Sparse Inverse Covariance Selection via Alternating Linearization Methods

Katya Scheinberg
Department of ISE
Lehigh University
katyas@lehigh.edu

Shiqian Ma, Donald Goldfarb
Department of IEOR
Columbia University
{sm2756,goldfarb}@columbia.edu

Abstract

Gaussian graphical models are of great interest in statistical learning. Because the conditional independencies between different nodes correspond to zero entries in the inverse covariance matrix of the Gaussian distribution, one can learn the structure of the graph by estimating a sparse inverse covariance matrix from sample data, by solving a convex maximum likelihood problem with an ℓ_1-regularization term. In this paper, we propose a first-order method based on an alternating linearization technique that exploits the problem's special structure; in particular, the subproblems solved in each iteration have closed-form solutions. Moreover, our algorithm obtains an ϵ-optimal solution in $O(1/\epsilon)$ iterations. Numerical experiments on both synthetic and real data from gene association networks show that a practical version of this algorithm outperforms other competitive algorithms.

1 Introduction

In multivariate data analysis, graphical models such as Gaussian Markov Random Fields provide a way to discover meaningful interactions among variables. Let $Y = \{y^{(1)}, \ldots, y^{(n)}\}$ be an n-dimensional random vector following an n-variate Gaussian distribution $\mathcal{N}(\mu, \Sigma)$, and let $G = (V, E)$ be a Markov network representing the conditional independence structure of $\mathcal{N}(\mu, \Sigma)$. Specifically, the set of vertices $V = \{1, \ldots, n\}$ corresponds to the set of variables in Y, and the edge set E contains an edge (i, j) if and only if $y^{(i)}$ is conditionally dependent on $y^{(j)}$ given all remaining variables; i.e., the lack of an edge between i and j denotes the conditional independence of $y^{(i)}$ and $y^{(j)}$, which corresponds to a zero entry in the inverse covariance matrix Σ^{-1} ([1]). Thus learning the structure of this graphical model is equivalent to the problem of learning the zero-pattern of Σ^{-1}. To estimate this sparse inverse covariance matrix, one can solve the following sparse inverse covariance selection (SICS) problem: $\max_{X \in S^n_{++}} \log\det(X) - \langle \hat{\Sigma}, X\rangle - \rho\|X\|_0$, where S^n_{++} denotes the set of $n \times n$ positive definite matrices, $\|X\|_0$ is the number of nonzeros in X, $\hat{\Sigma} = \frac{1}{p}\sum_{i=1}^{p}(Y_i - \hat{\beta})(Y_i - \hat{\beta})^\top$ is the sample covariance matrix, $\hat{\beta} = \frac{1}{p}\sum_{i=1}^{p} Y_i$ is the sample mean and Y_i is the i-th random sample of Y. This problem is NP-hard in general due to the combinatorial nature of the cardinality term $\rho\|X\|_0$ ([2]). To get a numerically tractable problem, one can replace the cardinality term $\|X\|_0$ by $\|X\|_1 := \sum_{i,j}|X_{ij}|$, the envelope of $\|X\|_0$ over the set $\{X \in \mathbb{R}^{n\times n} : \|X\|_\infty \le 1\}$ (see [3]). This results in the convex optimization problem (see e.g., [4, 5, 6, 7]):

$$\min_{X \in S^n_{++}} \quad -\log\det(X) + \langle \hat{\Sigma}, X\rangle + \rho\|X\|_1. \tag{1}$$

Note that (1) can be rewritten as $\min_{X\in S^n_{++}} \max_{\|U\|_\infty \le \rho} -\log\det X + \langle \hat{\Sigma} + U, X\rangle$, where $\|U\|_\infty$ is the largest absolute value of the entries of U. By exchanging the order of max and min, we obtain

the dual problem $\max_{\|U\|_\infty \le \rho} \min_{X \in S^n_{++}} -\log\det X + \langle \hat{\Sigma} + U, X\rangle$, which is equivalent to

$$\max_{W \in S^n_{++}} \{\log\det W + n : \|W - \hat{\Sigma}\|_\infty \le \rho\}. \tag{2}$$

Both the primal and dual problems have strictly convex objectives; hence, their optimal solutions are unique. Given a dual solution W, $X = W^{-1}$ is primal feasible resulting in the duality gap

$$gap := \langle \hat{\Sigma}, W^{-1}\rangle + \rho\|W^{-1}\|_1 - n. \tag{3}$$

The primal and the dual SICS problems (1) and (2) are semidefinite programming problems and can be solved via interior point methods (IPMs) in polynomial time. However, the per-iteration computational cost and memory requirements of an IPM are prohibitively high for the SICS problem. Although an approximate IPM has recently been proposed for the SICS problem [8], most of the methods developed for it are first-order methods. Banerjee et al. [7] proposed a block coordinate descent (BCD) method to solve the dual problem (2). Their method updates one row and one column of W in each iteration by solving a convex quadratic programming problem by an IPM. The $glasso$ method of Friedman et al. [5] is based on the same BCD approach as in [7], but it solves each subproblem as a LASSO problem by yet another coordinate descent (CD) method [9]. Sun et al. [10] proposed solving the primal problem (1) by using a BCD method. They formulate the subproblem as a min-max problem and solve it using a prox method proposed by Nemirovski [11]. The SINCO method proposed by Scheinberg and Rish [12] is a greedy CD method applied to the primal problem. All of these BCD and CD approaches lack iteration complexity bounds. They also have been shown to be inferior in practice to gradient based approaches. A projected gradient method for solving the dual problem (2) that is considered to be state-of-the-art for SICS was proposed by Duchi et al. [13]. However, there are no iteration complexity results for it either. Variants of Nesterov's method [14, 15] have been applied to solve the SICS problem. d'Aspremont et al. [16] applied Nesterov's optimal first-order method to solve the primal problem (1) after smoothing the nonsmooth ℓ_1 term, obtaining an iteration complexity bound of $O(1/\epsilon)$ for an ϵ-optimal solution, but the implementation in [16] was very slow and did not produce good results. Lu [17] solved the dual problem (2), which is a smooth problem, by Nesterov's algorithm, and improved the iteration complexity to $O(1/\sqrt{\epsilon})$. However, since the practical performance of this algorithm was not attractive, Lu gave a variant (VSM) of it that exhibited better performance. The iteration complexity of VSM is unknown. Yuan [18] proposed an alternating direction method based on an augmented Lagrangian framework (see the ADAL method (8) below). This method also lacks complexity results. The proximal point algorithm proposed by Wang et al. in [19] requires a reformulation of the problem that increases the size of the problem making it impractical for solving large-scale problems. Also, there is no iteration complexity bound for this algorithm. The IPM in [8] also requires such a reformulation.

Our contribution. In this paper, we propose an alternating linearization method (ALM) for solving the primal SICS problem. An advantage of solving the primal problem is that the ℓ_1 penalty term in the objective function directly promotes sparsity in the optimal inverse covariance matrix.

Although developed independently, our method is closely related to Yuan's method [18]. Both methods exploit the special form of the primal problem (1) by alternatingly minimizing one of the terms of the objective function plus an approximation to the other term. The main difference between the two methods is in the construction of these approximations. As we will show, our method has a theoretically justified interpretation and is based on an algorithmic framework with complexity bounds, while no complexity bound is available for Yuan's method. Also our method has an intuitive interpretation from a learning perspective. Extensive numerical test results on both synthetic data and real problems have shown that our ALM algorithm significantly outperforms other existing algorithms, such as the PSM algorithm proposed by Duchi et al. [13] and the VSM algorithm proposed by Lu [17]. Note that it is shown in [13] and [17] that PSM and VSM outperform the BCD method in [7] and $glasso$ in [5].

Organization of the paper. In Section 2 we briefly review alternating linearization methods for minimizing the sum of two convex functions and establish convergence and iteration complexity results. We show how to use ALM to solve SICS problems and give intuition from a learning perspective in Section 3. Finally, we present some numerical results on both synthetic and real data in Section 4 and compare ALM with PSM algorithm [13] and VSM algorithm [17].

2 Alternating Linearization Methods

We consider here the alternating linearization method (ALM) for solving the following problem:

$$\min \quad F(x) \equiv f(x) + g(x), \tag{4}$$

where f and g are both convex functions. An effective way to solve (4) is to "split" f and g by introducing a new variable, i.e., to rewrite (4) as

$$\min_{x,y}\{f(x) + g(y) : x - y = 0\}, \tag{5}$$

and apply an alternating direction augmented Lagrangian method to it. Given a penalty parameter $1/\mu$, at the k-th iteration, the augmented Lagrangian method minimizes the augmented Lagrangian function

$$\mathcal{L}(x, y; \lambda) := f(x) + g(y) - \langle \lambda, x - y\rangle + \frac{1}{2\mu}\|x - y\|_2^2,$$

with respect to x and y, i.e., it solves the subproblem

$$(x^k, y^k) := \arg\min_{x,y} \mathcal{L}(x, y; \lambda^k), \tag{6}$$

and updates the Lagrange multiplier λ via:

$$\lambda^{k+1} := \lambda^k - (x^k - y^k)/\mu. \tag{7}$$

Since minimizing $\mathcal{L}(x, y; \lambda)$ with respect to x and y jointly is usually difficult, while doing so with respect to x and y alternatingly can often be done efficiently, the following alternating direction version of the augmented Lagrangian method (ADAL) is often advocated (see, e.g., [20, 21]):

$$\left\{\begin{array}{ll} x^{k+1} & := \arg\min_x \mathcal{L}(x, y^k; \lambda^k) \\ y^{k+1} & := \arg\min_y \mathcal{L}(x^{k+1}, y; \lambda^k) \\ \lambda^{k+1} & := \lambda^k - (x^{k+1} - y^{k+1})/\mu. \end{array}\right. \tag{8}$$

If we also update λ after we solve the subproblem with respect to x, we get the following symmetric version of the ADAL method.

$$\left\{\begin{array}{ll} x^{k+1} & := \arg\min_x \mathcal{L}(x, y^k; \lambda_y^k) \\ \lambda_x^{k+1} & := \lambda_y^k - (x^{k+1} - y^k)/\mu \\ y^{k+1} & := \arg\min_y \mathcal{L}(x^{k+1}, y; \lambda_x^{k+1}) \\ \lambda_y^{k+1} & := \lambda_x^{k+1} - (x^{k+1} - y^{k+1})/\mu. \end{array}\right. \tag{9}$$

Algorithm (9) has certain theoretical advantages when f and g are smooth. In this case, from the first-order optimality conditions for the two subproblems in (9), we have that:

$$\lambda_x^{k+1} = \nabla f(x^{k+1}) \quad \text{and} \quad \lambda_y^{k+1} = -\nabla g(y^{k+1}). \tag{10}$$

Substituting these relations into (9), we obtain the following equivalent algorithm for solving (4), which we refer to as the alternating linearization minimization (ALM) algorithm.

Algorithm 1 Alternating linearization method (ALM) for smooth problem

Input: $x^0 = y^0$
for $k = 0, 1, \cdots$ **do**
1. Solve $x^{k+1} := \arg\min_x Q_g(x, y^k) \equiv f(x) + g(y^k) + \left\langle \nabla g(y^k), x - y^k \right\rangle + \frac{1}{2\mu}\|x - y^k\|_2^2$;
2. Solve $y^{k+1} := \arg\min_y Q_f(x^{k+1}, y) \equiv f(x^{k+1}) + \left\langle \nabla f(x^{k+1}), y - x^{k+1} \right\rangle + \frac{1}{2\mu}\|y - x^{k+1}\|_2^2 + g(y)$;

end for

Algorithm 1 can be viewed in the following way: at each iteration we construct a quadratic approximation of the function $g(x)$ at the current iterate y^k and minimize the sum of this approximation and $f(x)$. The approximation is based on linearizing $g(x)$ (hence the name ALM) and adding a "prox" term $\frac{1}{2\mu}\|x - y^k\|_2^2$. When μ is small enough ($\mu \leq 1/L(g)$, where $L(g)$ is the Lipschitz constant for

∇g) this quadratic function, $g(y^k)+\langle \nabla g(y^k), x-y^k\rangle + \frac{1}{2\mu}\|x-y^k\|_2^2$ is an upper approximation to $g(x)$, which means that the reduction in the value of $F(x)$ achieved by minimizing $Q_g(x,y^k)$ in Step 1 is not smaller than the reduction achieved in the value of $Q_g(x,y^k)$ itself. Similarly, in Step 2 we build an upper approximation to $f(x)$ at x^{k+1}, $f(x^{k+1})+\langle \nabla f(x^{k+1}), y-x^{k+1}\rangle + \frac{1}{2\mu}\|y-x^{k+1}\|_2^2$, and minimize the sum $Q_f(x^{k+1},y)$ of it and $g(y)$.

Let us now assume that $f(x)$ is in the class $C^{1,1}$ with Lipschitz constant $L(f)$, while $g(x)$ is simply convex. Then from the first-order optimality conditions for the second minimization in (9), we have $-\lambda_y^{k+1} \in \partial g(y^{k+1})$, the subdifferential of $g(y)$ at $y = y^{k+1}$. Hence, replacing $\nabla g(y^k)$ in the definition of $Q_g(x,y^k)$ by $-\lambda_y^{k+1}$ in (9), we obtain the following modified version of (9).

Algorithm 2 Alternating linearization method with skipping step

Input: $x^0 = y^0$
for $k = 0, 1, \cdots$ **do**
1. Solve $x^{k+1} := \arg\min_x Q(x,y^k) \equiv f(x) + g(y^k) - \langle \lambda^k, x-y^k\rangle + \frac{1}{2\mu}\|x-y^k\|_2^2$;
2. If $F(x^{k+1}) > Q(x^{k+1},y^k)$ then $x^{k+1} := y^k$.
3. Solve $y^{k+1} := \arg\min_y Q_f(x^{k+1},y)$;
4. $\lambda^{k+1} = \nabla f(x^{k+1}) - (x^{k+1}-y^{k+1})/\mu$.

end for

Algorithm 2 is identical to the symmetric ADAL algorithm (9) as long as $F(x^{k+1}) \le Q(x^{k+1},y^k)$ at each iteration (and to Algorithm 1 if $g(x)$ is in $C^{1,1}$ and $\mu \le 1/\max\{L(f),L(g)\}$). If this condition fails, then the algorithm simply sets $x^{k+1} \leftarrow y^k$. Algorithm 2 has the following convergence property and iteration complexity bound. For a proof see the Appendix.

Theorem 2.1. *Assume ∇f is Lipschitz continuous with constant $L(f)$. For $\beta/L(f) \le \mu \le 1/L(f)$ where $0 < \beta \le 1$, Algorithm 2 satisfies*

$$F(y^k) - F(x^*) \le \frac{\|x^0 - x^*\|^2}{2\mu(k+k_n)}, \forall k, \tag{11}$$

where x^* is an optimal solution of (4) *and k_n is the number of iterations until the $k-th$ for which $F(x^{k+1}) \le Q(x^{k+1},y^k)$. Thus Algorithm 2 produces a sequence which converges to the optimal solution in function value, and the number of iterations needed is $O(1/\epsilon)$ for an ϵ-optimal solution.*

If $g(x)$ is also a smooth function in the class $C^{1,1}$ with Lipschitz constant $L(g) \le 1/\mu$, then Theorem 2.1 also applies to Algorithm 1 since in this case $k_n = k$ (i.e., no "skipping" occurs). Note that the iteration complexity bound in Theorem 2.1 can be improved. Nesterov [15, 22] proved that one can obtain an optimal iteration complexity bound of $O(1/\sqrt{\epsilon})$, using only first-order information. His acceleration technique is based on using a linear combination of previous iterates to obtain a point where the approximation is built. This technique has been exploited and extended by Tseng [23], Beck and Teboulle [24], Goldfarb et al. [25] and many others. A similar technique can be adopted to derive a fast version of Algorithm 2 that has an improved complexity bound of $O(1/\sqrt{\epsilon})$, while keeping the computational effort in each iteration almost unchanged. However, we do not present this method here, since when applied to the SICS problem, it did not work as well as Algorithm 2.

3 ALM for SICS

The SICS problem

$$\min_{X \in S_{++}^n} \quad F(X) \equiv f(X) + g(X), \tag{12}$$

where $f(X) = -\log\det(X) + \langle \hat{\Sigma}, X\rangle$ and $g(X) = \rho\|X\|_1$, is of the same form as (4). However, in this case neither $f(X)$ nor $g(X)$ have Lipschitz continuous gradients. Moreover, $f(X)$ is only defined for positive definite matrices while $g(X)$ is defined everywhere. These properties of the objective function make the SICS problem especially challenging for optimization methods. Nevertheless, we can still apply (9) to solve the problem directly. Moreover, we can apply Algorithm 2 and obtain the complexity bound in Theorem 2.1 as follows.

The $\log\det(X)$ term in $f(X)$ implicitly requires that $X \in S^n_{++}$ and the gradient of $f(X)$, which is given by $-X^{-1} + \hat{\Sigma}$, is not Lipschitz continuous in S^n_{++}. Fortunately, as proved in Proposition 3.1 in [17], the optimal solution of (12) $X^* \succeq \alpha I$, where $\alpha = \frac{1}{\|\hat{\Sigma}\|+n\rho}$. Therefore, if we define $\mathcal{C} := \{X \in S^n : X \succeq \frac{\alpha}{2} I\}$, the SICS problem (12) can be formulated as:

$$\min_{X,Y}\{f(X) + g(Y) : X - Y = 0, X \in \mathcal{C}, Y \in \mathcal{C}\}. \tag{13}$$

We can include constraints $X \in \mathcal{C}$ in Step 1 and $Y \in \mathcal{C}$ in Step 3 of Algorithm 2. Theorem 2.1 can then be applied as discussed in [25]. However, a difficulty now arises when performing the minimization in Y. Without the constraint $Y \in \mathcal{C}$, only a matrix shrinkage operation is needed, but with this additional constraint the problem becomes harder to solve. Minimization in X with or without the constraint $X \in \mathcal{C}$ is accomplished by performing an SVD. Hence the constraint can be easily imposed.

Instead of imposing constraint $Y \in \mathcal{C}$ we can obtain feasible solutions by a line search on μ. We know that the constraint $X \succeq \frac{\alpha}{2} I$ is not tight at the solution. Hence if we start the algorithm with $X \succeq \alpha I$ and restrict the step size μ to be sufficiently small then the iterates of the method will remain in $\mathcal{C}$.

Note however, that the bound on the Lipschitz constant of the gradient of $f(X)$ is $1/\alpha^2$ and hence can be very large. It is not practical to restrict μ in the algorithm to be smaller than α^2, since μ determines the step size at each iteration. Hence, for a practical approach we can only claim that the theoretical convergence rate bound holds in only a small neighborhood of the optimal solution. We now present a practical version of our algorithm applied to the SICS problem.

Algorithm 3 Alternating linearization method (ALM) for SICS

Input: $X^0 = Y^0$, μ_0.
for $k = 0, 1, \cdots$ **do**
0. Pick $\mu_{k+1} \leq \mu_k$.
1. Solve $X^{k+1} := \arg\min_{X \in \mathcal{C}} f(X) + g(Y^k) - \langle \Lambda^k, X - Y^k \rangle + \frac{1}{2\mu_{k+1}} \|X - Y^k\|_F^2$;
2. If $g(X^{k+1}) > g(Y^k) - \langle \Lambda^k, X^{k+1} - Y^k \rangle + \frac{1}{2\mu_{k+1}} \|X^{k+1} - Y^k\|_F^2$, then $X^{k+1} := Y^k$.
3. Solve $Y^{k+1} := \arg\min_Y f(X^{k+1}) + \langle \nabla f(X^{k+1}), Y - X^{k+1} \rangle + \frac{1}{2\mu_{k+1}} \|Y - X^{k+1}\|_F^2 + g(Y)$;
4. $\Lambda^{k+1} = \nabla f(X^{k+1}) - (X^{k+1} - Y^{k+1})/\mu_{k+1}$.
end for

We now show how to solve the two optimization problems in Algorithm 3. The first-order optimality conditions for Step 1 in Algorithm 3, ignoring the constraint $X \in \mathcal{C}$ are:

$$\nabla f(X) - \Lambda^k + (X - Y^k)/\mu_{k+1} = 0. \tag{14}$$

Consider $V\text{Diag}(d)V^\top$ - the spectral decomposition of $Y^k + \mu_{k+1}(\Lambda^k - \hat{\Sigma})$ and let

$$\gamma_i = \left(d_i + \sqrt{d_i^2 + 4\mu_{k+1}}\right)/2, i = 1, \ldots, n. \tag{15}$$

Since $\nabla f(X) = -X^{-1} + \hat{\Sigma}$, it is easy to verify that $X^{k+1} := V\text{Diag}(\gamma)V^\top$ satisfies (14). When the constraint $X \in \mathcal{C}$ is imposed, the optimal solution changes to $X^{k+1} := V\text{Diag}(\gamma)V^\top$ with $\gamma_i = \max\left\{\alpha/2, \left(d_i + \sqrt{d_i^2 + 4\mu_{k+1}}\right)/2\right\}, i = 1, \ldots, n$. We observe that solving (14) requires approximately the same effort ($O(n^3)$) as is required to compute $\nabla f(X^{k+1})$. Moreover, from the solution to (14), $\nabla f(X^{k+1})$ is obtained with only a negligible amount of additional effort, since $(X^{k+1})^{-1} := V\text{Diag}(\gamma)^{-1}V^\top$.

The first-order optimality conditions for Step 2 in Algorithm 3 are:

$$0 \in \nabla f(X^{k+1}) + (Y - X^{k+1})/\mu_{k+1} + \partial g(Y). \tag{16}$$

Since $g(Y) = \rho\|Y\|_1$, it is well known that the solution to (16) is given by

$$Y^{k+1} = \text{shrink}(X^{k+1} - \mu_{k+1}(\hat{\Sigma} - (X^{k+1})^{-1}), \mu_{k+1}\rho),$$

where the "shrinkage operator" $\mathrm{shrink}(Z, \rho)$ updates each element Z_{ij} of the matrix Z by the formula $\mathrm{shrink}(Z, \rho)_{ij} = \mathrm{sgn}(Z_{ij}) \cdot \max\{|Z_{ij}| - \rho, 0\}$.

The $O(n^3)$ complexity of Step 1, which requires a spectral decomposition, dominates the $O(n^2)$ complexity of Step 2 which requires a simple shrinkage. There is no closed-form solution for the subproblem corresponding to Y when the constraint $Y \in \mathcal{C}$ is imposed. Hence, we neither impose this constraint explicitly nor do so by a line search on μ_k, since in practice this degrades the performance of the algorithm substantially. Thus, the resulting iterates Y^k may not be positive definite, while the iterates X^k remain so. Eventually due to the convergence of Y^k and X^k, the Y^k iterates become positive definite and the constraint $Y \in \mathcal{C}$ is satisfied.

Let us now remark on the learning based intuition behind Algorithm 3. We recall that $-\Lambda^k \in \partial g(Y^k)$. The two steps of the algorithm can be written as

$$X^{k+1} := \arg\min_{X \in \mathcal{C}} \{f(X) + \frac{1}{2\mu_{k+1}} \|X - (Y^k + \mu_{k+1}\Lambda^k)\|_F^2\} \tag{17}$$

and

$$Y^{k+1} := \arg\min_{Y} \{g(Y) + \frac{1}{2\mu_{k+1}} \|Y - (X^{k+1} - \mu_{k+1}(\hat{\Sigma} - (X^{k+1})^{-1}))\|_F^2\}. \tag{18}$$

The SICS problem is trying to optimize two conflicting objectives: on the one hand it tries to find a covariance matrix X^{-1} that best fits the observed data, i.e., is as close to $\hat{\Sigma}$ as possible, and on the other hand it tries to obtain a sparse matrix X. The proposed algorithm address these two objectives in an alternating manner. Given an initial "guess" of the sparse matrix Y^k we update this guess by a subgradient descent step of length μ_{k+1}: $Y^k + \mu_{k+1}\Lambda^k$. Recall that $-\Lambda^k \in \partial g(Y^k)$. Then problem (17) seeks a solution X that optimizes the first objective (best fit of the data) while adding a regularization term which imposes a Gaussian prior on X whose mean is the current *guess* for the sparse matrix: $Y^k + \mu_{k+1}\Lambda^k$. The solution to (17) gives us a *guess* for the inverse covariance X^{k+1}. We again update it by taking a gradient descent step: $X^{k+1} - \mu_{k+1}(\hat{\Sigma} - (X^{k+1})^{-1})$. Then problem (18) seeks a sparse solution Y while also imposing a Gaussian prior on Y whose mean is the *guess* for the inverse covariance matrix $X^{k+1} - \mu_{k+1}(\hat{\Sigma} - (X^{k+1})^{-1})$. Hence the sequence of X^k's is a sequence of positive definite inverse covariance matrices that converge to a sparse matrix, while the sequence of Y^k's is a sequence of sparse matrices that converges to a positive definite inverse covariance matrix.

An important question is how to pick μ_{k+1}. Theory tells us that if we pick a small enough value, then we can obtain the complexity bounds. However, in practice this value is too small. We discuss the simple strategy that we use in the next section.

4 Numerical Experiments

In this section, we present numerical results on both synthetic and real data to demonstrate the efficiency of our SICS ALM algorithm. Our codes for ALM were written in MATLAB. All numerical experiments were run in MATLAB 7.3.0 on a Dell Precision 670 workstation with an Intel Xeon(TM) 3.4GHZ CPU and 6GB of RAM.

Since $-\Lambda^k \in \partial g(Y^k)$, $\|\Lambda^k\|_\infty \leq \rho$; hence $\hat{\Sigma} - \Lambda^k$ is a feasible solution to the dual problem (2) as long as it is positive definite. Thus the duality gap at the k-th iteration is given by:

$$Dgap := -\log\det(X^k) + \langle \hat{\Sigma}, X^k \rangle + \rho\|X^k\|_1 - \log\det(\hat{\Sigma} - \Lambda^k) - n. \tag{19}$$

We define the relative duality gap as: $Rel.gap := Dgap/(1 + |pobj| + |dobj|)$, where $pobj$ and $dobj$ are respectively the objective function values of the primal problem (12) at point X^k, and the dual problem (2) at $\hat{\Sigma} - \Lambda^k$. Defining $d_k(\phi(x)) \equiv \max\{1, \phi(x^k), \phi(x^{k-1})\}$, we measure the relative changes of objective function value $F(X)$ and the iterates X and Y as follows:

$$Frel := \frac{|F(X^k) - F(X^{k-1})|}{d_k(|F(X)|)},\ Xrel := \frac{\|X^k - X^{k-1}\|_F}{d_k(\|X\|_F)},\ Yrel := \frac{\|Y^k - Y^{k-1}\|_F}{d(\|Y\|_F)}.$$

We terminate ALM when either

$$(i) \quad Dgap \leq \epsilon_{gap} \quad \text{or} \quad (ii) \quad \max\{Frel, Xrel, Yrel\} \leq \epsilon_{rel}. \tag{20}$$

Note that in (19), computing $\log\det(X^k)$ is easy since the spectral decomposition of X^k is already available (see (14) and (15)), but computing $\log\det(\hat{\Sigma} - \Lambda^k)$ requires another expensive spectral decomposition. Thus, in practice, we only check (20)(i) every N_{gap} iterations. We check (20)(ii) at every iteration since this is inexpensive.

A continuation strategy for updating μ is also crucial to ALM. In our experiments, we adopted the following update rule. After every N_μ iterations, we set $\mu := \max\{\mu \cdot \eta_\mu, \bar{\mu}\}$; i.e., we simply reduce μ by a constant factor η_μ every N_μ iterations until a desired lower bound on μ is achieved.

We compare ALM (i.e., Algorithm 3 with the above stopping criteria and μ updates), with the projected subgradient method (PSM) proposed by Duchi et al. in [13] and implemented by Mark Schmidt [1] and the smoothing method (VSM) [2] proposed by Lu in [17], which are considered to be the state-of-the-art algorithms for solving SICS problems. The per-iteration complexity of all three algorithms is roughly the same; hence a comparison of the number of iterations is meaningful. The parameters used in PSM and VSM are set at their default values. We used the following parameter values in ALM: $\epsilon_{gap} = 10^{-3}, \epsilon_{rel} = 10^{-8}, N_{gap} = 20, N_\mu = 20, \bar{\mu} = \max\{\mu_0\eta_\mu^8, 10^{-6}\}, \eta_\mu = 1/3$, where μ_0 is the initial μ which is set according to ρ; specifically, in our experiments, $\mu_0 = 100/\rho$, if $\rho < 0.5$, $\mu_0 = \rho$ if $0.5 \le \rho \le 10$, and $\mu_0 = \rho/100$ if $\rho > 10$.

4.1 Experiments on synthetic data

We randomly created test problems using a procedure proposed by Scheinberg and Rish in [12]. Similar procedures were used by Wang et al. in [19] and Li and Toh in [8]. For a given dimension n, we first created a sparse matrix $U \in \mathbb{R}^{n\times n}$ with nonzero entries equal to -1 or 1 with equal probability. Then we computed $S := (U * U^\top)^{-1}$ as the true covariance matrix. Hence, S^{-1} was sparse. We then drew $p = 5n$ iid vectors, $Y_1, \ldots, Y_p$, from the Gaussian distribution $\mathcal{N}(\mathbf{0}, S)$ by using the $mvnrnd$ function in MATLAB, and computed a sample covariance matrix $\hat{\Sigma} := \frac{1}{p}\sum_{i=1}^p Y_i Y_i^\top$. We compared ALM with PSM [13] and VSM [17] on these randomly created data with different ρ. The PSM code was terminated using its default stopping criteria, which included (20)(i) with $\epsilon_{gap} = 10^{-3}$. VSM was also terminated when $Dgap \le 10^{-3}$. Since PSM and VSM solve the dual problem (2), the duality gap which is given by (3) is available without any additional spectral decompositions. The results are shown in Table 1. All CPU times reported are in seconds.

Table 1: Comparison of ALM, PSM and VSM on synthetic data

	ALM				PSM				VSM			
n	iter	Dgap	Rel.gap	CPU	iter	Dgap	Rel.gap	CPU	iter	Dgap	Rel.gap	CPU
$\rho = 0.1$												
200	300	8.70e-4	1.51e-6	13	1682	9.99e-4	1.74e-6	38	857	9.97e-4	1.73e-6	37
500	220	5.55e-4	4.10e-7	84	861	9.98e-4	7.38e-7	205	946	9.98e-4	7.38e-7	377
1000	180	9.92e-4	3.91e-7	433	292	9.91e-4	3.91e-7	446	741	9.97e-4	3.94e-7	1928
1500	199	1.73e-3	4.86e-7	1405	419	9.76e-4	2.74e-7	1975	802	9.98e-4	2.80e-7	6340
2000	200	6.13e-5	1.35e-8	3110	349	1.12e-3	2.46e-7	3759	915	1.00e-3	2.20e-7	16085
$\rho = 0.5$												
200	140	9.80e-4	1.15e-6	6	6106	1.00e-3	1.18e-6	137	1000	9.99e-4	1.18e-6	43
500	100	1.69e-4	7.59e-8	39	903	9.90e-4	4.46e-7	212	1067	9.99e-4	4.50e-7	425
1000	100	9.28e-4	2.12e-7	247	489	9.80e-4	2.24e-7	749	1039	9.95e-4	2.27e-7	2709
1500	140	2.17e-4	3.39e-8	1014	746	9.96e-4	1.55e-7	3514	1191	9.96e-4	1.55e-7	9405
2000	160	4.70e-4	5.60e-8	2529	613	9.96e-4	1.18e-7	6519	1640	9.99e-4	1.19e-7	28779
$\rho = 1.0$												
200	180	4.63e-4	4.63e-7	8	7536	1.00e-3	1.00e-6	171	1296	9.96e-4	9.96e-7	57
500	140	4.14e-4	1.56e-7	55	2099	9.96e-4	3.76e-7	495	1015	9.97e-4	3.76e-7	406
1000	160	3.19e-4	6.07e-8	394	774	9.83e-4	1.87e-7	1172	1310	9.97e-4	1.90e-7	3426
1500	180	8.28e-4	1.07e-7	1304	1088	9.88e-4	1.27e-7	5100	1484	9.96e-4	1.28e-7	11749
2000	240	9.58e-4	9.37e-8	3794	1158	9.35e-4	9.15e-8	12310	2132	9.99e-4	9.77e-8	37406

From Table 1 we see that on these randomly created SICS problems, ALM outperforms PSM and VSM in both accuracy and CPU time with the performance gap increasing as ρ increases. For example, for $\rho = 1.0$ and $n = 2000$, ALM achieves $Dgap = 9.58e-4$ in about 1 hour and 15 minutes, while PSM and VSM need about 3 hours and 25 minutes and 10 hours and 23 minutes, respectively, to achieve similar accuracy.

[1] The MATLAB can be downloaded from http://www.cs.ubc.ca/∼schmidtm/Software/PQN.html

[2] The MATLAB code can be downloaded from http://www.math.sfu.ca/∼zhaosong

4.2 Experiments on real data

We tested ALM on real data from gene expression networks using the five data sets from [8] provided to us by Kim-Chuan Toh: *(1) Lymph node status; (2) Estrogen receptor; (3) Arabidopsis thaliana; (4) Leukemia; (5) Hereditary breast cancer.* See [8] and references therein for the descriptions of these data sets. Table 2 presents our test results. As suggested in [8], we set $\rho = 0.5$. From Table 2 we see that ALM is much faster and provided more accurate solutions than PSM and VSM.

Table 2: Comparison of ALM, PSM and VSM on real data

		ALM				PSM				VSM			
prob.	n	iter	Dgap	Rel.gap	CPU	iter	Dgap	Rel.gap	CPU	iter	Dgap	Rel.gap	CPU
(1)	587	60	9.41e-6	5.78e-9	35	178	9.22e-4	5.67e-7	64	467	9.78e-4	6.01e-7	273
(2)	692	80	6.13e-5	3.32e-8	73	969	9.94e-4	5.38e-7	531	953	9.52e-4	5.16e-7	884
(3)	834	100	7.26e-5	3.27e-8	150	723	1.00e-3	4.50e-7	662	1097	7.31e-4	3.30e-7	1668
(4)	1255	120	6.69e-4	1.97e-7	549	1405	9.89e-4	2.91e-7	4041	1740	9.36e-4	2.76e-7	8568
(5)	1869	160	5.59e-4	1.18e-7	2158	1639	9.96e-4	2.10e-7	14505	3587	9.93e-4	2.09e-7	52978

4.3 Solution Sparsity

In this section, we compare the sparsity patterns of the solutions produced by ALM, PSM and VSM. For ALM, the sparsity of the solution is given by the sparsity of Y. Since PSM and VSM solve the dual problem, the primal solution X, obtained by inverting the dual solution W, is never sparse due to floating point errors. Thus it is not fair to measure the sparsity of X or a truncated version of X. Instead, we measure the sparsity of solutions produced by PSM and VSM by appealing to complementary slackness. Specifically, the (i, j)-th element of the inverse covariance matrix is deemed to be nonzero if and only if $|W_{ij} - \hat{\Sigma}_{ij}| = \rho$. We give results for a random problem ($n = 500$) and the first real data set in Table 3. For each value of ρ, the first three rows show the number of nonzeros in the solution and the last three rows show the number of entries that are nonzero in the solution produced by one of the methods but are zero in the solution produced by the other method. The sparsity of the ground truth inverse covariance matrix of the synthetic data is 6.76%. From Table 3 we can see that when ρ is relatively large ($\rho \geq 0.5$), all three algorithms

Table 3: Comparison of sparsity of solutions produced by ALM, PSM and VSM

ρ	100	50	10	5	1	0.5	0.1	0.05	0.01
synthetic problem data									
ALM	700	2810	11844	15324	28758	37510	63000	75566	106882
PSM	700	2810	11844	15324	28758	37510	63000	75566	106870
VSM	700	2810	11844	15324	28758	37510	63000	75568	106876
ALM vs PSM	0	0	0	0	0	0	0	2	14
PSM vs VSM	0	0	0	0	0	0	0	0	8
VSM vs ALM	0	0	0	0	0	0	0	2	2
real problem data									
ALM	587	587	587	587	587	4617	37613	65959	142053
PSM	587	587	587	587	587	4617	37615	65957	142051
VSM	587	587	587	587	587	4617	37613	65959	142051
ALM vs PSM	0	0	0	0	0	0	0	2	2
PSM vs VSM	0	0	0	0	0	0	2	0	0
VSM vs ALM	0	0	0	0	0	0	0	0	0

produce solutions with exactly the same sparsity patterns. Only when ρ is very small, are there slight differences. We note that the ROC curves depicting the trade-off between the number of true positive elements recovered versus the number of false positive elements as a function of the regularization parameter ρ are also almost identical for the three methods.

Acknowledgements

We would like to thank Professor Kim-Chuan Toh for providing the data set used in Section 4.2. The research reported here was supported in part by NSF Grants DMS 06-06712 and DMS 10-16571, ONR Grant N00014-08-1-1118 and DOE Grant DE-FG02-08ER25856.

References

[1] S. Lauritzen. *Graphical Models*. Oxford University Press, 1996.

[2] B. K. Natarajan. Sparse approximate solutions to linear systems. *SIAM Journal on Computing*, 24:227–234, 1995.

[3] J.-B. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms II: Advanced Theory and Bundle Methods*. Springer-Verlag, New York, 1993.

[4] M. Yuan and Y. Lin. Model selection and estimation in the Gaussian graphical model. *Biometrika*, 94(1):19–35, 2007.

[5] J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 2007.

[6] M. Wainwright, P. Ravikumar, and J. Lafferty. High-dimensional graphical model selection using ℓ_1-regularized logistic regression. *NIPS*, 19:1465–1472, 2007.

[7] O. Banerjee, L. El Ghaoui, and A. d'Aspremont. Model selection through sparse maximum likelihood estimation for multivariate gaussian for binary data. *Journal of Machine Learning Research*, 9:485–516, 2008.

[8] L. Li and K.-C. Toh. An inexact interior point method for l_1-regularized sparse covariance selection. *preprint*, 2010.

[9] R. Tibshirani. Regression shrinkage and selection via the lasso. *J. Royal. Statist. Soc B.*, 58(1):267–288, 1996.

[10] L. Sun, R. Patel, J. Liu, K. Chen, T. Wu, J. Li, E. Reiman, and J. Ye. Mining brain region connectivity for alzheimer's disease study via sparse inverse covariance estimation. *KDD'09*, 2009.

[11] A. Nemirovski. Prox-method with rate of convergence $O(1/t)$ for variational inequalities with Lipschitz continuous monotone operators and smooth convex-concave saddle point problems. *SIAM Journal on Optimization*, 15(1):229–251, 2005.

[12] K. Scheinberg and I. Rish. Sinco - a greedy coordinate ascent method for sparse inverse covariance selection problem. 2009. Preprint available at http://www.optimization-online.org/DB_HTML/2009/07/2359.html.

[13] J. Duchi, S. Gould, and D. Koller. Projected subgradient methods for learning sparse Gaussian. *Conference on Uncertainty in Artificial Intelligence (UAI 2008)*, 2008.

[14] Y. E. Nesterov. Smooth minimization for non-smooth functions. *Math. Program. Ser. A*, 103:127–152, 2005.

[15] Y. E. Nesterov. Introductory lectures on convex optimization. 87:xviii+236, 2004. A basic course.

[16] A. D'Aspremont, O. Banerjee, and L. El Ghaoui. First-order methods for sparse covariance selection. *SIAM Journal on Matrix Analysis and its Applications*, 30(1):56–66, 2008.

[17] Z. Lu. Smooth optimization approach for sparse covariance selection. *SIAM J. Optim.*, 19(4):1807–1827, 2009.

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

[19] C. Wang, D. Sun, and K.-C. Toh. Solving log-determinant optimization problems by a Newton-CG primal proximal point algorithm. *preprint*, 2009.

[20] M. Fortin and R. Glowinski. *Augmented Lagrangian methods: applications to the numerical solution of boundary-value problems*. North-Holland Pub. Co., 1983.

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

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

[23] P. Tseng. On accelerated proximal gradient methods for convex-concave optimization. *submitted to SIAM J. Optim.*, 2008.

[24] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Imaging Sciences*, 2(1):183–202, 2009.

[25] D. Goldfarb, S. Ma, and K. Scheinberg. Fast alternating linearization methods for minimizing the sum of two convex functions. Technical report, Department of IEOR, Columbia University, 2010.