v}_1 = \widetilde{\mathbf{v}}_1/\|\widetilde{\mathbf{v}}_1\|$ has the same property. Theorem 2 shows that the same conclusion holds for the first k loading vectors.

Theorem 2. *The first k sparse loading vectors* $\mathbf{v}_1, \ldots, \mathbf{v}_k$ *obtained using the sPCA-rSVD procedure depend on* $\mathbf{X}$ *only through* $\mathbf{X}^T\mathbf{X}$.

The next theorem shows that the variance explained by the first k PCs also depends on $\mathbf{X}$ only through $\mathbf{X}^T\mathbf{X}$.

Theorem 3. *The variance* $\operatorname{tr}\left(\mathbf{X}_k^T\mathbf{X}_k\right)$ *depends on* $\mathbf{X}$ *only through* $\mathbf{X}^T\mathbf{X}$.

Theorems 2 and 3 suggest that the sparse loading vectors and the adjusted variance explained depend on the data matrix only through its sample covariance matrix. This implies that our procedure still applies in scenarios where only the sample covariance matrix is available, as is the case in the pitprops application (Section 6.1). Suppose $\mathbf{S}$ is the sample covariance matrix, one can arbitrarily choose a *pseudo-data* matrix $\mathbf{X}$ such that $\mathbf{X}^T\mathbf{X} = n\mathbf{S}$. A natural candidate of $\mathbf{X}$ is the square-root matrix of $n\mathbf{S}$, which can be obtained via an eigen decomposition of $n\mathbf{S}$. Another choice of $\mathbf{X}$ is the triangular matrix from the Cholesky decomposition of $n\mathbf{S}$.

2.5. Tuning parameter selection

2.5.1. K-fold cross validation (CV)

In the following discussion, we use the degree of sparsity as the tuning parameter.

Algorithm 2. K-fold CV Tuning Parameter Selection

1. Randomly group the rows of $\mathbf{X}$ into K roughly equal-sized groups, denoted as $\mathbf{X}^1, \ldots, \mathbf{X}^K$;
2. For each $j \in \{0, 1, 2, \ldots, p-1, p\}$, do the following:
   (a) For $k = 1, \ldots, K$, let $\mathbf{X}^{-k}$ be the data matrix $\mathbf{X}$ leaving out $\mathbf{X}^k$. Apply Algorithm 1 on $\mathbf{X}^{-k}$ to derive the loading vector $\mathbf{v}^{-k}(j)$. Then project $\mathbf{X}^k$ onto $\mathbf{v}^{-k}(j)$ to obtain the projection coefficients as $\mathbf{u}^k(j) = \mathbf{X}^k\mathbf{v}^{-k}(j)$;
   (b) Calculate the K-fold CV score defined as

$$\mathrm{CV}(j) = \sum_{k=1}^{K} \frac{\sum_{i=1}^{n_k}\sum_{l=1}^{p}\{x_{il}^k - u_i^k(j)v_l^{-k}(j)\}^2}{n_k p}, \tag{5}$$

   where n_k is the number of rows of $\mathbf{X}^k$, and u_i^k and v_l^{-k} are, respectively, the ith and lth elements of $\mathbf{u}^k$ and $\mathbf{v}^{-k}$;
3. Select the degree of sparsity as $j_0 = \operatorname{argmin}_j\{\mathrm{CV}(j)\}$.

In practice, K is usually chosen to be 5 or 10 for computational efficiency. The case where $K = n$ is known as *leave-one-out* CV, which can be computationally expensive for moderate to large data. We use $K = 5$ in our simulation studies.

2.5.2. An ad hoc approach

The CV approach is not suitable when only the sample covariance matrix is available. We propose here an ad hoc approach to select the tuning parameter as an alternative.

The degrees of sparsity are sequentially selected for the first k sparse loading vectors. Suppose the tuning parameters for the first $k-1$ loading vectors are selected and the corresponding loadings

are $\mathbf{v}_1, \ldots, \mathbf{v}_{k-1}$. The proposed ad hoc procedure for selecting the tuning parameter for the kth sparse loading vector is as follows:

Algorithm 3. Ad Hoc Tuning Parameter Selection
1. For each $j \in \{0, 1, 2, \ldots, p-1, p\}$,
   (a) Derive the kth loading vector $\mathbf{v}_k(j)$ using Algorithm 1;
   (b) Calculate the CPEV by the first k PCs as described in Section 2.3;
2. Plot CPEV as a function of j and select the degree of sparsity j_0 as the largest j such that CPEV does not drop too much (for example, less than 5% or 10%) from its peak value at $j = 0$.

Our experience shows that this ad hoc approach works reasonably well. See Sections 3.4 and 6.1 for examples of its application. Obviously application of our ad hoc approach requires experience and personal judgment, similar to using the screeplot in deciding on the number of important PCs in standard PCA.

3. Synthetic examples

3.1. Data generation from a sparse PCA model

A straightforward way to evaluate a sparse PCA procedure is to apply it to data whose covariance matrix actually has sparse eigenvectors. We describe here a general scheme to generate such data. Suppose we want to generate data from R^p such that the q ($q < p$) leading eigenvectors of the covariance matrix $\mathbf{\Sigma}$ are sparse. Denote the first q eigenvectors as $\mathbf{v}_1, \ldots, \mathbf{v}_q$, which are specified to be sparse and orthonormal. The remaining $p - q$ eigenvectors are not specified to be sparse. Denote the positive eigenvalues of $\mathbf{\Sigma}$ in decreasing order as $c_1, \ldots, c_p$.

We first need to generate the other $q - p$ orthonormal eigenvectors of $\mathbf{\Sigma}$. To this end, form a full-rank matrix $\mathbf{V}^* = [\mathbf{v}_1, \ldots, \mathbf{v}_q, \mathbf{v}^*_{q+1}, \ldots, \mathbf{v}^*_p]$, where $\mathbf{v}_1, \ldots, \mathbf{v}_q$ are the pre-specified sparse eigenvectors and $\mathbf{v}^*_{q+1}, \ldots, \mathbf{v}^*_p$ are arbitrary. For example, the vectors $\mathbf{v}^*_{q+1}, \ldots, \mathbf{v}^*_p$ can be randomly drawn from $U(0, 1)$; if $\mathbf{V}^*$ is not of full-rank for one random draw, we can draw another set of vectors. Then, we apply the Gram–Schmidt orthogonalization method to $\mathbf{V}^*$ to obtain an orthogonal matrix $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_q, \mathbf{v}_{q+1}, \ldots, \mathbf{v}_p]$, which is actually the matrix $\mathbf{Q}$ from the QR decomposition of $\mathbf{V}^*$. Given the orthogonal matrix $\mathbf{V}$, we form the covariance matrix $\mathbf{\Sigma}$ using the following eigen decomposition expression,

$$\mathbf{\Sigma} = c_1\mathbf{v}_1\mathbf{v}_1^T + c_2\mathbf{v}_2\mathbf{v}_2^T + c_3\mathbf{v}_3\mathbf{v}_3^T + \cdots + c_p\mathbf{v}_p\mathbf{v}_p^T = \mathbf{V}\mathbf{C}\mathbf{V}^T,$$

where $\mathbf{C} = \mathrm{diag}\{c_1, \ldots, c_p\}$ is the eigenvalue matrix. The first q eigenvectors of $\mathbf{\Sigma}$ are the pre-specified sparse vectors $\mathbf{v}_1, \ldots, \mathbf{v}_q$. To generate data from the covariance matrix $\mathbf{\Sigma}$, let $\mathbf{Z}$ be a random draw from $N(0, I_p)$ and $\mathbf{X} = \mathbf{V}\mathbf{C}^{1/2}\mathbf{Z}$, then $\mathrm{cov}(\mathbf{X}) = \mathbf{\Sigma}$, as desired.

3.2. Comparison of the soft, hard and SCAD thresholding

Example 1. We consider a covariance matrix with two specified sparse leading eigenvectors. The data are in R^{10} and generated as $\mathbf{X} \sim N(0, \mathbf{\Sigma}_1)$. Let

$$\widetilde{\mathbf{v}}_1 = (1, 1, 1, 1, 0, 0, 0, 0, 0.9, 0.9)^T, \qquad \widetilde{\mathbf{v}}_2 = (0, 0, 0, 0, 1, 1, 1, 1, -0.3, 0.3)^T.$$

Table 1
(Example 1) Comparison of PCA and sparse PCA methods: median angles between the extracted loading vectors and the truth, percentages of correctly/incorrectly identified zero loadings

Method	$\mathbf{v}_1$			$\mathbf{v}_2$		
	Median angle	Correct (%)	Incorrect (%)	Median angle	Correct (%)	Incorrect (%)
	$n = 30$					
PCA	15.05	0.17	0.00	28.83	0.00	1.00
sPCA-rSVD-soft	10.86	92.50	5.00	17.06	71.25	19.17
sPCA-rSVD-hard	7.50	90.50	6.33	17.14	70.50	19.67
sPCA-rSVD-SCAD	11.39	92.00	5.33	15.78	71.50	19.17
Simple	8.10	90.75	6.17	24.41	66.50	22.33
SPCA ($k = 2$)	13.71	91.50	5.83	28.94	67.75	21.67
SPCA ($k = 1$)	28.24	80.25	13.17			
	$n = 300$					
PCA	4.80	1	0	8.21	0.75	0.00
sPCA-rSVD-soft	2.48	100	0	5.54	98.00	1.50
sPCA-rSVD-hard	2.19	100	0	4.20	98.25	1.17
sPCA-rSVD-SCAD	2.19	100	0	4.54	98.00	1.33
Simple	2.48	100	0	5.88	95.50	3.00
SPCA ($k = 2$)	4.11	100	0	9.95	97.25	2.17
SPCA ($k = 1$)	7.71	100	0			

The first two eigenvectors of $\mathbf{\Sigma}_1$ are then chosen to be

$$\mathbf{v}_1 = \widetilde{\mathbf{v}}_1/\|\widetilde{\mathbf{v}}_1\| = (0.422, 0.422, 0.422, 0.422, 0, 0, 0, 0, 0.380, 0.380)^T,$$
$$\mathbf{v}_2 = \widetilde{\mathbf{v}}_2/\|\widetilde{\mathbf{v}}_2\| = (0, 0, 0, 0, 0.489, 0.489, 0.489, 0.489, -0.147, 0.147)^T,$$

both of which have a degree of sparsity of 4. The 10 eigenvalues of $\mathbf{\Sigma}_1$ are, respectively, 200, 100, 50, 50, 6, 5, 4, 3, 2 and 1. The first two eigenvectors explain about 70% of the total variance.

We simulate 100 data sets of size $n = 30$ and 300, respectively, with the covariance matrix being $\mathbf{\Sigma}_1$. For each simulated data set, the first two sparse loading vectors are calculated using the sPCA-rSVD procedures with the soft, hard and SCAD thresholding rules; the procedures are referred as *sPCA-rSVD-soft*, *sPCA-rSVD-hard* and *sPCA-rSVD-SCAD*, respectively. To facilitate later comparison with simple thresholding and SPCA, for which there is no automatic way of selecting the degree of sparsity of the PC loading vectors, the true degree of sparsity is used when applying the sPCA-rSVD procedures (referred to as the oracle methods below).

Table 1 reports the medians of the angles between the extracted loading vectors and the corresponding truth for each procedure, as well as the percentages of correctly/incorrectly identified zero loadings for the loading vectors. All the sPCA-rSVD procedures appear to perform reasonably well and give comparable results. Comparing with standard PCA, sPCA-rSVD results in smaller median angles, which suggests that sparsity does improve statistical efficiency.

The three sPCA-rSVD procedures have also been applied to the simulated data sets using the tuning parameters selected by the CV approach. The results are summarized in Table 2. Comparing with the results in Table 1, we see that the CV methods perform almost as good as the oracle methods for $\mathbf{v}_1$, and slightly worse for $\mathbf{v}_2$.

Table 2
(Example 1) Five-fold CV tuning parameter selection: median angles between the extracted loading vectors and the truth, percentages of correctly/incorrectly identified zero loadings

Method	$\mathbf{v}_1$			$\mathbf{v}_2$		
	Median angle	Correct (%)	Incorrect (%)	Median angle	Correct (%)	Incorrect (%)
	$n = 30$					
sPCA-rSVD-soft	11.91	45.00	2.33	23.28	46.50	12.50
sPCA-rSVD-hard	10.89	62.25	2.33	25.15	52.25	18.17
sPCA-rSVD-SCAD	10.68	45.25	2.50	22.40	43.25	12.83
	$n = 300$					
sPCA-rSVD-soft	2.95	69.00	0.00	6.09	44.00	1.17
sPCA-rSVD-hard	2.83	83.25	0.00	7.47	67.50	2.67
sPCA-rSVD-SCAD	2.83	74.75	0.00	5.90	57.25	1.33

Table 3
(Example 2) HDLSS simulation with $n = 50$ and $p = 500$: median angles between the extracted loading vectors and the truth, percentages of correctly/incorrectly identified zero loadings

Method	$\mathbf{v}_1$			$\mathbf{v}_2$		
	Median angle	Correct (%)	Incorrect (%)	Median angle	Correct (%)	Incorrect (%)
PCA	19.69	5.79	0.10	20.39	4.60	0.20
sPCA-rSVD-soft	1.36	99.59	20.00	1.66	99.59	20.00
sPCA-rSVD-hard	1.21	99.59	20.00	1.53	99.59	20.00
sPCA-rSVD-SCAD	1.21	99.59	20.00	1.53	99.59	20.00
sPCA-rSVD-soft-CV	1.82	98.97	12.20	1.95	98.89	13.00
sPCA-rSVD-hard-CV	1.98	98.98	11.70	2.14	98.95	11.40
sPCA-rSVD-SCAD-CV	2.05	98.85	10.30	1.85	98.88	11.90
SPCA ($k = 2$)	4.95	99.63	18.00	6.21	99.63	18.00
SPCA ($k = 1$)	44.21	99.43	28.00			

3.3. High-dimension-low-sample-size (HDLSS) settings

Example 2. The data are in R^p with $p = 500$ and generated as $\mathbf{X} \sim N(0, \mathbf{\Sigma}_2)$. Let $\widetilde{\mathbf{v}}_1$ and $\widetilde{\mathbf{v}}_2$ be two 500-dimensional vectors such that $\widetilde{\mathbf{v}}_{1k} = 1,\ k = 1, \ldots, 10$, and $\widetilde{\mathbf{v}}_{1k} = 0,\ k = 11, \ldots, 500$; and $\widetilde{\mathbf{v}}_{2k} = 0,\ k = 1, \ldots, 10$, $\widetilde{\mathbf{v}}_{2k} = 1,\ k = 11, \ldots, 20$, and $\widetilde{\mathbf{v}}_{2k} = 0,\ k = 21, \ldots, 500$. The first two eigenvectors of $\mathbf{\Sigma}_2$ are chosen to be $\mathbf{v}_1 = \widetilde{\mathbf{v}}_1/\|\widetilde{\mathbf{v}}_1\|$ and $\mathbf{v}_2 = \widetilde{\mathbf{v}}_2/\|\widetilde{\mathbf{v}}_2\|$. To make these two eigenvectors dominate, we let the eigenvalues be $c_1 = 400,\ c_2 = 300$ and $c_k = 1$ for $k = 3, \ldots, 500$. The simulation scheme of Section 3.1 is used to generate data.

We simulate 100 data sets of size $n = 50$ with $\mathbf{\Sigma}_2$ being the covariance matrix. The sPCA-rSVD-soft/hard/SCAD procedures are applied to these HDLSS data sets with the degree of sparsity being specified as the truth (the oracle method) or by the five-fold CV. The results are summarized in Table 3. The three thresholding rules have comparable performance. The sPCA-rSVD procedures

Table 4
(Example 3) Comparison of PCA and sPCA-rSVD-hard

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}	CPEV
PC1 PCA	−0.100	−0.100	−0.100	−0.099	0.400	0.400	0.400	0.400	0.400	0.401	60.3
sPCA-rSVD-hard	0	0	0	0	0.415	0.415	0.415	0.414	0.395	0.395	59.3
PC2 PCA	0.482	0.482	0.482	0.482	0.132	0.131	0.131	0.132	−0.023	−0.022	99.7
sPCA-rSVD-hard	0.5	0.5	0.5	0.5	0	0	0	0	0	0	98.5

with five-fold CV give almost as good results as with the oracle method. The effectiveness of introducing sparsity is apparent in improving statistical efficiency of extracting the PCs as evidenced in direct comparison with standard PCA.

3.4. An ad hoc approach to sparsity degree selection

We use the synthetic example in Zou et al. [20] to illustrate our ad hoc approach to sparsity degree selection.

Example 3. Ten variables are generated as follows:

$$X_i = V_j + \varepsilon_i, \quad \varepsilon_i \sim N(0, 1), \quad i = 1, \ldots, 10,$$

with $j = 1$ for $i = 1, \ldots, 4$, $j = 2$ for $i = 5, \ldots, 8$, $j = 3$ for $i = 9, 10$, and the three hidden factors V_1, V_2 and V_3 are created as:

$$V_1 \sim N(0, 290), \quad V_2 \sim N(0, 300), \quad V_3 = -0.3V_1 + 0.925V_2 + \varepsilon, \quad \varepsilon \sim N(0, 300).$$

The ε's and the V's are independent. We sample 5000 data points from the ten-dimensional distribution, instead of using the exact covariance matrix as done by Zou et al. [20]. Note that in this example the true covariance matrix does not have sparse loading vectors (Table 4). Sparse PCA extracts some sparse basis vectors to best approximate the original data.

There are essentially two underlying factors, V_1 and V_2, that are nearly equally important. Standard PCA suggests that the first two PCs explain about 99.7% of the total variance (Table 4). Zou et al. [20] argued that the number of zero elements should be 6 for both loading vectors; however, the first two sparse PCs then only explain about 80.3% of the variance. Under this specification, our procedure generates similar results as SPCA (not shown).

We now use the ad hoc approach (Section 2.5.2) along with sPCA-rSVD-hard to select the "optimal" degree of sparsity. Fig. 1 plots the CPEV as a function of the degree of sparsity for the first two PCs, which suggests that the degree of sparsity is 4 and 6, respectively, different from the suggestion of Zou et al. The two leading sparse PCs explain about 98.5% of the total variance, slightly less than the corresponding 99.7% obtained by the standard PCA. The extracted PC loadings are reported in Table 4. According to Fig. 1, it is perceivable to argue that the sparsity for PC2 is 4 as well, thus increasing the CPEV to 99.6%, almost the same as the standard PCA.

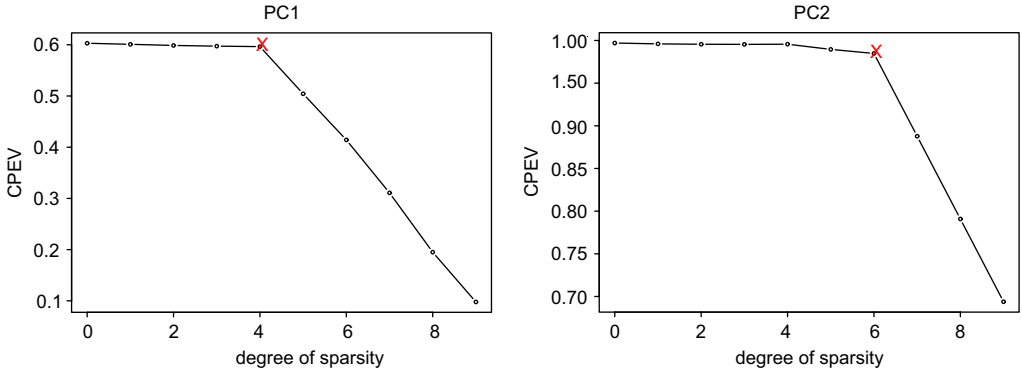

Fig. 1. (Example 3) Plot of CPEV as functions of sparsity. The degrees of sparsity of the two loadings are suggested to be 4 and 6, respectively.

4. sPCA-rSVD-soft vs. SPCA

This section provides some remarks on two sparse PCA methods: our sPCA-rSVD-soft and SPCA of Zou et al. [20]. Both approaches relate PCA to regression problems, and then employ a lasso (L_1) penalty to produce sparsity, as well as some iterative algorithm for computation.

Despite these similarities, there are major differences between the two approaches. First of all, they solve different optimization problems. As we discussed in Section 2.3, to get the first loading vector, sPCA-rSVD solves

$$\min_{\widetilde{\mathbf{v}}} \{-2\|\mathbf{X}\widetilde{\mathbf{v}}\| + \|\widetilde{\mathbf{v}}\|^2 + \lambda|\widetilde{\mathbf{v}}|_1\},$$

while the same argument yields that SPCA solves

$$\min_{\widetilde{\mathbf{v}}}\{-2\|\mathbf{X}^T\mathbf{X}\widetilde{\mathbf{v}}\| + \|\mathbf{X}\widetilde{\mathbf{v}}\|^2 + \lambda\|\widetilde{\mathbf{v}}\|^2 + \lambda_1|\widetilde{\mathbf{v}}|_1\}.$$

The objective functions of the two optimization problems are different.

The difference in computational algorithm is also significant. Operationally, SPCA solves the following optimization problem:

$$(\widehat{\mathbf{A}}, \widehat{\mathbf{B}}) = \operatorname*{argmin}_{\mathbf{A},\mathbf{B}} \sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{A}\mathbf{B}^T\mathbf{x}_i\|^2 + \lambda \sum_{j=1}^{k} \|\beta_j\|^2 + \sum_{j=1}^{k} \lambda_{1,j}\|\beta_j\|_1$$
$$\text{s.t.} \quad \mathbf{A}^T\mathbf{A} = I_{k\times k},$$

where $\mathbf{A} = [\alpha_1, \ldots, \alpha_k]$ and $\mathbf{B} = [\beta_1, \ldots, \beta_k]$ with k being the number of PCs to be extracted. This problem can be solved by alternating optimization over $\mathbf{A}$ and $\mathbf{B}$. For a fixed $\mathbf{A}$, the optimal $\mathbf{B}$ is obtained by solving the following *elastic net* problem [19],

$$\hat{\beta}_j = \operatorname*{argmin}_{\beta_j} \|Y_j^* - \mathbf{X}\beta_j\|^2 + \lambda\|\beta_j\|^2 + \lambda_{1,j}\|\beta_j\|_1,$$

where $Y_j^* = \mathbf{X}\alpha_j$. This problem can be solved using the LARS-EN algorithm. For a fixed $\mathbf{B}$, the optimal A can be obtained by minimizing $\sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{A}\mathbf{B}^T\mathbf{x}_i\|^2 = \|\mathbf{X} - \mathbf{X}\mathbf{B}\mathbf{A}^T\|^2$, subject to

Table 5
Computing time (in s) for Example 1 with 100 simulated data sets

	sPCA-rSVD-soft	sPCA-rSVD-hard	sPCA-rSVD-SCAD	SPCA ($k = 1$)	SPCA ($k = 2$)
$n = 30$	1.36	1.06	1.75	13.00	142.50
$n = 300$	1.77	1.61	1.85	10.32	60.40

$\mathbf{A}^T\mathbf{A} = \mathbf{I}_{k\times k}$. This is a Procrustes problem, and the solution is provided by considering the SVD, $\mathbf{X}^T\mathbf{X}\mathbf{B} = \mathbf{U}\mathbf{D}\mathbf{V}^T$, and setting $\widehat{\mathbf{A}} = \mathbf{U}\mathbf{V}^T$. Zou et al. [20] also developed the gene expression arrays SPCA algorithm to boost the computation for $p \gg n$ data.

The iterative algorithm in sPCA-rSVD-soft involves somewhat simpler building blocks, including simple linear regression and componentwise thresholding rules. This simplicity makes sPCA-rSVD-soft much easier to implement. The sPCA-rSVD-soft is also computationally less expensive since there is no need to perform an SVD in each iteration (Table 5). The sPCA-rSVD procedure treats both $n > p$ and $p \gg n$ cases in a unified manner.

We now discuss some numerical comparison of SPCA and sPCA-rSVD-soft. Both procedures are applied to the simulated data sets in Examples 1 and 2. Since SPCA does not have an automatic procedure for tuning parameter selection, we let the number of zero loadings equal to its true value for each loading vector. SPCA is implemented with $k = 1$ and 2, respectively, and generates different results (Tables 1 and 3). The sPCA-rSVD-soft does a better job than SPCA, especially the SPCA with $k = 1$. From boxplots of the estimated loadings (*not shown here*), we also observe that SPCA results in larger bias and variance when estimating the nonzero loadings. It is not well understood why the performance of SPCA appears to be sensitive to the choice of k. The sPCA-rSVD-soft does not have this problem since the PCs are extracted sequentially.

To get some idea of the computing cost of various sparse PCA procedures, Table 5 reports the CPU time used in producing the results in Example 1. The sPCA-rSVD seems to be computationally more efficient than SPCA. The fact that SPCA takes longer time to run for $n = 30$ than $n = 300$ is due to more iterations needed for algorithm convergence.

5. sPCA-rSVD-hard vs. simple thresholding

Simple thresholding is an ad hoc approach that sets zero the loadings whose absolute values below a certain threshold. Although frequently used in practice, simple thresholding can be potentially misleading in several aspects [2]. Our sPCA-rSVD-hard procedure also sets zero the loadings with small absolute values via the hard thresholding rule. In spite of its similarity to simple thresholding, sPCA-rSVD-hard works very well in simulated examples (Sections 3.2 and 3.3), and does not have the shortcomings of the simple thresholding discussed by Cadima and Jolliffe [2]. According to Table 1, while the performance of estimating $\mathbf{v}_1$ is comparable, sPCA-rSVD-hard improves over the simple thresholding for $\mathbf{v}_2$.

We think the difference is due to the way the thresholding is applied: sPCA-rSVD-hard applies hard thresholding on the loading vectors sequentially in an iterative manner; while simple thresholding extracts all the loadings first before applying hard thresholding. Therefore, the sparsity of the earlier PCs is not taken into account by simple thresholding when estimating the latter ones. Such a problem is avoided in sPCA-rSVD-hard. Moreover, the sequential extraction allows sPCA-rSVD-hard to take into account the relationship between the variables, while simple thresholding fails to do so.

Table 6
(Example 3) Illustration of the instability of simple thresholding

	Simple					sPCA-rSVD-hard
X_1	0	0	0	0	0	0
X_2	0	0	0	0	0	0
X_3	0	0	0	0	0	0
X_4	0	0	0	0	0	0
X_5	0	0.494	0.491	0	0.499	0.5
X_6	0.500	0	0	0	0.500	0.5
X_7	0.499	0	0	0.499	0.500	0.5
X_8	0	0.493	0.491	0.500	0.500	0.5
X_9	0.500	0.506	0.508	0.500	0	0
X_{10}	0.501	0.507	0.509	0.501	0	0

The first loading vector is extracted to have 6 zero components.

We now use Example 3 in Section 3.4 to further understand the difference. In the course of following Zou et al.'s suggestion to extract the first loading vector with 6 zero components, one shortcoming of simple thresholding is identified: its instability as a result of high correlation among the original variables. As we change random seed and simulate a new data matrix, the selection changes dramatically among X_5–X_{10}. These variables are highly correlated due to the high correlation between the underlying factors V_2 and V_3. Table 6 presents the loadings for five simulated data sets, and one can see the instability of simple thresholding clearly. For example, for the first data set, simple thresholding leaves out X_5 and X_8 while including X_9 and X_{10}. The result is rather misleading because X_5–X_8 are essentially the same, which should appear together. On the other hand, sPCA-rSVD-hard always selects these four variables, and the loadings remain very stable among the simulations. Table 6 also reports the average loading vector produced by sPCA-rSVD-hard, which is the same as the one obtained by SPCA.

6. Real examples

6.1. Pitprops data

Jeffers [9] used the pitprops data to illustrate the difficulty of interpreting PCs, which have 180 observations and 13 variables. The correlation matrix of the pitprops data has been used repeatedly in the literature to illustrate various sparse PCA methods [12,20]. Following the literature, below we apply our sPCA-rSVD approaches to the pitprops data to extract the *first six* sparse PC loading vectors.

Since the data matrix is a correlation matrix, we apply our procedure to its square-root matrix as justified by the discussion in Section 2.4. The ad hoc parameter selection procedure in Section 2.5.2 is used to select the tuning parameters. Below we present the result from sPCA-rSVD-soft. The CPEV selection plot is in Fig. 2, with the *subjectively* selected degree of sparsity marked for each PC, which are 6, 11, 9, 6, 11 and 10, respectively. Table 7 reports the loading vectors by PCA and sparse PCA. The loadings from sPCA-rSVD-soft are much more sparse than the regular loadings, yet still account for nearly the same amount of variance (84.5% vs. 87.0%). See Zou et al. [20] for sparse PC loadings obtained by simple thresholding, SCoTLASS and SPCA.

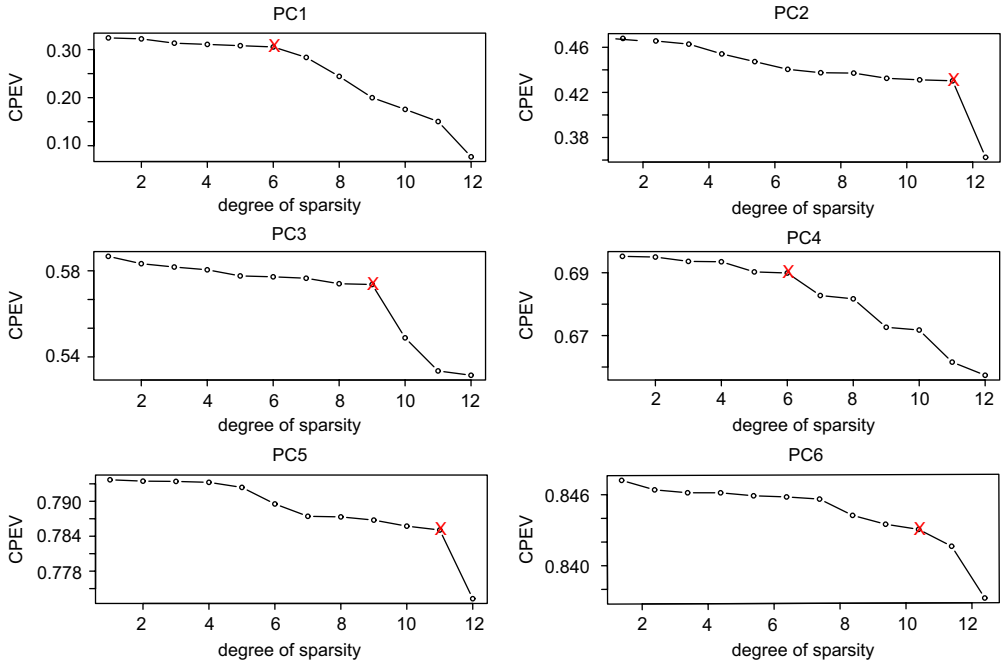

Fig. 2. (Pitprops data) CPEV plot for sPCA-rSVD-soft with selected degrees of sparsity marked.

Table 7
(Pitprops data) Loadings of the first six PCs by PCA and sPCA-rSVD-soft

Variable	PCA						sPCA-rSVD-soft					
	PC1	PC2	PC3	PC4	PC5	PC6	PC1	PC2	PC3	PC4	PC5	PC6
x_1	−0.404	0.218	−0.207	0.091	−0.083	0.120	−0.449	0	0	−0.114	0	0
x_2	−0.406	0.186	−0.235	0.103	−0.113	0.163	−0.460	0	0	−0.102	0	0
x_3	−0.124	0.541	0.141	−0.078	0.350	−0.276	0	−0.707	0	0	0	0
x_4	−0.173	0.456	0.352	−0.055	0.356	−0.054	0	−0.707	0	0	0	0
x_5	−0.057	−0.170	0.481	−0.049	0.176	0.626	0	0	0.550	0	0	−0.744
x_6	−0.284	−0.014	0.475	0.063	−0.316	0.052	−0.199	0	0.546	−0.176	0	0
x_7	−0.400	−0.190	0.253	0.065	−0.215	0.003	−0.399	0	0.366	0	0	0
x_8	−0.294	−0.189	−0.243	−0.286	0.185	−0.055	−0.279	0	0	0.422	0	0
x_9	−0.357	0.017	−0.208	−0.097	−0.106	0.034	−0.380	0	0	0	0	0
x_{10}	−0.379	−0.248	−0.119	0.205	0.156	−0.173	−0.407	0	0	0.283	0.231	0
x_{11}	0.011	0.205	−0.070	−0.804	−0.343	0.175	0	0	0	0	−0.973	0
x_{12}	0.115	0.343	0.092	0.301	−0.600	−0.170	0	0	0	−0.785	0	0.161
x_{13}	0.113	0.309	−0.326	0.303	0.080	0.626	0	0	−0.515	−0.265	0	−0.648
Sparsity	0	0	0	0	0	0	6	11	9	6	11	10
CPEV	32.5	50.7	65.2	73.7	80.7	87.0	30.6	45.0	59.0	70.0	78.5	84.5

The degrees of sparsity of PCs are selected according to Fig. 2.

6.2. *NCI60 cell line data*

Microarray gene expression data are usually HDLSS data, where the expression levels of thousands of genes are measured simultaneously over a small number of samples. The problem of gene selection is of great interest to identify subsets of "intrinsic" or "disease" genes which are

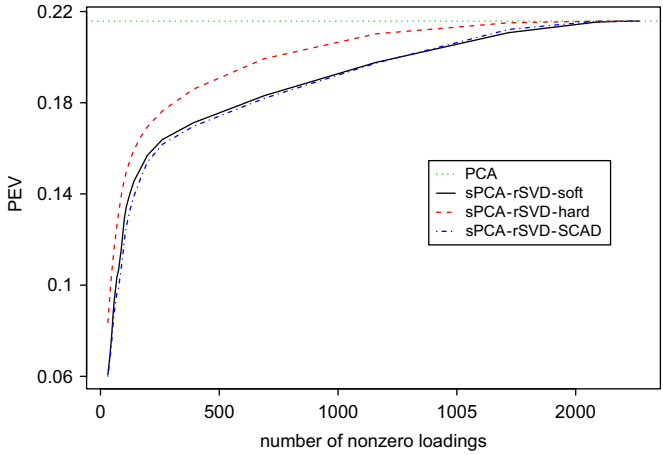

Fig. 3. (NCI60 data) Plot of PEV as a function of number of non-zero loadings for the first PC.

biologically relevant to certain outcomes, such as cancer types, and to use the subsets for further studies, such as to classify cancer types. Several gene selection methods in the literature build upon PCA (or SVD), such as *gene-shaving* [8] and *meta-genes* [18].

We use sparse PCA as a gene selection method and investigate the performance of various sparse PCA methods using the NCI60 cell line data, available at `http://discoer.nci.nih.gov/`, where measurements were made using two platforms, cDNA and Affy. There are 60 common biological samples measured on each of the two platforms with 2267 common genes. Benito et al. [1] proposed to use DWD [15] as a systematic bias adjustment method to eliminate the platform effect of the NCI60 data. Thus, the processed data have $p = 2267$ genes and $n = 120$ samples. The first PC explains about 21% of the total variance.

We apply our sPCA-rSVD procedures on the processed data to extract the first sparse PC. Fig. 3 plots the percentage of explained variance (PEV) as a function of number of non-zero loadings. As one can see, the PEV curves for sPCA-rSVD-soft/SCAD are very similar, both of which are consistently below the curve for sPCA-rSVD-hard. This suggests that, using the same number of genes, the sparse PC from sPCA-rSVD-hard always explains more variance. According to the sPCA-rSVD-hard curve, using as few as 200 to 300 genes, the sparse PC can account for 17–18% of the total variance. Compared with the 21%, explained by the standard PC, the cost is affordable. Simple thresholding and SPCA are also applied to this data set, and their PEV curves are similar to the sPCA-rSVD-hard/soft curves, respectively. Note that such similarities may not hold in general as shown in previous sections.

7. Discussion

Zou et al. [20] remarked that a good sparse PCA method should (at least) possess the following properties: without any sparsity constraint, the method reduces to PCA; it is computationally efficient for both small p and large p data; it avoids misidentifying important variables. We have developed a new sparse PCA procedure based on regularized SVD that have all these properties. Moreover, our procedure is statistically more efficient than standard PCA if the data are actually from a sparse PCA model (Tables 1 and 3). Our general framework allows using different

penalties. In addition to the soft/hard thresholding and SCAD penalties that we have considered, one can apply the Bridge penalty [6] or the hybrid penalty that combines the L_0 and L_1 penalties [14].

When the soft thresholding penalty is used, our procedure has similarities to the SPCA of Zou et al. [20]. On the other hand, as we have shown in Section 4, the two approaches exhibit major differences. It appears that our sPCA-rSVD procedure is more efficient, both statistically and computationally. One attractive feature of the sPCA-rSVD procedure is its simplicity. It can be viewed as a simple modification—adding a thresholding step—of the alternating least squares algorithm for computing SVD. There is no need to apply the sophisticated LARS-EN algorithm and solve a Procrustes problem during each iteration.

When the hard thresholding penalty is used, our procedure has similarities to the often-used simple thresholding approach. Our procedure can be roughly described as "iterative componentwise simple thresholding." It shares the simplicity of the simple thresholding; furthermore, through iteration and sequential PC extraction, it avoids misidentification of "underlying" important variables possibly masked by high correlation, a serious drawback of simple thresholding.

Appendix A.

Lemma A.1. *Let* $\widetilde{\mathbf{v}}' = \widetilde{\mathbf{v}}/\|\widetilde{\mathbf{v}}\|$ *and* $\widetilde{\mathbf{V}} = [\widetilde{\mathbf{v}}'; \widetilde{\mathbf{v}}_\perp]$ *be a* $p \times p$ *orthogonal matrix. Then we have*

$$\begin{aligned}\|\mathbf{X} - \widetilde{\mathbf{u}}\widetilde{\mathbf{v}}^T\|_F^2 &= \|\mathbf{X}\widetilde{\mathbf{V}} - \widetilde{\mathbf{u}}\widetilde{\mathbf{v}}^T\widetilde{\mathbf{V}}\|_F^2 = \|[\mathbf{X}\widetilde{\mathbf{v}}'; \mathbf{X}\widetilde{\mathbf{v}}_\perp] - [\widetilde{\mathbf{u}}\|\widetilde{\mathbf{v}}\|; 0]\|_F^2 \\ &= \|\mathbf{X}\widetilde{\mathbf{v}}' - \widetilde{\mathbf{u}}\|\widetilde{\mathbf{v}}\|\|^2 + \|\mathbf{X}\widetilde{\mathbf{v}}_\perp\|_F^2 \\ &= \|\widetilde{\mathbf{v}}\|^2\|\mathbf{X}\widetilde{\mathbf{v}}/\|\widetilde{\mathbf{v}}\|^2 - \widetilde{\mathbf{u}}\|^2 + \|\mathbf{X}\widetilde{\mathbf{v}}_\perp\|_F^2.\end{aligned}$$

Thus, for a fixed $\widetilde{\mathbf{v}}$*, minimization of* (2) *reduces to minimization of* $\|\mathbf{X}\widetilde{\mathbf{v}}/\|\widetilde{\mathbf{v}}\|^2 - \widetilde{\mathbf{u}}\|^2$*. On the other hand, we have that* $\min_{\widetilde{\mathbf{u}}:\|\widetilde{\mathbf{u}}\|=1} \|\xi - \widetilde{\mathbf{u}}\|$ *is solved by* $\widetilde{\mathbf{u}} = \xi/\|\xi\|$*. In fact,* $\|\xi - \widetilde{\mathbf{u}}\|^2 = \|\xi\|^2 + 1 - 2\langle\xi, \widetilde{\mathbf{u}}\rangle$*, since* $\|\widetilde{\mathbf{u}}\| = 1$*. By the Cauchy–Schwarz inequality,* $\langle\xi, \widetilde{\mathbf{u}}\rangle \leqslant \|\xi\|$*, with equality if and only if* $\widetilde{\mathbf{u}} = c\,\xi$*. Hence,* $\|\widetilde{\mathbf{u}}\| = 1$ *implies that* $c = 1/\|\xi\|$*. Combining all these, we obtain Lemma* A.1.

Theorem A.1. *Let* $\mathbf{H}_k = \mathbf{V}_k\left(\mathbf{V}_k^T\mathbf{V}_k\right)^{-1}\mathbf{V}_k^T$ *and denote the ith row of* $\mathbf{X}$ *as* x_i^T*. The projection of* x_i *onto the linear space spanned by the first k sparse PCs is* $\mathbf{H}_k x_i$*. It is easily seen that* $\mathrm{tr}\left(\mathbf{X}_k^T\mathbf{X}_k\right) = \sum_{i=1}^n \|\mathbf{H}_k x_i\|^2$ *and* $\mathrm{tr}\left(\mathbf{X}^T\mathbf{X}\right) = \sum_{i=1}^n \|x_i\|^2$*. Since* $\|\mathbf{H}_k x_i\| \leqslant \|\mathbf{H}_{k+1}x_i\| \leqslant \|x_i\|$*, the desired result follows.*

Lemma A.2. *Simple calculation yields*

$$\|\mathbf{X} - \widetilde{\mathbf{u}}\widetilde{\mathbf{v}}^T\|_F^2 = \mathrm{tr}(\mathbf{X}\mathbf{X}^T) - 2\widetilde{\mathbf{v}}^T\mathbf{X}^T\widetilde{\mathbf{u}} + \|\widetilde{\mathbf{u}}\|^2\|\widetilde{\mathbf{v}}\|^2.$$

Thus, minimization of (2) *is equivalent to minimization of*

$$-2\widetilde{\mathbf{v}}^T\mathbf{X}^T\widetilde{\mathbf{u}} + \|\widetilde{\mathbf{u}}\|^2\|\widetilde{\mathbf{v}}\|^2 + P_\lambda(\widetilde{\mathbf{v}}). \tag{6}$$

According to Lemma A.1*, for a fixed* $\widetilde{\mathbf{v}}$*, the minimizer of* (6) *is* $\widetilde{\mathbf{u}} = \mathbf{X}\widetilde{\mathbf{v}}/\|\mathbf{X}\widetilde{\mathbf{v}}\|$*, which in turn suggests that minimizing* (6) *is equivalent to minimizing* $-2\|\mathbf{X}\widetilde{\mathbf{v}}\| + \|\widetilde{\mathbf{v}}\|^2 + P_\lambda(\widetilde{\mathbf{v}})$*.*

Theorem A.2. *According to our procedure,* $\widetilde{\mathbf{v}}_1$ *is the minimizer of* (6) *and* $\widetilde{\mathbf{u}}_1 = \mathbf{X}\widetilde{\mathbf{v}}_1/\|\mathbf{X}\widetilde{\mathbf{v}}_1\|$*. Lemma* A.2 *shows that* $\widetilde{\mathbf{v}}_1$ *depends on* $\mathbf{X}$ *only through* $\mathbf{X}^T\mathbf{X}$*. Our procedure derives the sparse loading vectors sequentially. Form the residual matrix* $\mathbf{X}_1 = \mathbf{X} - \widetilde{\mathbf{u}}_1\widetilde{\mathbf{v}}_1^T = \mathbf{X}\left(I - \widetilde{\mathbf{v}}_1\widetilde{\mathbf{v}}_1^T/\|\mathbf{X}\widetilde{\mathbf{v}}_1\|\right)$*.*

The second sparse loading vector $\widetilde{\mathbf{v}}_2$ *is the minimizer of* (6) *with* $\mathbf{X}$ *replaced by* $\mathbf{X}_1$. *Thus*, $\widetilde{\mathbf{v}}_2$ *depends on* $\mathbf{X}_1$ *only through* $\mathbf{X}_1^T\mathbf{X}_1 = \left(I - \widetilde{\mathbf{v}}_1\widetilde{\mathbf{v}}_1^T/\|\mathbf{X}\widetilde{\mathbf{v}}_1\|\right)\mathbf{X}^T\mathbf{X}\left(I - \widetilde{\mathbf{v}}_1\widetilde{\mathbf{v}}_1^T/\|\mathbf{X}\widetilde{\mathbf{v}}_1\|\right)$, *which implies that* $\widetilde{\mathbf{v}}_2$ *depends on* $\mathbf{X}$ *only through* $\mathbf{X}^T\mathbf{X}$. *Moreover*,

$$\widetilde{\mathbf{u}}_2 = \mathbf{X}_1\widetilde{\mathbf{v}}_2/\|\mathbf{X}_1\widetilde{\mathbf{v}}_2\| = \mathbf{X}\left(I - \widetilde{\mathbf{v}}_1\widetilde{\mathbf{v}}_1^T/\|\mathbf{X}\widetilde{\mathbf{v}}_1\|\right)\widetilde{\mathbf{v}}_2/\|\mathbf{X}_1\widetilde{\mathbf{v}}_2\|.$$

By induction, we can show that the residual matrix $\mathbf{X}_{k-1}$ *of the first* $k-1$ *PCs is*

$$\mathbf{X}_{k-1} = \mathbf{X}\prod_{i=1}^{k-1}\left(I - \widetilde{\mathbf{v}}_i\widetilde{\mathbf{v}}_i^T/\|\mathbf{X}_{i-1}\widetilde{\mathbf{v}}_i\|\right),$$

where $\mathbf{X}_0 \equiv \mathbf{X}$. *Furthermore*, $\widetilde{\mathbf{v}}_k$ *depends on* $\mathbf{X}$ *only through* $\mathbf{X}^T\mathbf{X}$, *and*

$$\widetilde{\mathbf{u}}_k = \mathbf{X}_{k-1}\widetilde{\mathbf{v}}_k/\|\mathbf{X}_{k-1}\widetilde{\mathbf{v}}_k\| = \mathbf{X}\prod_{i=1}^{k-1}\left(I - \widetilde{\mathbf{v}}_i\widetilde{\mathbf{v}}_i^T/\|\mathbf{X}_{i-1}\widetilde{\mathbf{v}}_i\|\right)\widetilde{\mathbf{v}}_k/\|\mathbf{X}_{k-1}\widetilde{\mathbf{v}}_k\|.$$

As a result, $\mathbf{v}_1, \ldots, \mathbf{v}_k$ *depend on* $\mathbf{X}$ *only through* $\mathbf{X}^T\mathbf{X}$.

Theorem A.3. *Let* $\mathbf{V}_k = [\mathbf{v}_1, \ldots, \mathbf{v}_k]$ *be the loading matrix of the first k loading vectors. Then, as discussed in Section* 2.3, *the corresponding projection is* $\mathbf{X}_k = \mathbf{X}\mathbf{V}_k\left(\mathbf{V}_k^T\mathbf{V}_k\right)^{-1}\mathbf{V}_k^T \equiv \mathbf{X}\mathbf{H}_k$. *It follows that*

$$\operatorname{tr}\left(\mathbf{X}_k^T\mathbf{X}_k\right) = \operatorname{tr}\left(\mathbf{X}_k\mathbf{X}_k^T\right) = \operatorname{tr}\left(\mathbf{X}\mathbf{H}_k^2\mathbf{X}^T\right) = \operatorname{tr}\left(\mathbf{X}^T\mathbf{X}\mathbf{H}_k\right).$$

According to Theorem A.2, $\mathbf{H}_k$ *depends on* $\mathbf{X}$ *only through* $\mathbf{X}^T\mathbf{X}$, *so does* $\operatorname{tr}\left(\mathbf{X}_k^T\mathbf{X}_k\right)$.

Acknowledgment

The authors want to extend grateful thanks to the Editors and the reviewers whose comments have greatly improved the scope and presentation of the paper. Haipeng Shen's work is partially supported by National Science Foundation (NSF) grant DMS-0606577. Jianhua Z. Huang's work is partially supported by NSF grant DMS-0606580 and a grant from NCI.

References

[1] M. Benito, J. Parker, Q. Du, J. Wu, D. Xiang, C.M. Perou, J.S. Marron, Adjustment of systematic microarray data biases, Bioinformatics 20 (2004) 105–114.
[2] J. Cadima, I.T. Jolliffe, Loadings and correlations in the interpretation of principal components, J. Appl. Statist. 22 (1995) 203–214.
[3] D. Donoho, I. Johnstone, Ideal spatial adaptation via wavelet shrinkage, Biometrika 81 (1994) 425–455.
[4] C. Eckart, G. Young, The approximation of one matrix by another of lower rank, Psychometrika 1 (1936) 211–218.
[5] J. Fan, R. Li, Variable selection via nonconcave penalized likelihood and its oracle properties, J. Amer. Statist. Assoc. 96 (2001) 1348–1360.
[6] I. Frank, J. Friedman, A statistical view of some chemometrics regression tools, Technometrics 35 (1993) 109–135.
[7] K.R. Gabriel, S. Zamir, Lower rank approximation of matrices by least squares with any choice of weights, Technometrics 21 (1979) 489–498.
[8] T. Hastie, R. Tibshirani, A. Eisen, R. Levy, L. Staudt, D. Chan, P. Brown, Gene shaving as a method for identifying distinct sets of genes with similar expression patterns, Genome Biology 1 (2000) 1–21.
[9] J. Jeffers, Two case studies in the application of principal component, Appl. Statist. 16 (1967) 225–236.

[10] I.T. Jolliffe, Rotation of principal components: choice of normalization constraints, J. Appl. Statist. 22 (1995) 29–35.
[11] I.T. Jolliffe, Principal Component Analysis, second ed., Springer, New York, 2002.
[12] I.T. Jolliffe, N.T. Trendafilov, M. Uddin, A modified principal component technique based on the LASSO, J. Comput. Graph. Statist. 12 (2003) 531–547.
[13] I.T. Jolliffe, M. Uddin, The simplified component technique: an alternative to rotated principal components, J. Comput. Graph. Statist. 9 (2000) 689–710.
[14] Y. Liu, Y. Wu, Variable selection via a combination of the L_0 and L_1 penalties, J. Comput. Graph. Statist., 2007, accepted for publication.
[15] J.S. Marron, M. Todd, J. Ahn, Distance weighted discrimination, J. Amer. Statist. Assoc. 2005, tentatively accepted for publication.
[16] R. Tibshirani, Regression shrinkage and selection via the lasso, J. Roy. Statist. Soc. Ser. B 58 (1996) 267–288.
[17] S. Vines, Simple principal components, Appl. Statist. 49 (2000) 441–451.
[18] M. West, Bayesian factor regression models in the "Large p, Small n" paradigm, Bayesian Statist. 7 (2003) 723–732.
[19] H. Zou, T. Hastie, Regularization and variable selection via the elastic net, J. Roy. Statist. Soc. Ser. B 67 (2005) 301–320.
[20] H. Zou, T. Hastie, R. Tibshirani, Sparse principal component analysis, J. Comput. Graph. Statist. 15 (2006) 265–286.