ALTERNATING DIRECTION METHOD OF MULTIPLIERS FOR SPARSE PRINCIPAL COMPONENT ANALYSIS

SHIQIAN MA*

November 28, 2011

Abstract. We consider a convex relaxation of sparse principal component analysis proposed by d'Aspremont et al. in [9]. This convex relaxation is a nonsmooth semidefinite programming problem in which the ℓ_1 norm of the desired matrix is imposed in either the objective function or the constraint to improve the sparsity of the resulting matrix. The sparse principal component is obtained by a rank-one decomposition of the resulting sparse matrix. We propose an alternating direction method based on a variable-splitting technique and an augmented Lagrangian framework for solving this nonsmooth semidefinite programming problem. In contrast to the first-order method proposed in [9] that solves approximately the dual problem of the original semidefinite programming problem, our method deals with the primal problem directly and solves it exactly, which guarantees that the resulting matrix is a sparse matrix. Global convergence result is established for the proposed method. Numerical results on both synthetic problems and the real applications from classification of text data and senate voting data are reported to demonstrate the efficacy of our method.

Key words. Sparse PCA, Semidefinite Programming, Alternating Direction Method, Augmented Lagrangian Method, Deflation, Projection onto the Simplex

AMS subject classifications. 62H25, 90C25, 90C22, 62H30, 65K05

1. Introduction.

Principal component analysis (PCA) plays an important role in applications arising from data analysis, dimension reduction and bioinformatics etc. PCA finds a few linear combinations of the original variables. These linear combinations, which are called principal components (PC), are orthogonal to each other and explain most of the variance of the data. Specifically, for a given data matrix $M \in \mathbb{R}^{p\times n}$ which consists of n samples of the p variables, PCA corresponds to a singular value decomposition (SVD) of M or an eigenvalue decomposition of the sample covariance matrix $\Sigma = MM^\top \in \mathbb{R}^{p\times p}$. Thus, for a given sample covariance matrix Σ, PCA is usually formulated as an eigenvalue problem:

$$x^* := \arg\max \quad x^\top \Sigma x, \quad \text{s.t.} \quad \|x\|_2 \le 1, \tag{1.1}$$

where $\|x\|_2$ is the Euclidean norm of vector x. Problem (1.1) gives the eigenvector that corresponds to the largest eigenvalue of Σ. However, the loading vector x^* is not expected to have many zero coefficients. This makes it hard to explain the PCs. For example, in the text classification problem, we are given a binary data matrix $M \in \mathbb{R}^{p\times n}$ that records the occurrences of p words in n postings. That is, $M_{ij} = 1$ if the i-th word appears in the j-th posting and $M_{ij} = 0$ if the i-th word does not appear in the j-th posting. The standard PCA cannot tell which words contribute most to the explained variance since the loadings are linear combinations of all the variables. Thus, sparse PCs are needed because it is easier to analyze which variables contribute most to the explained variance.

Many techniques were proposed to extract sparse PCs from given sample covariance matrix Σ or sample data matrix M. One natural thought is to impose a cardinality constraint to (1.1), which leads to the following formulation for sparse PCA:

$$x^* := \arg\max \quad x^\top \Sigma x, \quad \text{s.t.} \quad \|x\|_2 \le 1, \quad \|x\|_0 \le K, \tag{1.2}$$

*Institute for Mathematics and Its Applications, 400 Lind Hall, 207 Church Street SE, University of Minnesota, Minneapolis, MN 55455, USA. Email: maxxa007@ima.umn.edu.

where $\|x\|_0$ (the ℓ_0 norm of x) counts the number of nonzeros of x and the integer K controls the sparsity of the solution. Since the cardinality constraint $\|x\|_0 \leq K$ makes the problem numerically intractable, many different models were proposed in the literature to overcome this difficulty.

In [9], d'Aspremont et al. proposed to approximately solve (1.2) by its convex relaxation, which is a nonsmooth semidefinite programming (SDP) problem. This is the first work that attempts to approximately solve (1.2) by a convex problem. The SDP formulation is based on the lifting and projection technique, which is a standard technique in using SDP to approximate combinatorial problems (see e.g., [1, 3, 34]). Note that if we denote $X = xx^\top$, then (1.2) can be rewritten as

$$\max_{X\in\mathbb{R}^{p\times p}} \{\langle \Sigma, X\rangle, \text{ s.t. } \mathbf{Tr}(X) = 1, \|X\|_0 \leq K^2, X \succeq 0, \operatorname{rank}(X) = 1\}, \tag{1.3}$$

where $\mathbf{Tr}(X)$ denotes the trace of matrix X. The rank constraint is then dropped and the cardinality constraint is replaced by ℓ_1 norm constraint, and this leads to following convex problem, which is an SDP.

$$\max_{X\in\mathbb{R}^{p\times p}} \{\langle \Sigma, X\rangle, \text{ s.t. } \mathbf{Tr}(X) = 1, \|X\|_1 \leq K, X \succeq 0\}, \tag{1.4}$$

where the ℓ_1 norm of X is defined as $\|X\|_1 := \sum_{ij} |X_{ij}|$ and using the convex constraint $\|X\|_1 \leq K$ to impose the sparsity of the solution is inspired by the recent emergence of compressed sensing (see e.g., [5, 10]). Note that $\|X\|_1 \leq K$ is used in (1.4) instead of $\|X\|_1 \leq K^2$. This is due to the fact that, when $X = xx^\top$ and $\mathbf{Tr}(X) = 1$, we have $\|X\|_F = 1$, and also that if $\|X\|_0 \leq K^2$, then $\|X\|_1 \leq K\|X\|_F$. After the optimal solution X^* to (1.4) is obtained, the vector $\hat{x}$ from the rank-one decomposition of X^*, i.e., $X^* = \hat{x}\hat{x}^\top$ is used as an approximation of the solution of (1.2). This is the whole procedure of the lifting and projection technique. Although some standard methods such as interior point methods can be used to solve the SDP (1.4) (see e.g., [1, 3, 34]), it is not wise to do so because (1.4) is a nonsmooth problem, and transferring it to a standard SDP increases the size of the problem dramatically.

It is known that (1.4) is equivalent to the following problem with an appropriately chosen parameter $\rho > 0$:

$$\max_{X\in\mathbb{R}^{p\times p}} \{\langle \Sigma, X\rangle - \rho\|X\|_1 \text{ s.t. } \mathbf{Tr}(X) = 1, X \succeq 0\}. \tag{1.5}$$

Note that (1.5) can be rewritten as

$$\max_{X\succeq 0, \mathbf{Tr}(X)=1} \min_{\|U\|_\infty \leq \rho} \langle \Sigma + U, X\rangle, \tag{1.6}$$

where $\|U\|_\infty$ denotes the largest component of U in magnitude, i.e., $\|U\|_\infty = \max_{ij} |U_{ij}|$. The dual problem of (1.5) is given by interchanging the max and min in (1.6), i.e.,

$$\min_{\|U\|_\infty \leq \rho} \max_{X\succeq 0, \mathbf{Tr}(X)=1} \langle \Sigma + U, X\rangle,$$

which can be further reduced to

$$\min_{U\in\mathbb{R}^{p\times p}} \lambda_{\max}(\Sigma + U), \qquad \text{s.t.} \qquad \|U\|_\infty \leq \rho, \tag{1.7}$$

where $\lambda_{\max}(Z)$ denotes the largest eigenvalue of matrix Z. d'Aspremont et al. [9] proposed to solve the dual problem (1.7) using Nesterov's first-order algorithm (see e.g., [27, 28]), which is an accelerated projected gradient method. However, since the objective function of (1.7) is nonsmooth, one needs to smooth it in order to apply Nesterov's algorithm. Thus, the authors of [9] actually solve an approximation of the dual problem (1.7), which can be formulated as follows.

$$\min \quad f_\mu(U), \text{ s.t. } \|U\|_\infty \leq \rho, \tag{1.8}$$

where $\mu > 0$ is the smoothing parameter, $f_\mu(U) := \max\{\langle \Sigma + U, X\rangle - \mu d(X), \text{ s.t. } \mathbf{Tr}(X) = 1, X \succeq 0\}$ and $d(X) := \mathbf{Tr}(X \log X) + \log(n)$. It is shown in [9] that an approximate solution X^k to the primal problem (1.5) can be obtained by $X^k = \nabla f_\mu(U^k)$, where U^k is an approximate solution of (1.8). It is easy to see that X^k is not guaranteed to be a sparse matrix. Besides, although there is no duality gap between (1.5) and (1.7), the authors solve an approximation of (1.7). It needs also to be noted that Nesterov's algorithm used in [9] cannot solve the constrained problem (1.4). Although (1.4) and (1.5) are equivalent with appropriately chosen parameters K and ρ, in many applications, it is usually easier to choose an appropriate K since we know how many nonzeros are preferred in the sparse PCs.

Nonconvex reformulations of (1.2) include the followsings. Zou et al. [42] considered a regression type formulation of (1.2) with Lasso and elastic net regularizations. d'Aspremont et al. [8] considered a penalty version of (1.2),

$$\phi(\rho) \equiv \max_{\|x\|_2 \leq 1} x^\top \Sigma x - \rho \|x\|_0. \tag{1.9}$$

d'Aspremont [8] showed that (1.9) is equivalent to the following problem that maximizes a convex function over spherical constraint:

$$\phi(\rho) = \max_{\|x\|_2 = 1} \sum_{i=1}^{p} ((a_i^\top x)^2 - \rho)_+, \tag{1.10}$$

where $(\alpha)_+ := \max\{\alpha, 0\}$, $\Sigma = A^\top A$ and a_i is the i-th column of $A \in \mathbb{R}^{p\times p}$. Clearly, (1.10) is a non-convex problem. d'Aspremont et al. thus proposed in [8] to solve (1.10) by a greedy method. Journee et al. [22] considered the same formulation (1.10) and proposed a gradient type method, which is actually a generalized power method, to solve it.

Note that (1.2) only gives the largest sparse PC. In many applications, several leading sparse PCs are needed in order to explain more variance. Multiple sparse PCs are usually found by solving a sequence of sparse PCA problems (1.2), with Σ constructed via the so-called *deflation* technique for each sparse PC. Lu and Zhang [24] proposed the following model to compute the leading r sparse PCs of Σ simultaneously:

$$\begin{array}{ll} \max\limits_{V \in \mathbb{R}^{p\times r}} & \mathbf{Tr}(V^\top \Sigma V) - \rho \|V\|_1 \\ \text{s.t.} & |V_i^\top \Sigma V_j| \leq \Delta_{ij}, \forall i \neq j, \\ & V^\top V = I, \end{array} \tag{1.11}$$

where each column of V corresponds to a loading vector of the sample covariance matrix Σ and $\Delta_{ij} \geq 0 (i \neq j)$ are the parameters that control the correlation of the PCs. Lu and Zhang [24] proposed an augmented Lagrangian method to solve (1.11). Note that for these nonconvex formulations, algorithms proposed in the literature usually have only local convergence and global convergence is not guaranteed.

In this paper, we propose an alternating direction method based on a variable-splitting technique and an augmented Lagrangian framework for solving directly the primal problems (1.4) and (1.5). Our method solves two subproblems in each iteration. One subproblem has a closed-form solution that corresponds to projecting a given matrix onto the simplex of the cone of semidefinite matrices. This projection requires an eigenvalue decomposition. The other subproblem has a closed-form solution that corresponds to a vector shrinkage operation (for Problem (1.5)) or a projection onto the ℓ_1 ball (for Problem (1.4)). Thus, our method produces two iterative points at each iteration. One iterative point is a semidefinite matrix with trace equal to one and the other one is a sparse matrix. Eventually these two points will converge to the same point, and thus we get an optimal solution which is a sparse and semidefinite matrix. Compared with the Nesterov's first-order method suggested in [9] for solving the approximated dual problem (1.8), our method can solve the nonsmooth primal problems (1.4) and (1.5) uniformly. Also, since we deal with the primal problems directly, the ℓ_1 norm in the constraint or the objective function guarantees that our solution is a sparse matrix, while Nesterov's method in [9] does not guarantee this since it solves the approximated dual problem.

The rest of the paper is organized as follows. In Section 2, we introduce our alternating direction method of multipliers for solving the nonsmooth SDP problems (1.4) and (1.5). The global convergence results of the alternating direction method of multipliers are given in Section 3. We discuss some practical issues including the deflation technique for computing multiple sparse PCs in Section 4. In Section 5, we use our alternating direction method of multipliers to solve sparse PCA problems arising from different applications such as classification of text data and senate voting records to demonstrate the efficacy of our method. We make some conclusions in Section 6.

2. Alternating Direction Method of Multipliers.

We first introduce some notation. We use $\mathcal{C}$ to denote the simplex of the cone of the semidefinite matrices, i.e., $\mathcal{C} = \{X \in \mathbb{R}^{p\times p} \mid \mathbf{Tr}(X) = 1, X \succeq 0\}$. We use $\mathcal{B}$ to denote the ℓ_1-ball with radius K in $\mathbb{R}^{p\times p}$, i.e., $\mathcal{B} = \{X \in \mathbb{R}^{p\times p} \mid \|X\|_1 \le K\}$. $I_{\mathcal{A}}(X)$ denotes the indicator function of set $\mathcal{A}$, i.e.,

$$I_{\mathcal{A}}(X) = \begin{cases} 0 & \text{if } X \in \mathcal{A}, \\ +\infty & \text{otherwise.} \end{cases} \tag{2.1}$$

We know that $I_{\mathcal{C}}(X)$ and $I_{\mathcal{B}}(X)$ are both convex functions since $\mathcal{C}$ and $\mathcal{B}$ are both convex sets. We then can reformulate (1.4) and (1.5) uniformly as the following unconstrained problem:

$$\min \quad -\langle \Sigma, X\rangle + I_{\mathcal{C}}(X) + h(X), \tag{2.2}$$

where $h(X) = I_{\mathcal{B}}(X)$ for (1.4) and $h(X) = \rho\|X\|_1$ for (1.5). Note that $h(X)$ is convex in both cases. (2.2) can be also viewed as the following inclusion problem:

$$\text{Find } X, \text{ s.t. } 0 \in -\Sigma + \partial I_{\mathcal{C}}(X) + \partial h(X). \tag{2.3}$$

Problem (2.3) finds zero of the sum of two monotone operators. Methods based on operator-splitting techniques, such as Douglas-Rachford method and Peachman-Rachford method, are usually used to solve Problem (2.3) (see e.g., [11, 30, 23, 13, 14, 6, 7]). From the convex optimization perspective, the alternating direction method of multipliers (ADMM) for solving (2.2) is a direct application of the Douglas-Rachford method. ADMM has been successfully used to solve structured convex optimization problems arising from image processing, compressed sensing, machine learning, semidefinite programming etc. (see e.g., [16, 15, 36, 38, 19, 39, 33, 17, 18, 26, 37, 2]). We now show how ADMM can be used to solve the sparse PCA problem (2.2).

ADMM is based on a variable-splitting technique and an augmented Lagrangian framework. By introducing a new variable Y, (2.2) can be rewritten as

$$\begin{array}{ll} \min & -\langle \Sigma, X\rangle + I_{\mathcal{C}}(X) + h(Y) \\ \text{s.t.} & X = Y. \end{array} \tag{2.4}$$

Note that although the number of variables is increased, the two nonsmooth functions $I_{\mathcal{C}}(\cdot)$ and $h(\cdot)$ are now separated since they are associated with different variables. For this equality-constrained problem, augmented Lagrangian method is a standard approach to solve it. A typical iteration of augmented Lagrangian method for solving (2.4) is given by:

$$\left\{\begin{array}{lll} (X^{k+1}, Y^{k+1}) & := & \arg\min\limits_{(X,Y)} \mathcal{L}_\mu(X, Y; \Lambda^k) \\ \Lambda^{k+1} & := & \Lambda^k - \frac{1}{\mu}(X^{k+1} - Y^{k+1}), \end{array}\right. \tag{2.5}$$

where the augmented Lagrangian function $\mathcal{L}_\mu(X, Y; \Lambda)$ is defined as:

$$\mathcal{L}_\mu(X, Y; \Lambda) := -\langle \Sigma, X\rangle + I_{\mathcal{C}}(X) + h(Y) - \langle \Lambda, X - Y\rangle + \frac{1}{2\mu}\|X - Y\|_F^2, \tag{2.6}$$

where $\mu > 0$ is a penalty parameter and Λ is the Lagrange multiplier associated with the linear constraint $X = Y$. Note that it is usually hard to minimize the augmented Lagrangian function $\mathcal{L}_\mu(X, Y; \Lambda^k)$ with respect to X and Y simultaneously. In fact, it is as difficult as solving the original problem (2.4). However, if we minimize the augmented Lagrangian function with respect to X and Y alternatingly, we obtain two subproblems in each iteration and both of them are relatively easy to solve. This results in the following alternating direction method of multipliers.

$$\left\{\begin{array}{lll} X^{k+1} & := & \arg\min_X \mathcal{L}_\mu(X, Y^k; \Lambda^k) \\ Y^{k+1} & := & \arg\min_Y \mathcal{L}_\mu(X^{k+1}, Y; \Lambda^k) \\ \Lambda^{k+1} & := & \Lambda^k - (X^{k+1} - Y^{k+1})/\mu, \end{array}\right. \tag{2.7}$$

It can be shown that the two subproblems in (2.7) are both relatively easy to solve in the sparse PCA problem. Before we do that, we characterize two nice properties of the indicator function (2.1).

- **Property 1.** The proximal mapping of the indicator function $I_{\mathcal{A}}(\cdot)$ is the Euclidean projection onto $\mathcal{A}$, i.e.,

  $$\operatorname{prox}_{I_{\mathcal{A}}}(X) \equiv \mathcal{P}_{\mathcal{A}}(X), \tag{2.8}$$

  where

  $$\operatorname{prox}_{I_{\mathcal{A}}}(X) := \arg\min_U \{I_{\mathcal{A}}(U) + \frac{1}{2}\|U - X\|_F^2\}, \tag{2.9}$$

and

$$\mathcal{P}_{\mathcal{A}}(X) := \arg\min_{U}\{\frac{1}{2}\|U - X\|_F^2, \text{ s.t. } U \in \mathcal{A}\}. \tag{2.10}$$

- **Property 2.** The optimality conditions for Problem (2.10) are given by

  $$X - U^* \in \partial I_{\mathcal{A}}(U^*), \tag{2.11}$$

  which is equivalent to

  $$\langle X - U^*, Z - U^* \rangle \leq 0, \forall Z \in \mathcal{A}, \tag{2.12}$$

  where U^* is the optimal solution of (2.10).

Now, the first subproblem in (2.7) can be reduced to:

$$X^{k+1} := \arg\min\left\{\mu I_{\mathcal{C}}(X) + \frac{1}{2}\|X - (Y^k + \mu\Lambda^k + \mu\Sigma)\|_F^2\right\}, \tag{2.13}$$

which can be further reduced to projection onto $\mathcal{C}$ using Property 1,

$$X^{k+1} = \mathcal{P}_{\mathcal{C}}(Y^k + \mu\Lambda^k + \mu\Sigma) := \arg\min\left\{\frac{1}{2}\|X - (Y^k + \mu\Lambda^k + \mu\Sigma)\|_F^2, \text{ s.t. } X \in \mathcal{C}\right\}. \tag{2.14}$$

When $h(Y) = I_{\mathcal{B}}(Y)$ as in Problem (1.4), the second subproblem in (2.7) can be reduced to:

$$Y^{k+1} := \arg\min\left\{\mu I_{\mathcal{B}}(Y) + \frac{1}{2}\|Y - (X^{k+1} - \mu\Lambda^k)\|_F^2\right\}, \tag{2.15}$$

which can be further reduced to projection onto $\mathcal{B}$ using Property 1,

$$Y^{k+1} = \mathcal{P}_{\mathcal{B}}(X^{k+1} - \mu\Lambda^k) := \arg\min\left\{\frac{1}{2}\|Y - (X^{k+1} - \mu\Lambda^k)\|_F^2, \text{ s.t. } Y \in \mathcal{B}\right\}. \tag{2.16}$$

When $h(Y) = \rho\|Y\|_1$ as in Problem (1.5), the second subproblem in (2.7) can be reduced to:

$$Y^{k+1} := \arg\min\left\{\mu\rho\|Y\|_1 + \frac{1}{2}\|Y - (X^{k+1} - \mu\Lambda^k)\|_F^2\right\}. \tag{2.17}$$

Problem (2.17) has a closed-form solution that is given by

$$Y^{k+1} = \text{Shrink}(X^{k+1} - \mu\Lambda^k, \mu\rho), \tag{2.18}$$

where the shrinkage operator is defined as:

$$(\text{Shrink}(Z, \tau))_{ij} := \text{sgn}(Z_{ij}) \max\{|Z_{ij}| - \tau, 0\}, \forall i, j. \tag{2.19}$$

In the following, we will show that (2.13) and (2.15) are easy to solve, i.e., the two projections (2.14) and (2.16) can be done efficiently. First, since the problem of projection onto $\mathcal{C}$

$$\mathcal{P}_{\mathcal{C}}(X) = \arg\min\{\frac{1}{2}\|Z - X\|_F^2, \text{ s.t. } \mathbf{Tr}(Z) = 1, Z \succeq 0\} \tag{2.20}$$

is unitary-invariant, its solution is given by $\mathcal{P}_{\mathcal{C}}(X) = U\,\mathrm{diag}(\gamma)U^\top$, where $X = U\,\mathrm{diag}(\sigma)U^\top$ is the eigenvalue decomposition of X, and γ is the projection of σ onto the simplex in the Euclidean space, i.e.,

$$\gamma := \arg\min\{\frac{1}{2}\|\xi - \sigma\|_2^2, \text{ s.t. } \sum_{i=1}^{p} \xi_i = 1, \xi \geq 0\}. \tag{2.21}$$

We consider a slightly more general problem

$$\xi^* := \arg\min\{\frac{1}{2}\|\xi - \sigma\|_2^2, \text{ s.t. } \sum_{i=1}^{p} \xi_i = r, \xi \geq 0\}, \tag{2.22}$$

where scalar $r > 0$. Note that (2.21) is a special case of (2.22) with $r = 1$. From the first-order optimality conditions for (2.22), it is easy to show that the optimal solution of (2.22) is given by

$$\xi_i^* := \max\{\sigma_i - \theta, 0\}, \forall i = 1, \ldots, p,$$

where the scalar θ is the solution of the following piecewise linear equation:

$$\sum_{i=1}^{p} \max\{\sigma_i - \theta, 0\} = r. \tag{2.23}$$

It is known that the piecewise linear equation (2.23) can be solved quite efficiently and thus solving (2.22) can be done easily. In fact, the following procedure (Algorithm 1) gives the optimal solution of (2.22). We refer the readers to [31] for the proof of the validity of the algorithm. It is easy to see that Algorithm 1 has

Algorithm 1: Projection onto the simplex in the Euclidean space

```
Input: A vector $\sigma \in \mathbb{R}^p$ and a scalar $r > 0$.
Sort $\sigma$ into $\hat{\sigma}$ as a non-decreasing order: $\hat{\sigma}_1 \leq \hat{\sigma}_2 \leq \ldots \leq \hat{\sigma}_p$
Find index $\hat{j}$, the smallest $j$ such that $\hat{\sigma}_j - \frac{1}{p-j+1}\left(\sum_{i=j}^{p} \hat{\sigma}_i - r\right) > 0$
Compute $\theta = \frac{1}{p-\hat{j}+1}\left(\sum_{i=\hat{j}}^{p} \hat{\sigma}_i - r\right)$
Output: A vector $\gamma$, s.t. $\gamma_i = \max\{\sigma_i - \theta, 0\}, i = 1, \ldots, p$.
```

an $O(p \log p)$ complexity. Linear time algorithms for solving (2.22) are studied in [4, 29, 12]. Thus, solving (2.13) corresponds to an eigenvalue decomposition and a projection onto the simplex in the Euclidean space, and they both can be done efficiently.

Solving (2.15) (or equivalently (2.16)) corresponds to a projection onto the ℓ_1-ball: $\|Y\|_1 \leq K$. It has been shown in [12, 35] that projection onto the ℓ_1-ball can be done easily. In fact, the solution of

$$\hat{\gamma} = \arg\min\{\frac{1}{2}\|\xi - \hat{\sigma}\|_2^2, \text{ s.t. } \|\xi\|_1 \leq r\} \tag{2.24}$$

is given by $\hat{\gamma}_i = \mathrm{sgn}(\hat{\sigma}_i)\gamma_i, \forall i = 1, \ldots, p$, where γ is the solution of

$$\min \quad \frac{1}{2}\|\gamma - |\hat{\sigma}|\|_2^2, \quad \text{s.t. } \sum_{i=1}^{p} \gamma_i = r, \gamma \geq 0,$$

i.e., the projection of $|\hat{\sigma}|$ (elementwise absolute value of $\hat{\sigma}$) onto the simplex. Thus, (2.15) can be rewritten as

$$\operatorname{vec}(Y^{k+1}) = \arg\min\{\frac{1}{2}\|y - \operatorname{vec}(X^{k+1} - \mu\Lambda^k)\|_2^2, \text{ s.t. } \|y\|_1 \leq K\}, \tag{2.25}$$

and it corresponds to a projection onto the simplex in the Euclidean space, where $\operatorname{vec}(Y)$ denotes the vector form of Y which is obtained by stacking the columns of Y into a long vector.

To summarize, our ADMM for solving (1.4) and (1.5) can be uniformly described as Algorithm 2.

Algorithm 2: ADMM for solving (1.4) and (1.5)

```
Initialization: $Y^0 = 0$, $\Lambda^0 = 0$.
for $k=0,1,\ldots$ do
    Compute the eigenvalue decomposition: $Y^k + \mu\Lambda^k + \mu\Sigma = U\,\mathrm{diag}(\sigma)U^\top$
    Project $\sigma$ onto the simplex in Euclidean space by Algorithm 1, and denote the solution by $\gamma$
    Compute $X^{k+1} = U\,\mathrm{diag}(\gamma)U^\top$
    Perform one of the followings:
        • if (1.4) is solved, update $Y^{k+1}$ by solving (2.25)
        • if (1.5) is solved, update $Y^{k+1}$ by (2.18)
    Update $\Lambda^{k+1}$ by $\Lambda^{k+1} = \Lambda^k - (X^{k+1} - Y^{k+1})/\mu$
```

REMARK 2.1. *Although Algorithm 2 suggests that we need to compute the eigenvalue decomposition of $Y^k + \mu\Lambda^k + \mu\Sigma$ in order to get the solution to* (2.13)*, we actually only need to compute the positive eigenvalues and corresponding eigenvectors of $Y^k + \mu\Lambda^k + \mu\Sigma$.*

3. Global Convergence Results.

In this section, we prove that the sequence (X^k, Y^k, Λ^k) produced by the alternating direction method of multipliers (2.7) (i.e., Algorithm 2) converges to (X^*, Y^*, Λ^*), where (X^*, Y^*) is an optimal solution to (2.4) and Λ^* is the corresponding optimal dual variable. Although the proof of global convergence results of ADMM has been studied extensively in the literature (see e.g., [14, 20]), we here give a very simple proof of the convergence of our ADMM that utilizes the special structures of the sparse PCA problem. We only prove the case when $h(Y) = I_{\mathcal{B}}(Y)$ and leave the case when $h(Y) = \rho\|Y\|_1$ to the readers since their proofs are almost identical.

Before we give the main theorem about the global convergence of (2.7) (Algorithm 2), we need the following lemma.

LEMMA 3.1. *Assume that (X^*, Y^*) is an optimal solution of* (2.4) *and Λ^* is the corresponding optimal dual variable associated with the equality constraint $X = Y$. Then the sequence (X^k, Y^k, Λ^k) produced by* (2.7) *satisfies*

$$\|U^k - U^*\|_G^2 - \|U^{k+1} - U^*\|_G^2 \geq \|U^k - U^{k+1}\|_G^2, \tag{3.1}$$

where $U^* = \begin{pmatrix} \Lambda^* \\ Y^* \end{pmatrix}$, $U^k = \begin{pmatrix} \Lambda^k \\ Y^k \end{pmatrix}$ *and* $G = \begin{pmatrix} \mu I & 0 \\ 0 & \frac{1}{\mu} I \end{pmatrix}$, *and the norm* $\|\cdot\|_G^2$ *is defined as* $\|U\|_G^2 = \langle U, GU\rangle$ *and the corresponding inner product* $\langle\cdot,\cdot\rangle_G$ *is defined as* $\langle U, V\rangle_G = \langle U, GV\rangle$.

Proof. Since (X^*, Y^*, Λ^*) is optimal to (2.4), it follows from the KKT conditions that the followings hold:

$$0 \in -\Sigma + \partial I_{\mathcal{C}}(X^*) - \Lambda^*, \tag{3.2}$$

$$0 \in \partial I_{\mathcal{B}}(Y^*) + \Lambda^*, \tag{3.3}$$

and

$$X^* = Y^* \in \mathcal{C} \cap \mathcal{B}. \tag{3.4}$$

By using Property 2, (3.2) and (3.3) can be respectively reduced to:

$$\langle \Sigma + \Lambda^*, X - X^* \rangle \leq 0, \forall X \in \mathcal{C}, \tag{3.5}$$

and

$$\langle -\Lambda^*, Y - Y^* \rangle \leq 0, \forall Y \in \mathcal{B}. \tag{3.6}$$

Note that the optimality conditions for the first subproblem (i.e., the subproblem with respect to X) in (2.7) are given by $X^{k+1} \in \mathcal{C}$ and

$$0 \in -\Sigma + \partial I_{\mathcal{C}}(X^{k+1}) - \Lambda^k + \frac{1}{\mu}(X^{k+1} - Y^k). \tag{3.7}$$

By using Property 2 and the updating formula for Λ^k in (2.7), i.e.,

$$\Lambda^{k+1} = \Lambda^k - \frac{1}{\mu}(X^{k+1} - Y^{k+1}), \tag{3.8}$$

(3.7) can be rewritten as

$$\langle \Sigma + \Lambda^{k+1} + \frac{1}{\mu}(Y^k - Y^{k+1}), X - X^{k+1} \rangle \leq 0, \forall X \in \mathcal{C}. \tag{3.9}$$

Letting $X = X^{k+1}$ in (3.5) and $X = X^*$ in (3.9), and summing the two resulting inequalities, we get,

$$\langle \Lambda^{k+1} - \Lambda^* + \frac{1}{\mu}(Y^k - Y^{k+1}), X^* - X^{k+1} \rangle \leq 0. \tag{3.10}$$

The optimality conditions for the second subproblem (i.e., the subproblem with respect to Y) in (2.7) are given by $Y^{k+1} \in \mathcal{B}$ and

$$0 \in \partial I_{\mathcal{B}}(Y^{k+1}) + \Lambda^k + \frac{1}{\mu}(Y^{k+1} - X^{k+1}). \tag{3.11}$$

By using Property 2 and (3.8), (3.11) can be rewritten as

$$\langle -\Lambda^{k+1}, Y - Y^{k+1} \rangle \leq 0, \forall Y \in \mathcal{B}. \tag{3.12}$$

Letting $Y = Y^{k+1}$ in (3.6) and $Y = Y^*$ in (3.12), and summing the two resulting inequalities, we obtain,

$$\langle \Lambda^* - \Lambda^{k+1}, Y^* - Y^{k+1} \rangle \leq 0. \tag{3.13}$$

Summing (3.10) and (3.13), and using the facts that $X^* = Y^*$ and $X^{k+1} = \mu(\Lambda^k - \Lambda^{k+1}) + Y^{k+1}$, we obtain,

$$\mu\langle \Lambda^k - \Lambda^{k+1}, \Lambda^{k+1} - \Lambda^*\rangle + \frac{1}{\mu}\langle Y^k - Y^{k+1}, Y^{k+1} - Y^*\rangle \geq -\langle Y^k - Y^{k+1}, \Lambda^k - \Lambda^{k+1}\rangle. \tag{3.14}$$

Rearranging the left hand side of (3.14) by using $\Lambda^{k+1} - \Lambda^* = (\Lambda^{k+1} - \Lambda^k) + (\Lambda^k - \Lambda^*)$ and $Y^{k+1} - Y^* = (Y^{k+1} - Y^k) + (Y^k - Y^*)$, we get

$$\mu\langle \Lambda^k - \Lambda^*, \Lambda^k - \Lambda^{k+1}\rangle + \frac{1}{\mu}\langle Y^k - Y^*, Y^k - Y^{k+1}\rangle \geq \mu\|\Lambda^k - \Lambda^{k+1}\|^2 + \frac{1}{\mu}\|Y^k - Y^{k+1}\|^2 - \langle \Lambda^{k+1} - \Lambda^k, Y^{k+1} - Y^k\rangle. \tag{3.15}$$

Using the notation of U^k, U^* and G, (3.15) can be rewritten as

$$\langle U^k - U^*, U^k - U^{k+1}\rangle_G \geq \|U^k - U^{k+1}\|_G^2 - \langle \Lambda^k - \Lambda^{k+1}, Y^k - Y^{k+1}\rangle. \tag{3.16}$$

Combining (3.16) with the identity

$$\|U^{k+1} - U^*\|_G^2 = \|U^{k+1} - U^k\|_G^2 - 2\langle U^k - U^{k+1}, U^k - U^*\rangle_G + \|U^k - U^*\|_G^2,$$

we get

$$\begin{aligned} & \|U^k - U^*\|_G^2 - \|U^{k+1} - U^*\|_G^2 \\ = \; & 2\langle U^k - U^{k+1}, U^k - U^*\rangle - \|U^{k+1} - U^k\|_G^2 \\ \geq \; & 2\|U^k - U^{k+1}\|_G^2 - 2\langle \Lambda^k - \Lambda^{k+1}, Y^k - Y^{k+1}\rangle - \|U^{k+1} - U^k\|_G^2 \\ = \; & \|U^k - U^{k+1}\|_G^2 - 2\langle \Lambda^k - \Lambda^{k+1}, Y^k - Y^{k+1}\rangle. \end{aligned} \tag{3.17}$$

Now, using (3.12) for k instead of $k+1$ and letting $Y = Y^{k+1}$, we get,

$$\langle -\Lambda^k, Y^{k+1} - Y^k\rangle \leq 0. \tag{3.18}$$

Letting $Y = Y^k$ in (3.12) and adding it to (3.18) yields,

$$\langle \Lambda^k - \Lambda^{k+1}, Y^k - Y^{k+1}\rangle \leq 0. \tag{3.19}$$

By substituting (3.19) into (3.17) we get the desired result (3.1). $\square$

We are now ready to give the main convergence result of (2.7) (Algorithm 2).

THEOREM 3.2. *The sequence* $\{(X^k, Y^k, \Lambda^k)\}$ *produced by* (2.7) *(Algorithm 2) from any starting point converges to an optimal solution to Problem* (2.4).

Proof. From Lemma 3.1 we can easily get that

- (i) $\|U^k - U^{k+1}\|_G \to 0$;
- (ii) $\{U^k\}$ lies in a compact region;
- (iii) $\|U^k - U^*\|_G^2$ is monotonically non-increasing and thus converges.

It follows from (i) that $\Lambda^k - \Lambda^{k+1} \to 0$ and $Y^k - Y^{k+1} \to 0$. Then (3.8) implies that $X^k - X^{k+1} \to 0$ and $X^k - Y^k \to 0$. From (ii) we obtain that, U^k has a subsequence $\{U^{k_j}\}$ that converges to $\hat{U} = (\hat{\Lambda}, \hat{Y})$, i.e., $\Lambda^{k_j} \to \hat{\Lambda}$ and $Y^{k_j} \to \hat{Y}$. From $X^k - Y^k \to 0$ we also get that $X^{k_j} \to \hat{X} := \hat{Y}$. Therefore, $(\hat{X}, \hat{Y}, \hat{\Lambda})$ is a limit point of $\{(X^k, Y^k, \Lambda^k)\}$.

Note that by using (3.8), (3.7) can be rewritten as

$$0 \in -\Sigma + \partial I_{\mathcal{C}}(X^{k+1}) - \Lambda^{k+1} + \frac{1}{\mu}(Y^{k+1} - Y^k), \tag{3.20}$$

which implies that

$$0 \in -\Sigma + \partial I_{\mathcal{C}}(\hat{X}) - \hat{\Lambda}. \tag{3.21}$$

Note also that (3.11) implies that

$$0 \in \partial I_{\mathcal{B}}(\hat{Y}) + \hat{\Lambda}. \tag{3.22}$$

Moreover, it follows from $X^k \in \mathcal{C}$ and $Y^k \in \mathcal{B}$ that

$$\hat{X} \in \mathcal{C} \text{ and } \hat{Y} \in \mathcal{B}. \tag{3.23}$$

(3.21), (3.22), (3.23) together with $\hat{X} = \hat{Y}$ imply that $(\hat{X}, \hat{Y}, \hat{\Lambda})$ satisfies the KKT conditions for (2.4) and thus is an optimal solution to (2.4). Therefore, we showed that any limit point of $\{(X^k, Y^k, \Lambda^k)\}$ is an optimal solution to (2.4). □

4. The Deflation Techniques and Other Practical Issues.

It should be noticed that the solution of Problem (1.1) only gives the largest eigenvector (the eigenvector corresponding to the largest eigenvalue) of Σ. In many applications, the largest eigenvector is not enough to explain the total variance of the data. Thus one usually needs to compute several leading eigenvectors to explain more variance of the data. Hotelling's deflation method [32] is usually used to extract the leading eigenvectors sequentially. The Hotelling's deflation method extracts the r-th leading eigenvector of Σ by solving

$$x_r = \arg\max\{x^\top \Sigma_{r-1} x, \text{ s.t. } \|x\|_2 \leq 1\},$$

where $\Sigma_0 := \Sigma$ and

$$\Sigma_r = \Sigma_{r-1} - x_r x_r^\top \Sigma_{r-1} x_r x_r^\top.$$

It is easy to verify that Hotelling's deflation method preserves the positive-semidefiniteness of matrix Σ_r. However, as pointed out in [25], it does not preserve the positive-semidefiniteness of Σ_r when it comes to the sparse PCA problem (1.2), because the solution x_r is no longer an eigenvector of Σ_{r-1}. Thus, the second leading eigenvector produced by solving the sparse PCA problem may not explain well the variance of the data. We should point out that the deflation method used in [9] is the Hotelling's deflation method.

Several deflation techniques to overcome this difficulty for sparse PCA were proposed by Mackey in [25]. In our numerical experiments, we chose to use the Schur complement deflation method in [25]. The Schur complement deflation method updates matrix Σ_r by

$$\Sigma_r = \Sigma_{r-1} - \frac{\Sigma_{r-1} x_r x_r^\top \Sigma_{r-1}}{x_r^\top \Sigma_{r-1} x_r}. \tag{4.1}$$

The Schur complement deflation method has the following properties as shown in [25]. (i) Schur complement deflation preserves the positive-semidefiniteness of Σ_r, i.e., $\Sigma_r \succeq 0$. (ii) Schur complement deflation renders

x_s orthogonal to Σ_r for $s \leq r$, i.e., $\Sigma_r x_s = 0, \forall s \leq r$.

When we want to find the leading r sparse PCs of Σ, we use ADMM to solve sequentially r problems (1.4) or (1.5) with Σ updated by the Schur complement deflation method (4.1). We denote the leading r sparse PCs obtained by our ADMM as $X_r = (x_1, \ldots, x_r)$. Usually the total variance explained by X_r is given by $\mathbf{Tr}(X_r^\top \Sigma X_r)$. However, because we do not require $x_1, \ldots, x_r$ to be orthogonal to each other when we sequentially solve the SDPs (1.4) or (1.5), these loadings are correlated. Thus, $\mathbf{Tr}(X_r^\top \Sigma X_r)$ will overestimate the total explained variance by $x_1, \ldots, x_r$. To alleviate the overestimated variance, Zou et al. [42] suggested that the explained total variance should be computed using the following procedure, which was called *adjusted variance*:

$$AdjVar(X_r) := \mathbf{Tr}(R^2),$$

where $X_r = QR$ is the QR decomposition of X_r. In our numerical experiments, we always report the adjusted variance as the explained variance.

It is also worth noticing that the problems we solve are convex relaxations of the original problems (1.2) and (1.9). Hence, one needs to postprocess the matrix X obtained by solving (1.4) or (1.5) to get the solution to (1.2) or (1.9). To get the solution to the original sparse PCA problem (1.2) or (1.9) from the solution X of the convex SDP problem, we simply perform a rank-one decomposition to X, i.e., $X = xx^\top$. Since X is a sparse matrix, x should be a sparse vector. This postprocessing technique is also used in [9].

Since the sequences $\{X^k\}$ and $\{Y^k\}$ generated by ADMM converge to the same point eventually, we terminate ADMM when the difference between X^k and Y^k is sufficiently small. In our numerical experiments, we terminate ADMM when

$$\frac{\|X^k - Y^k\|_F}{\max\{1, \|X^k\|_F, \|Y^k\|_F\}} < 10^{-4}.$$

5. Numerical Results. In this section, we use our ADMM to solve the SDP formulations (1.4) and (1.5) of sparse PCA on both synthetic and real data sets. We compare the performance of ADMM with two methods for solving sparse PCA. One method is DSPCA [9] for solving (1.5) and the other method is ALSPCA [24] for solving (1.11). The Matlab codes of DSPCA and ALSPCA were downloaded from the authors' websites. Note that the main parts of the DSPCA codes were actually written in C-Mex files. Our codes were written in Matlab. All experiments were run in MATLAB 7.3.0 on a laptop with 1.66GHZ CPU and 2GB of RAM.

5.1. A synthetic example. We tested our ADMM on a synthetic data set suggested by Zou et al. in [42]. This synthetic example has three hidden factors:

$$V_1 \sim \mathcal{N}(0, 290), V_2 \sim \mathcal{N}(0, 300), V_3 = -0.3V_1 + 0.925V_2 + \epsilon,$$

where $\epsilon \sim \mathcal{N}(0, 1)$, and V_1, V_2 and ϵ are independent. The 10 observable variables are given by the following procedure:

$$\begin{aligned}
X_i &= V_1 + \epsilon_i^1, \quad \epsilon_i^1 \sim \mathcal{N}(0,1), \quad i = 1,2,3,4, \\
X_i &= V_2 + \epsilon_i^2, \quad \epsilon_i^2 \sim \mathcal{N}(0,1), \quad i = 5,6,7,8, \\
X_i &= V_3 + \epsilon_i^3, \quad \epsilon_i^3 \sim \mathcal{N}(0,1), \quad i = 9,10,
\end{aligned}$$

where ϵ_i^j are independent for $j = 1, 2, 3$ and $i = 1, \ldots, 10$. The exact covariance matrix of $(X_1, \ldots, X_{10})$ is used to find the standard PCs by standard PCA and sparse PCs by ADMM. Note that the variances of V_1, V_2 and V_3 indicate that V_2 is slightly more important than V_1 and they are both more important than V_3. Also, we note that the first two PCs explain more than 99% of the total variance. Thus, using the first two PCs should be able to explain most of the variance, and the first sparse PC explains most of the variance of V_2 using (X_5, X_6, X_7, X_8) and the second sparse PC explains most of the variance of V_1 using (X_1, X_2, X_3, X_4). Based on these observations, we set $K = 4$ in (1.4) for computing both the first and the second sparse PCs. When we computed the second sparse PC, we used the Schur complement deflation method described in Section 4 to construct the corresponding sample covariance matrix. The penalty parameter μ in ADMM was set to 0.8. The PCs given by the standard PCA and sparse PCA using our ADMM for solving (1.5) and the explained variances are shown in Table 5.1. From Table 5.1 we see that ADMM gives sparse loadings using (X_5, X_6, X_7, X_8) in the first PC and (X_1, X_2, X_3, X_4) in the second PC. The first two sparse PCs explain 80.41% of the total variance.

Table 5.1
Loadings and explained variance for the first two PCs

	Standard PCA		ADMM	
Variables	PC1	PC2	PC1	PC2
X_1	-0.1157	-0.4785	0	0.5000
X_2	-0.1157	-0.4785	0	0.5000
X_3	-0.1157	-0.4785	0	0.5000
X_4	-0.1157	-0.4785	0	0.5000
X_5	0.3953	-0.1449	0.5000	0
X_6	0.3953	-0.1449	0.5000	0
X_7	0.3953	-0.1449	0.5000	0
X_8	0.3953	-0.1449	0.5000	0
X_9	0.4008	0.0095	0	0
X_{10}	0.4008	0.0095	0	0
Total explained variance	99.68%		80.41%	

5.2. Pit props data. The pit props data set has been a standard benchmark for testing sparse PCA algorithms since it was introduced by Jeffers in [21]. The pit props data set has 180 observations and 13 measured variables. Thus the covariance matrix Σ is a 13×13 matrix. We used our ADMM to compute the first six sparse PCs sequentially via the Schur complement deflation technique discussed in Section 4. We set $K = (6, 2, 2, 1, 1, 1)$ for the six problems (1.4) as suggested in [9]. We set $\mu = 0.8$ in ADMM. The first six sparse PCs obtained by ADMM are shown in Table 5.2. We compared the results with ALSPCA for solving (1.11). For ALSPCA, we used the parameters as suggested by the authors, i.e., $r = 6, \rho = 0.70, \Delta_{ij} = 0.50, \forall i \neq j$. The results given by ALSPCA are reported in Table 5.3. Since there was no clue how to choose the six parameters ρ in the six problems (1.5) when DSPCA is used to solve them, we did not compare with DSPCA for solving (1.5). From Tables 5.2 and 5.3 we see that both ADMM and ALSPCA gave a solution with 15 nonzeros in the first six sparse PCs, and the solution given by ADMM explains slightly more variance than the solution given by ALSPCA.

5.3. Random examples. We created some random examples to test the speed of ADMM and compared it with DSPCA [9] and ALSPCA [24]. The sample covariance matrix Σ was created by adding some small noise to a sparse rank-one matrix. Specifically, we first created a sparse vector $\hat{x} \in \mathbb{R}^p$ with s nonzeros randomly chosen from the Gaussian distribution $\mathcal{N}(0, 1)$. We then got the sample covariance

Table 5.2
First six sparse PCs of the pit props data set given by ADMM

Variables	PC1	PC2	PC3	PC4	PC5	PC6
Topdiam	-0.4908	0	0	0	0	0
Length	-0.5067	0	0	0	0	0
Moist	0	-0.7175	0	0	0	0
Testsg	0	-0.6965	0	0	0	0
Ovensg	0	0	0.9263	0	0	0
Ringtop	-0.0668	0	0.3511	0	0	0
Ringbut	-0.3565	0	0.1369	0	0	0
Bowmax	-0.2334	0	0	0	0	0
Bowdist	-0.3861	0	0	0	0	0
Whorls	-0.4089	0	0	0	0	0
Clear	0	0	0	1.0000	0	0
Knots	0	0	0	0	1.0000	0
Diaknot	0	0	0	0	0	1.0000
Total sparsity: 15, total explained variance: 74.31%						

Table 5.3
First six sparse PCs of the pit props data set given by ALSPCA

Variables	PC1	PC2	PC3	PC4	PC5	PC6
Topdiam	0.4052	0	0	0	0	0
Length	0.4248	0	0	0	0	0
Moist	0	-0.7262	0	0	0	0
Testsg	0.0014	-0.6874	0	0	0	0
Ovensg	0	0	-1.0000	0	0	0
Ringtop	0.1857	0	0	0	0	0
Ringbut	0.4122	0	0	0	0	0
Bowmax	0.3277	0	0	0	0	0
Bowdist	0.3829	0	0	0	0	0
Whorls	0.4437	0.0021	0	0	0	0
Clear	0	0	0	1.0000	0	0
Knots	0	0	0	0	1.0000	0
Diaknot	0	0	0	0	0	1.0000
Total sparsity: 15, total explained variance: 73.29%						

matrix $\Sigma = \hat{x}\hat{x}^\top + \sigma vv^\top$, where σ denotes the noise level and $v \in \mathbb{R}^p$ is a random vector with entries uniformly drawn from $[0,1]$. We applied DSPCA, ALSPCA and ADMM to find the largest sparse PC of Σ. We report the comparison results in Tables 5.4 and 5.5 that correspond to noise levels $\sigma = 0.01$ and $\sigma = 0.1$ respectively. When using DSPCA to solve (1.5) and ALSPCA to solve (1.11), we set different ρ's to get solutions with different sparsity levels. Specifically, we tested DSPCA for $\rho = 0.01, 0.1, 1$ in Table 5.4 with $\sigma = 0.01$ and $\rho = 0.1, 1, 10$ in Table 5.5 with $\sigma = 0.1$; we tested ALSPCA for $\rho = 0.01, 0.1, 1$ in both Tables 5.4 and 5.5. We set different K's in (1.4) to control the sparsity level when using ADMM to solve it. In both Tables 5.4 and 5.5, we tested four data sets with dimension p and sparsity s setting as $(p,s) = (100,10), (100,20), (200,10)$ and $(200,20)$. We used the following continuation technique for μ in ADMM: $\mu_0 = 1, \mu_k = \max\{2\mu_{k-1}/3, 10^{-4}\}$. Δ_{ij} were set to 0.1 for all $i \neq j$ in all the tests for ALSPCA as suggested in [24] for tests on random data sets.

We report the cardinality of the largest sparse PC (Card), the percentage of the explained variance (PEV) and the CPU time in Tables 5.4 and 5.5. From Table 5.4 we see that, for $\sigma = 0.01$, all three algorithms DSPCA, ADMM and ALSPCA are sensitive to the parameters that control the sparsity, i.e., ρ and K. $\rho = 0.01$ always gave the best results for DSPCA and ALSPCA and the explained variance is very close to the standard PCA. $\rho = 0.1$ still provided relatively good solutions for DSPCA and ALSPCA in terms of both sparsity and the explained variance. When ρ was increased to 1, the solutions given by ALSPCA were too sparse to give a relatively large explained variance; while the solutions given by DSPCA sometimes had more nonzeros than the desired sparsity level (when $(p, s) = (100, 10)$), and even when the solutions were of the desired sparsity level, the explained variances were affected a lot (when $(p, s) = (100, 20)$ and $(200, 20)$). For ADMM, $K = 5, 4, 3$ were tested for $s = 10$ and $K = 10, 9, 8$ were tested for $s = 20$. Results shown in Table 5.4 indicate that $K = s/2$ usually produced good results. When K was changed from 5 to 4 and 3 for $s = 10$, the sparsity and explained variance of the solution changed a lot. When K was changed from 10 to 9 and 8 for $s = 20$, the solution was not affected too much in terms of the explained variance. Especially, for $(p, s) = (100, 20)$ and $(200, 20)$ and $K = 8$, ADMM gave solutions with sparsity 13 that explain 96.10% and 93.66% variance respectively, which are both very close to the results given by the standard PCA. From Table 5.5 we see that, for $\sigma = 0.1$, i.e., when the noise level was large, DSPCA and ALSPCA were more sensitive to the noise compared with their performance when $\sigma = 0.01$. More specifically, in Table 5.5, $\rho = 0.1$ usually gave a solution with the best explained variance and appropriate sparsity for DSPCA, expect for $(p, s) = (100, 10)$, where the solution produced by DSPCA had 21 nonzeros, which was much more than the desired sparsity 10. The solutions given by DSPCA for $\rho = 1$ and $\rho = 10$ were not very satisfied. For ALSPCA, when $(p, s) = (100, 10)$ and $(100, 20)$, $\rho = 0.1$ gave good results, while the results given by $\rho = 0.01$ and $\rho = 1$ were not very satisfied. However, we observed that the performance of ADMM when $\sigma = 0.1$ was consistent with its performance when $\sigma = 0.01$, i.e., its performance was not very sensitive to the noise.

From both Tables 5.4 and 5.5, we see that ALSPCA was slightly faster than ADMM, and they were both significantly faster than DSPCA. This is reasonable because ALSPCA solves the non-convex problem (1.11) and thus eigenvalue decomposition is not required, which costs most of the computational effort in DSPCA and ADMM.

5.4. Text data classification. Sparse PCA can also be used to classify the keywords in text data. This application has been studied by Zhang, d'Aspremont and El Ghaoui in [40] and Zhang and El Ghaoui in [41]. In this section, we show that by using our ADMM to solve the sparse PCA problem, we can also classify the keywords from text data very well. The data set we used is a small version of the "20-newsgroups" data[1], which is also used in [40]. This data set consists of the binary occurrences of 100 specific words across 16242 postings, i.e., the data matrix M is of the size 100×16242 and $M_{ij} = 1$ if the i-th word appears at least once in the j-th posting and $M_{ij} = 0$ if the i-th word does not appear in the j-th posting. These words can be approximately divided into different groups such as "computer", "religion" etc. We want to find the words that contribute as much variance as possible and also discover which words are in the same category. By viewing each posting as a sample of the 100 variables, we have 16424 samples of the variables, and thus the sample covariance matrix $\Sigma \in \mathbb{R}^{100\times 100}$. Using standard PCA, it is hard to interpret which words contribute to each of the leading eigenvalues since the loadings are dense. However, sparse PCA can explain as much the variance explained by the standard PCs, and meanwhile interpret well which words contribute together to the corresponding variance. We applied our ADMM to solve (1.4) to find the first three sparse PCs. We set $K = 5$ in all three problems and the following continuation technique was used for μ: $\mu_0 = 100, \mu_k = \max\{2\mu_{k-1}/3, 10^{-4}\}$. The resulting three sparse PCs have 10, 12 and 17 nonzeros

[1] This data set can be downloaded from http://cs.nyu.edu/∼roweis/data.html.

Table 5.4

Comparisons of ADMM, ALSPCA and DSPCA on random examples with $\sigma = 0.01$

(p, s)	Method	Parameters	Card	PEV	CPU
(100,10)	PCA			96.16%	
	DSPCA	$\rho = 0.01$	10	96.16%	12.12
		$\rho = 0.1$	10	95.81%	8.29
		$\rho = 1$	13	87.28%	6.56
	ADMM	$K = 5$	9	95.30%	0.26
		$K = 4$	6	91.55%	0.28
		$K = 3$	4	79.30%	0.19
	ALSPCA	$\rho = 0.01$	10	96.16%	0.13
		$\rho = 0.1$	9	96.02%	0.39
		$\rho = 1$	4	89.54%	0.13
(100,20)	PCA			98.07%	
	DSPCA	$\rho = 0.01$	20	98.07%	10.93
		$\rho = 0.1$	20	97.71%	8.47
		$\rho = 1$	20	85.25%	5.75
	ADMM	$K = 10$	20	97.98%	0.28
		$K = 9$	18	97.40%	0.28
		$K = 8$	13	96.10%	0.31
	ALSPCA	$\rho = 0.01$	20	98.07%	0.13
		$\rho = 0.1$	18	97.83%	0.13
		$\rho = 1$	8	83.87%	0.13
(200,10)	PCA			91.43%	
	DSPCA	$\rho = 0.01$	10	91.42%	88.87
		$\rho = 0.1$	10	91.09%	61.36
		$\rho = 1$	8	82.91%	45.97
	ADMM	$K = 5$	9	90.61%	1.23
		$K = 4$	6	87.04%	1.27
		$K = 3$	4	75.40%	0.88
	ALSPCA	$\rho = 0.01$	10	91.42%	0.23
		$\rho = 0.1$	9	91.29%	0.23
		$\rho = 1$	4	85.13%	0.29
(200,20)	PCA			95.58%	
	DSPCA	$\rho = 0.01$	20	95.58%	79.87
		$\rho = 0.1$	20	95.22%	63.12
		$\rho = 1$	20	83.09%	42.22
	ADMM	$K = 10$	20	95.49%	1.57
		$K = 9$	18	94.93%	1.67
		$K = 8$	13	93.66%	1.71
	ALSPCA	$\rho = 0.01$	20	95.58%	0.23
		$\rho = 0.1$	18	95.34%	0.23
		$\rho = 1$	8	81.73%	0.23

respectively. The total explained variance by these three sparse PCs is 12.72%, while the variance explained by the largest three PCs by the standard PCA is 19.10%.

The words corresponding to the first three sparse PCs generated by our ADMM are listed in Table 5.6. From Table 5.6 we see that the words in the first sparse PC are approximately in the category "school", the words in the second PC are approximately in the category "religion", and the words in the third sparse PC are approximately in the category "computer". So our ADMM can classify the keywords into appropriate categories very well.

Table 5.5
Comparisons of ADMM, ALSPCA and DSPCA on random examples with $\sigma = 0.1$

(p, s)	Method	Parameters	Card	PEV	CPU
(100,10)	PCA			71.51%	
	DSPCA	$\rho = 0.1$	21	71.23%	9.42
		$\rho = 1$	10	64.92%	4.40
		$\rho = 10$	1	27.04%	4.14
	ADMM	$K = 5$	9	70.83%	0.24
		$K = 4$	6	68.03%	0.25
		$K = 3$	4	58.93%	0.17
	ALSPCA	$\rho = 0.01$	31	71.49%	0.13
		$\rho = 0.1$	9	71.39%	0.13
		$\rho = 1$	4	66.53%	0.13
(100,20)	PCA			83.59%	
	DSPCA	$\rho = 0.1$	20	83.27%	8.58
		$\rho = 1$	20	72.75%	5.82
		$\rho = 10$	56	26.80%	3.01
	ADMM	$K = 10$	20	83.50%	0.32
		$K = 9$	18	83.02%	0.40
		$K = 8$	13	81.90%	0.27
	ALSPCA	$\rho = 0.01$	25	83.58%	0.13
		$\rho = 0.1$	19	83.37%	0.13
		$\rho = 1$	8	71.37%	0.13
(200,10)	PCA			51.69%	
	DSPCA	$\rho = 0.1$	10	51.46%	19.19
		$\rho = 1$	88	46.95%	28.51
		$\rho = 10$	1	19.80%	28.04
	ADMM	$K = 5$	9	51.15%	1.22
		$K = 4$	6	49.13%	1.38
		$K = 3$	4	42.61%	0.87
	ALSPCA	$\rho = 0.01$	10	51.61%	0.26
		$\rho = 0.1$	9	51.53%	0.25
		$\rho = 1$	4	48.01%	0.24
(200,20)	PCA			68.38%	
	DSPCA	$\rho = 0.1$	20	68.12%	74.87
		$\rho = 1$	20	59.54%	42.63
		$\rho = 10$	64	22.03%	34.83
	ADMM	$K = 9$	18	67.91%	1.46
		$K = 8$	14	67.01%	1.74
		$K = 7$	11	65.26%	1.76
	ALSPCA	$\rho = 0.01$	20	68.37%	0.24
		$\rho = 0.1$	18	68.20%	0.25
		$\rho = 1$	8	58.37%	0.24

5.5. Senate voting data. In this section, we use sparse PCA to analyze the voting records of the 109th US Senate, which was also studied by Zhang, d'Aspremont and El Ghaoui in [40]. The votes are recorded as 1 for "yes" and -1 for "no". Missing votes are recorded as 0. There are 100 senators (55 Republican, 44 Democratic and 1 independent) and 542 bills involved in the data set. However, there are many missing votes in the data set. To obtain a meaningful data matrix, we only choose the bills for which the number of missing votes is at most one. There are only 66 such bills among the 542 bills. So our data matrix M is a 66×100 matrix with entries 1, -1 and 0, and each column of M corresponds to one senator's voting. The

Table 5.6
Words associated with the first three sparse PCs using ADMM

1st PC (10 words)	2nd PC (12 words)	3rd PC (17 words)
case	bible	computer
course	case	email
email	christian	files
fact	course	ftp
help	evidence	graphics
number	fact	number
problem	god	phone
question	government	problem
system	human	program
university	jesus	research
	religion	science
	world	software
		space
		state
		university
		version
		windows
Total sparsity: 39, total explained variance: 12.72%		

sample covariance matrix $\Sigma = MM^\top$ in our test is a 66×66 matrix.

To see how standard PCA and sparse PCA perform in classifying the voting records, we implemented the following procedure as suggested in [40]. We used standard PCA to find the largest two PCs (denoted as v_1 and v_2) of Σ. We then projected each column of M onto the subspace spanned by v_1 and v_2, i.e., we found $\bar{\alpha}_i$ and $\bar{\beta}_i$ for each column M_i such that

$$(\bar{\alpha}_i, \bar{\beta}_i) := \arg\min_{(\alpha_i,\beta_i)} \|\alpha_i v_1 + \beta_i v_2 - M_i\|.$$

We then drew each column M_i as a point $(\bar{\alpha}_i, \bar{\beta}_i)$ in the two-dimensional subspace spanned by v_1 and v_2. The left figure in Figure 5.1 shows the 100 points. We see from this figure that senators are separated very well by partisanship. However, it is hard to interpret which bills are responsible to the explained variance, because all the bills are involved in the PCs. By using sparse PCA, we can interpret the explained variance by just a few bills. We applied our ADMM to find the first two sparse PCs (denoted as s_1 and s_2) of Σ. We set $K = 4$ for both problems and used the following continuation technique on μ: $\mu_0 = 100, \mu_k = \max\{2\mu_{k-1}/3, 10^{-4}\}$.

The resulting two sparse PCs s_1 and s_2 produced by our ADMM have 9 and 5 nonzeros respectively. We projected each column of M onto the subspace spanned by these two sparse PCs. The right figure in Figure 5.1 shows the 100 projections onto the subspace spanned by the sparse PCs s_1 and s_2. We see from this figure that the senators are still separated well by partisanship. Now since only a few bills are involved in the two sparse PCs, we can interpret which bills are responsible most for the classification. The bills involved in the first two PCs are listed in Table 5.7. From Table 5.7 we see that the most controversial issues between Republicans and Democrats are topics such as “Budget” and “Appropriations”. Other controversial issues involve topics like “Energy”, “Abortion” and “Health”.

6. Conclusion.

In this paper, we proposed alternating direction method of multipliers to solve an SDP relaxation of the sparse PCA problem. Our method incorporated a variable-splitting technique to

TABLE 5.7
Bills involved in the first two PCs by ADMM

Bills in the first sparse PC
Budget, Spending and Taxes_Education Funding Amendment_3804
Budget, Spending and Taxes_Reinstate Pay-As-You-Go through 2011 Amendment_3806
Energy Issues_LIHEAP Funding Amendment_3808
Abortion Issues_Unintended Pregnancy Amendment_3489
Budget, Spending and Taxes_Budget FY2006 Appropriations Resolution_3488
Budget, Spending and Taxes_Budget Reconciliation bill_3665
Budget, Spending and Taxes_Budget Reconciliation bill_3789
Budget, Spending and Taxes_Education Amendment_3490
Health Issues_Medicaid Amendment_3496
Bills in the second sparse PC
Appropriations_Agriculture, Rural Development, FDA Appropriations Act_3677
Appropriations_Emergency Supplemental Appropriations Act, 2005_3515
Appropriations_Emergency Supplemental Appropriations Act, 2006_3845
Appropriations_Interior Department FY 2006 Appropriations Bill_3595
Executive Branch_John Negroponte, Director of National Intelligence_3505

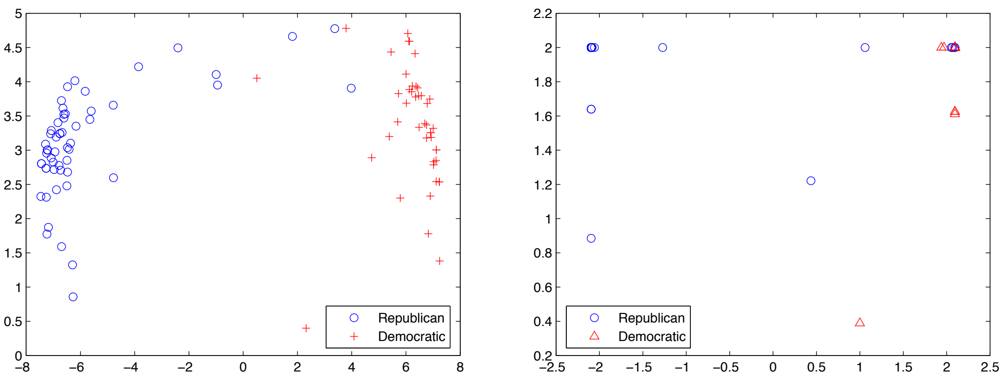

FIG. 5.1. *Projection of the senate voting records onto the subspace spanned by the top 2 principal components: Left: standard PCA; Right: sparse PCA*

separate the ℓ_1 norm constraint, which controls the sparsity of the solution, and the positive-semidefiniteness constraint. This method resulted in two relatively simple subproblems that have closed-form solutions in each iteration. Global convergence results were established for the proposed method. Numerical results on both synthetic data and real data from classification of text data and senate voting records demonstrated the efficacy of our method.

Compared with Nesterov's first-order method DSPCA for sparse PCA studied in [9], our ADMM method solves the primal problems directly and guarantees sparse solutions. Numerical results also indicate that ADMM is much faster than DSPCA. Compared with methods for solving nonconvex formulations of sparse PCA, the nonsmooth SDP formulation considered in this paper usually requires more computational effort in each iteration. However, the global convergence of our ADMM for solving the nonsmooth SDP is guaranteed, while methods for solving nonconvex problems usually have only local convergence.

Acknowledgement. The author is grateful to Alexandre d'Aspremont for discussions on using DSPCA. The author thanks Stephen J. Wright and Lingzhou Xue for reading an earlier version of the manuscript and for helpful discussions. This work was partially supported by the NSF Postdoctoral Fellowship through the Institute for Mathematics and Its Applications at University of Minnesota.

REFERENCES

[1] Farid Alizadeh, *Interior point methods in semidefinite programming with applications to combinatorial optimization*, SIAM Journal on Optimization, 5 (1993), pp. 13–51.

[2] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, *Distributed optimization and statistical learning via the alternating direction method of multipliers*, Foundations and Trends in Machine Learning, (2011).

[3] Stephen Boyd and Lieven Vandenberghe, *Convex optimization*, Cambridge University Press, Cambridge, 2004.

[4] P. Brucker, *An $\mathcal{O}(n)$ algorithm for quadratic knapsack problems*, Operations Research Letters, 3 (1984), pp. 163–166.

[5] E. J. Candès, J. Romberg, and T. Tao, *Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information*, IEEE Transactions on Information Theory, 52 (2006), pp. 489–509.

[6] P. L. Combettes and Jean-Christophe Pesquet, *A Douglas-Rachford splitting approach to nonsmooth convex variational signal recovery*, IEEE Journal of Selected Topics in Signal Processing, 1 (2007), pp. 564–574.

[7] P. L. Combettes and V. R. Wajs, *Signal recovery by proximal forward-backward splitting*, SIAM Journal on Multiscale Modeling and Simulation, 4 (2005), pp. 1168–1200.

[8] A. d'Aspremont, F. Bach, and L. El Ghaoui, *Optimal solutions for sparse principal component analysis*, Journal of Machine Learning Research, 9 (2008), pp. 1269–1294.

[9] A. d'Aspremont, L. El Ghaoui, M. I. Jordan, and G. R. G. Lanckriet, *A direct formulation for sparse pca using semidefinite programming*, SIAM Review, 49 (2007), pp. 434–448.

[10] D. Donoho, *Compressed sensing*, IEEE Transactions on Information Theory, 52 (2006), pp. 1289–1306.

[11] J. Douglas and H. H. Rachford, *On the numerical solution of the heat conduction problem in 2 and 3 space variables*, Transactions of the American Mathematical Society, 82 (1956), pp. 421–439.

[12] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra, *Efficient projections onto the l1-ball for learning in high dimensions*, in ICML, 2008.

[13] J. Eckstein, *Splitting methods for monotone operators with applications to parallel optimization*, PhD thesis, Massachusetts Institute of Technology, 1989.

[14] J. Eckstein and D. P. Bertsekas, *On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators*, Math. Program., 55 (1992), pp. 293–318.

[15] D. Gabay, *Applications of the method of multipliers to variational inequalities*, in Augmented Lagrangian Methods: Applications to the Solution of Boundary Value Problems, M. Fortin and R. Glowinski, eds., North-Hollan, Amsterdam, 1983.

[16] R. Glowinski and P. Le Tallec, *Augmented Lagrangian and Operator-Splitting Methods in Nonlinear Mechanics*, SIAM, Philadelphia, Pennsylvania, 1989.

[17] D. Goldfarb and S. Ma, *Fast multiple splitting algorithms for convex optimization*, tech. report, Department of IEOR, Columbia University. Preprint available at http://arxiv.org/abs/0912.4570, 2009.

[18] D. Goldfarb, S. Ma, and K. Scheinberg, *Fast alternating linearization methods for minimizing the sum of two convex functions*, tech. report, Department of IEOR, Columbia University. Preprint available at http://arxiv.org/abs/0912.4571, 2010.

[19] T. Goldstein and S. Osher, *The split Bregman method for L1-regularized problems*, SIAM J. Imaging Sci., 2 (2009), pp. 323–343.

[20] B. S. He, L.-Z. Liao, D. Han, and H. Yang, *A new inexact alternating direction method for monotone variational inequalities*, Math. Program., 92 (2002), pp. 103–118.

[21] J. Jeffers, *Two case studies in the application of principal component analysis*, Appl. Stat., 16 (1967), pp. 225–236.

[22] M. Journee, Yu. Nesterov, P. Richtarik, and R. Sepulchre, *Generalized power method for sparse principal component analysis*, Journal of Machine Learning Research, 11 (2010), pp. 517–553.

[23] P. L. Lions and B. Mercier, *Splitting algorithms for the sum of two nonlinear operators*, SIAM Journal on Numerical Analysis, 16 (1979), pp. 964–979.

[24] Z. Lu and Y. Zhang, *An augmented lagrangian approach for sparse principal component analysis*, Mathematical Programming, (2011).

[25] L. Mackey, *Deflation methods for sparse PCA*, in Advances in Neural Information Processing Systems (NIPS), 2008.

[26] J. Malick, J. Povh, F. Rendl, and A. Wiegele, *Regularization methods for semidefinite programming*, SIAM Journal on Optimization, 20 (2009), pp. 336–356.

[27] Y. E. Nesterov, *A method for unconstrained convex minimization problem with the rate of convergence $\mathcal{O}(1/k^2)$*, Dokl. Akad. Nauk SSSR, 269 (1983), pp. 543–547.

[28] ———, *Smooth minimization for non-smooth functions*, Math. Program. Ser. A, 103 (2005), pp. 127–152.

[29] P. M. Pardalos and N. Kovoor, *An algorithm for a singly constrained class of quadratic programs subject to upper and lower bounds*, Mathematical Programming, 46 (1990), pp. 321–328.

[30] D. H. Peaceman and H. H. Rachford, *The numerical solution of parabolic elliptic differential equations*, SIAM Journal on Applied Mathematics, 3 (1955), pp. 28–41.

[31] Shalev-Shwartz S. and Y. Singer, *Efficient learning of label ranking by soft projections onto polyhedra*, Journal of Machine Learning Research, 7 (2006), pp. 1567–1599.

[32] Y. Saad, *Projection and deflation methods for partial pole assignment in linear state feedback*, IEEE Trans. Automat. Contr., 33 (1998), pp. 290–297.

[33] K. Scheinberg, S. Ma, and D. Goldfarb, *Sparse inverse covariance selection via alternating linearization methods*, in Proceedings of the Neural Information Processing Systems (NIPS), 2010.

[34] M. J. Todd, *Semidefinite optimization*, Acta Numer., 10 (2001), pp. 515–560.

[35] E. van den Berg and M. P. Friedlander, *Probing the Pareto frontier for basis pursuit solutions*, SIAM J. on Scientific Computing, 31 (2008), pp. 890–912.

[36] Y. Wang, J. Yang, W. Yin, and Y. Zhang, *A new alternating minimization algorithm for total variation image reconstruction*, SIAM Journal on Imaging Sciences, 1 (2008), pp. 248–272.

[37] Z. Wen, D. Goldfarb, and W. Yin, *Alternating direction augmented Lagrangian methods for semidefinite programming*, Mathematical Programming Computation, 2 (2010), pp. 203–230.

[38] J. Yang and Y. Zhang, *Alternating direction algorithms for ℓ_1 problems in compressive sensing*, SIAM Journal on Scientific Computing, 33 (2011), pp. 250–278.

[39] X. Yuan, *Alternating direction methods for sparse covariance selection*, (2009). Preprint available at http://www.optimization-online.org/DB_HTML/2009/09/2390.html.

[40] Y. Zhang, A. d'Aspremont, and L. El Ghaoui, *Sparse PCA: Convex relaxations, algorithms and applications*, Handbook on Semidefinite, Cone and Polynomial Optimization, M. Anjos and J.B. Lasserre, editors, (2011).

[41] Y. Zhang and L. El Ghaoui, *Large-scale sparse principal component analysis with application to text data*, in NIPS, 2011.

[42] H. Zou, T. Hastie, and R. Tibshirani, *Sparse principle component analysis*, J. Comput. Graph. Stat., 15 (2006), pp. 265–286.