Sparse Inverse Covariance Matrix Estimation Using Quadratic Approximation

Cho-Jui Hsieh, Mátyás A. Sustik, Inderjit S. Dhillon, and Pradeep Ravikumar
Department of Computer Science
University of Texas at Austin
Austin, TX 78712 USA
{cjhsieh,sustik,inderjit,pradeepr}@cs.utexas.edu

Abstract

The ℓ_1 regularized Gaussian maximum likelihood estimator has been shown to have strong statistical guarantees in recovering a sparse inverse covariance matrix, or alternatively the underlying graph structure of a Gaussian Markov Random Field, from very limited samples. We propose a novel algorithm for solving the resulting optimization problem which is a regularized log-determinant program. In contrast to other state-of-the-art methods that largely use first order gradient information, our algorithm is based on Newton's method and employs a quadratic approximation, but with some modifications that leverage the structure of the sparse Gaussian MLE problem. We show that our method is superlinearly convergent, and also present experimental results using synthetic and real application data that demonstrate the considerable improvements in performance of our method when compared to other state-of-the-art methods.

1 Introduction

Gaussian Markov Random Fields; Covariance Estimation. Increasingly, in modern settings statistical problems are high-dimensional, where the number of parameters is large when compared to the number of observations. An important class of such problems involves estimating the graph structure of a Gaussian Markov random field (GMRF) in the high-dimensional setting, with applications ranging from inferring gene networks and analyzing social interactions. Specifically, given n independently drawn samples $\{\mathbf{y_1}, \mathbf{y_2}, \ldots, \mathbf{y_n}\}$ from a p-variate Gaussian distribution, so that $\mathbf{y_i} \sim \mathcal{N}(\mu, \Sigma)$, the task is to estimate its inverse covariance matrix Σ^{-1}, also referred to as the *precision* or *concentration* matrix. The non-zero pattern of this inverse covariance matrix Σ^{-1} can be shown to correspond to the underlying graph structure of the GMRF. An active line of work in high-dimensional settings where $p < n$ is thus based on imposing some low-dimensional structure, such as sparsity or graphical model structure on the model space. Accordingly, a line of recent papers [2, 8, 20] has proposed an estimator that minimizes the Gaussian negative log-likelihood regularized by the ℓ_1 norm of the entries (off-diagonal entries) of the inverse covariance matrix. The resulting optimization problem is a log-determinant program, which is convex, and can be solved in polynomial time.

Existing Optimization Methods for the regularized Gaussian MLE. Due in part to its importance, there has been an active line of work on efficient optimization methods for solving the ℓ_1 regularized Gaussian MLE problem. In [8, 2] a block coordinate descent method has been proposed which is called the *graphical lasso* or GLASSO for short. Other recent algorithms proposed for this problem include PSM that uses projected subgradients [5], ALM using alternating linearization [14], IPM an inexact interior point method [11] and SINCO a greedy coordinate descent method [15].

For typical high-dimensional statistical problems, optimization methods typically suffer sub-linear rates of convergence [1]. This would be too expensive for the Gaussian MLE problem, since the

number of matrix entries scales quadratically with the number of nodes. Luckily, the log-determinant problem has special structure; the log-determinant function is strongly convex and one can observe linear (i.e. geometric) rates of convergence for the state-of-the-art methods listed above. However, at most linear rates in turn become infeasible when the problem size is very large, with the number of nodes in the thousands and the number of matrix entries to be estimated in the millions. Here we ask the question: *can we obtain superlinear rates of convergence for the optimization problem underlying the ℓ_1 regularized Gaussian MLE?*

One characteristic of these state-of-the-art methods is that they are first-order iterative methods that mainly use gradient information at each step. Such first-order methods have become increasingly popular in recent years for high-dimensional problems in part due to their ease of implementation, and because they require very little computation and memory at each step. The caveat is that they have at most linear rates of convergence [3]. For superlinear rates, one has to consider second-order methods which at least in part use the Hessian of the objective function. There are however some caveats to the use of such second-order methods in high-dimensional settings. First, a straightforward implementation of each second-order step would be very expensive for high-dimensional problems. Secondly, the log-determinant function in the Gaussian MLE objective acts as a barrier function for the positive definite cone. This barrier property would be lost under quadratic approximations so there is a danger that Newton-like updates will not yield positive-definite matrices, unless one explicitly enforces such a constraint in some manner.

Our Contributions. In this paper, we present a new second-order algorithm to solve the ℓ_1 regularized Gaussian MLE. We perform Newton steps that use iterative quadratic approximations of the Gaussian negative log-likelihood, but with three innovations that enable finessing the caveats detailed above. First, we provide an efficient method to compute the Newton direction. As in recent methods [12, 9], we build on the observation that the Newton direction computation is a *Lasso* problem, and perform iterative coordinate descent to solve this Lasso problem. However, the naive approach has an update cost of $O(p^2)$ for performing each coordinate descent update in the inner loop, which makes this resume infeasible for this problem. But we show how a careful arrangement and caching of the computations can reduce this cost to $O(p)$. Secondly, we use an Armijo-rule based step size selection rule to obtain a step-size that ensures sufficient descent *and* positive-definiteness of the next iterate. Thirdly, we use the form of the stationary condition characterizing the optimal solution to then *focus* the Newton direction computation on a small subset of *free* variables, in a manner that preserves the strong convergence guarantees of second-order descent.

Here is a brief outline of the paper. In Section 3, we present our algorithm that combines quadratic approximation, Newton's method and coordinate descent. In Section 4, we show that our algorithm is not only convergent but superlinearly so. We summarize the experimental results in Section 5, using real application data from [11] to compare the algorithms, as well as synthetic examples which reproduce experiments from [11]. We observe that our algorithm performs overwhelmingly better (quadratic instead of linear convergence) than the other solutions described in the literature.

2 Problem Setup

Let $\mathbf{y}$ be a p-variate Gaussian random vector, with distribution $\mathcal{N}(\mu, \Sigma)$. We are given n independently drawn samples $\{\mathbf{y_1}, \ldots, \mathbf{y_n}\}$ of this random vector, so that the sample covariance matrix can be written as

$$S = \frac{1}{n}\sum_{k=1}^{n}(\mathbf{y_k} - \hat{\mu})(\mathbf{y_k} - \hat{\mu})^T, \text{ where } \hat{\mu} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{y_i}. \tag{1}$$

Given some regularization penalty $\lambda > 0$, the ℓ_1 regularized Gaussian MLE for the inverse covariance matrix can be estimated by solving the following regularized *log-determinant* program:

$$\arg\min_{X \succ 0} \big\{ -\log\det X + \text{tr}(SX) + \lambda\|X\|_1 \big\} = \arg\min_{X \succ 0} f(X), \tag{2}$$

where $\|X\|_1 = \sum_{i,j=1}^{p} |X_{ij}|$ is the elementwise ℓ_1 norm of the $p \times p$ matrix X. Our results can be also extended to allow a regularization term of the form $\|\lambda \circ X\|_1 = \sum_{i,j=1}^{p} \lambda_{ij}|X_{ij}|$, i.e. different nonnegative weights can be assigned to different entries. This would include for instance the popular off-diagonal ℓ_1 regularization variant where we penalize $\sum_{i \neq j} |X_{ij}|$, but not the diagonal entries. The addition of such ℓ_1 regularization promotes sparsity in the inverse covariance matrix, and thus encourages sparse graphical model structure. For further details on the background of ℓ_1 regularization in the context of GMRFs, we refer the reader to [20, 2, 8, 15].

3 Quadratic Approximation Method

Our approach is based on computing iterative quadratic approximations to the regularized Gaussian MLE objective $f(X)$ in (2). This objective function f can be seen to comprise of two parts, $f(X) \equiv g(X) + h(X)$, where

$$g(X) = -\log\det X + \operatorname{tr}(SX) \text{ and } h(X) = \lambda\|X\|_1. \tag{3}$$

The first component $g(X)$ is twice differentiable, and strictly convex, while the second part $h(X)$ is convex but non-differentiable. Following the standard approach [17, 21] to building a quadratic approximation around any iterate X_t for such composite functions, we build the second-order Taylor expansion of the smooth component $g(X)$. The second-order expansion for the log-determinant function (see for instance [4, Chapter A.4.3]) is given by $\log\det(X_t + \Delta) \approx \log\det X_t + \operatorname{tr}(X_t^{-1}\Delta) - \frac{1}{2}\operatorname{tr}(X_t^{-1}\Delta X_t^{-1}\Delta)$. We introduce $W_t = X_t^{-1}$ and write the second-order approximation $\bar{g}_{X_t}(\Delta)$ to $g(X) = g(X_t + \Delta)$ as

$$\bar{g}_{X_t}(\Delta) = \operatorname{tr}((S - W_t)\Delta) + (1/2)\operatorname{tr}(W_t\Delta W_t\Delta) - \log\det X_t + \operatorname{tr}(SX_t). \tag{4}$$

We define the Newton direction D_t for the entire objective $f(X)$ can then be written as the solution of the regularized quadratic program:

$$D_t = \arg\min_{\Delta} \bar{g}_{X_t}(\Delta) + h(X_t + \Delta). \tag{5}$$

This Newton direction can be used to compute iterative estimates $\{X_t\}$ for solving the optimization problem in (2). In the sequel, we will detail three innovations which makes this resume feasible. Firstly, we provide an efficient method to compute the Newton direction. As in recent methods [12], we build on the observation that the Newton direction computation is a *Lasso* problem, and perform iterative coordinate descent to find its solution. However, the naive approach has an update cost of $O(p^2)$ for performing each coordinate descent update in the inner loop, which makes this resume infeasible for this problem. We show how a careful arrangement and caching of the computations can reduce this cost to $O(p)$. Secondly, we use an Armijo-rule based step size selection rule to obtain a step-size that ensures sufficient descent *and* positive-definiteness of the next iterate. Thirdly, we use the form of the stationary condition characterizing the optimal solution to then *focus* the Newton direction computation on a small subset of *free* variables, in a manner that preserves the strong convergence guarantees of second-order descent. We outline each of these three innovations in the following three subsections. We then detail the complete method in Section 3.4.

3.1 Computing the Newton Direction

The optimization problem in (5) is an ℓ_1 regularized least squares problem, also called *Lasso* [16]. It is straightforward to verify that for a symmetric matrix Δ we have $\operatorname{tr}(W_t\Delta W_t\Delta) = \operatorname{vec}(\Delta)^T(W_t \otimes W_t)\operatorname{vec}(\Delta)$, where $\otimes$ denotes the Kronecker product and $\operatorname{vec}(X)$ is the vectorized listing of the elements of matrix X.

In [7, 18] the authors show that coordinate descent methods are very efficient for solving lasso type problems. However, an obvious way to update each element of Δ to solve for the Newton direction in (5) needs $O(p^2)$ floating point operations since $Q := W_t \otimes W_t$ is a $p^2 \times p^2$ matrix, thus yielding an $O(p^4)$ procedure for approximating the Newton direction. As we show below, our implementation reduces the cost of one variable update to $O(p)$ by exploiting the structure of Q or in other words the specific form of the second order term $\operatorname{tr}(W_t\Delta W_t\Delta)$. Next, we discuss the details.

For notational simplicity we will omit the Newton iteration index t in the derivations that follow. (Hence, the notation for $\bar{g}_{X_t}$ is also simplified to $\bar{g}$.) Furthermore, we omit the use of a separate index for the coordinate descent updates. Thus, we simply use D to denote the current iterate approximating the Newton direction and use D' for the updated direction. Consider the coordinate descent update for the variable X_{ij}, with $i < j$ that preserves symmetry: $D' = D + \mu(\mathbf{e}_i\mathbf{e}_j^T + \mathbf{e}_j\mathbf{e}_i^T)$. The solution of the one-variable problem corresponding to (5) yields μ:

$$\arg\min_{\mu} \quad \bar{g}(D + \mu(\mathbf{e}_i\mathbf{e}_j^T + \mathbf{e}_j\mathbf{e}_i^T)) + 2\lambda|X_{ij} + D_{ij} + \mu|. \tag{6}$$

As a matter of notation: we use $\mathbf{x}_i$ to denote the i-th column of the matrix X. We expand the terms appearing in the definition of $\bar{g}$ after substituting $D' = D + \mu(\mathbf{e}_i\mathbf{e}_j^T + \mathbf{e}_j\mathbf{e}_i^T)$ for Δ in (4) and omit the terms not dependent on μ. The contribution of $\operatorname{tr}(SD') - \operatorname{tr}(WD')$ yields $2\mu(S_{ij} - W_{ij})$, while

the regularization term contributes $2\lambda|X_{ij} + D_{ij} + \mu|$, as seen from (6). The quadratic term can be rewritten using $\mathrm{tr}(AB) = \mathrm{tr}(BA)$ and the symmetry of D and W to yield:

$$\mathrm{tr}(WD'WD') = \mathrm{tr}(WDWD) \quad + \quad 4\mu\mathbf{w}_i^T D\mathbf{w}_j + 2\mu^2(W_{ij}^2 + W_{ii}W_{jj}). \tag{7}$$

In order to compute the single variable update we seek the minimum of the following function of μ:

$$\frac{1}{2}(W_{ij}^2 + W_{ii}W_{jj})\mu^2 + (S_{ij} - W_{ij} + \mathbf{w}_i^T D\mathbf{w}_j)\mu + \lambda|X_{ij} + D_{ij} + \mu|. \tag{8}$$

Letting $a = W_{ij}^2 + W_{ii}W_{jj}$, $b = S_{ij} - W_{ij} + \mathbf{w}_i^T D\mathbf{w}_j$, and $c = X_{ij} + D_{ij}$ the minimum is achieved for:

$$\mu = -c + \mathcal{S}(c - b/a, \lambda/a), \tag{9}$$

where $\mathcal{S}(z, r) = \mathrm{sign}(z)\max\{|z| - r, 0\}$ is the soft-thresholding function. The values of a and c are easy to compute. The main cost arises while computing the third term contributing to coefficient b, namely $\mathbf{w}_i^T D\mathbf{w}_j$. Direct computation requires $O(p^2)$ time. Instead, we maintain $U = DW$ by updating two rows of the matrix U for every variable update in D costing $O(p)$ flops, and then compute $\mathbf{w}_i^T\mathbf{u}_j$ using also $O(p)$ flops. Another way to view this arrangement is that we maintain a decomposition $WDW = \sum_{k=1}^{p}\mathbf{w}_k\mathbf{u}_k^T$ throughout the process by storing the $\mathbf{u}_k$ vectors, allowing $O(p)$ computation of update (9). In order to maintain the matrix U we also need to update two coordinates of each $\mathbf{u}_k$ when D_{ij} is modified. We can compactly write the row updates of U as follows: $\mathbf{u}_{i\cdot} \leftarrow \mathbf{u}_{i\cdot} + \mu\mathbf{w}_{j\cdot}$ and $\mathbf{u}_{j\cdot} \leftarrow \mathbf{u}_{j\cdot} + \mu\mathbf{w}_{i\cdot}$, where $\mathbf{u}_{i\cdot}$ refers to the i-th *row* vector of U.

We note that the calculation of the Newton direction can be simplified if X is a diagonal matrix. For instance, if we are starting from a diagonal matrix X_0, the terms $\mathbf{w}_i^T D\mathbf{w}_j$ equal $D_{ij}/((X_0)_{ii}(X_0)_{jj})$, which are independent of each other implying that we only need to update each variable according to (9) only once, and the resulting D will be the optimum of (5). Hence, the time cost of finding the first Newton direction is reduced from $O(p^3)$ to $O(p^2)$.

3.2 Computing the Step Size

Following the computation of the Newton direction D_t, we need to find a step size $\alpha \in (0, 1]$ that ensures positive definiteness of the next iterate $X_t + \alpha D_t$ and sufficient decrease in the objective function.

We adopt Armijo's rule [3, 17] and try step-sizes $\alpha \in \{\beta^0, \beta^1, \beta^2, \dots\}$ with a constant decrease rate $0 < \beta < 1$ (typically $\beta = 0.5$) until we find the smallest $k \in \mathbb{N}$ with $\alpha = \beta^k$ such that $X_t + \alpha D_t$ (a) is positive-definite, and (b) satisfies the following condition:

$$f(X_t + \alpha D_t) \leq f(X_t) + \alpha\sigma\Delta_t, \;\; \Delta_t = \mathrm{tr}(\nabla g(X_t)D_t) + \lambda\|X_t + D_t\|_1 - \lambda\|X_t\|_1 \tag{10}$$

where $0 < \sigma < 0.5$ is a constant. To verify positive definiteness, we use a Cholesky factorization costing $O(p^3)$ flops during the objective function evaluation to compute $\log\det(X_t + \alpha D_t)$ and this step dominates the computational cost in the step-size computations. In the Appendix in Lemma 9 we show that for any X_t and D_t, there exists a $\bar{\alpha}_t > 0$ such that (10) and the positive-definiteness of $X_t + \alpha D_t$ are satisfied for any $\alpha \in (0, \bar{\alpha}_t]$, so we can always find a step size satisfying (10) and the positive-definiteness even if we do not have the exact Newton direction. Following the line search and the Newton step update $X_{t+1} = X_t + \alpha D_t$ we efficiently compute $W_{t+1} = X_{t+1}^{-1}$ by reusing the Cholesky decomposition of X_{t+1}.

3.3 Identifying which variables to update

In this section, we propose a way to select which variables to update that uses the stationary condition of the Gaussian MLE problem. At the start of any outer loop computing the Newton direction, we partition the variables into *free* and *fixed* sets based on the value of the gradient. Specifically, we classify the $(X_t)_{ij}$ variable as *fixed* if $|\nabla_{ij}g(X_t)| < \lambda - \epsilon$ and $(X_t)_{ij} = 0$, where $\epsilon > 0$ is small. (We used $\epsilon = 0.01$ in our experiments.) The remaining variables then constitute the *free* set. The following lemma shows the property of the fixed set:

Lemma 1. *For any X_t and the corresponding* fixed *and* free *sets S_{fixed}, S_{free}, the optimized update on the* fixed *set would not change any of the coordinates. In other words, the solution of the following optimization problem is $\Delta = 0$:*

$$\arg\min_{\Delta} f(X_t + \Delta) \textit{ such that } \Delta_{ij} = 0 \;\; \forall (i, j) \in S_{free}.$$

The proof is given in Appendix 7.2.3. Based on the above observation, we perform the inner loop coordinate descent updates restricted to the free set only (to find the Newton direction). This reduces the number of variables over which we perform the coordinate descent from $O(p^2)$ to the number of non-zeros in X_t, which in general is much smaller than p^2 when λ is large and the solution is sparse. We have observed huge computational gains from this modification, and indeed in our main theorem we show the superlinear convergence rate for the algorithm that includes this heuristic.

The attractive facet of this modification is that it leverages the sparsity of the solution and intermediate iterates in a manner that falls within a block coordinate descent framework. Specifically, suppose as detailed above at any outer loop Newton iteration, we partition the variables into the fixed and free set, and then first perform a Newton update restricted to the fixed block, followed by a Newton update on the free block. According to Lemma 1 a Newton update restricted to the fixed block does not result in any changes.

In other words, performing the inner loop coordinate descent updates restricted to the free set is equivalent to two block Newton steps restricted to the fixed and free sets consecutively. Note further, that the union of the free and fixed sets is the set of all variables, which as we show in the convergence analysis in the appendix, is sufficient to ensure the convergence of the block Newton descent.

But would the size of free set be small? We initialize X_0 to the identity matrix, which is indeed sparse. As the following lemma shows, if the limit of the iterates (the solution of the optimization problem) is sparse, then after a *finite* number of iterations, the iterates X_t would also have the same sparsity pattern.

Lemma 2. *Assume $\{X_t\}$ converges to X^*. If for some index pair (i,j), $|\nabla_{ij} g(X^*)| < \lambda$ (so that $X^*_{ij} = 0$), then there exists a constant $\bar{t} > 0$ such that for all $t > \bar{t}$, the iterates X_t satisfy*

$$|\nabla_{ij} g(X_t)| < \lambda \text{ and } (X_t)_{ij} = 0. \tag{11}$$

The proof comes directly from Lemma 11 in the Appendix. Note that $|\nabla_{ij} g(X^*)| < \lambda$ implying $X^*_{ij} = 0$ follows from the optimality condition of (2). A similar (so called shrinking) strategy is used in SVM or ℓ_1-regularized logistic regression problems as mentioned in [19]. In Appendix 7.4 we show in experiments this strategy can reduce the size of variables very quickly.

3.4 The Quadratic Approximation based Method

We now have the machinery for a description of our algorithm QUIC standing for *QUadratic Inverse Covariance*. A high level summary of the algorithm is shown in Algorithm 1, while the the full details are given in Algorithm 2 in the Appendix.

Algorithm 1: Quadratic Approximation method for Sparse Inverse Covariance Learning (QUIC)

Input : Empirical covariance matrix S, scalar λ, initial X_0, inner stopping tolerance ϵ
Output: Sequence of X_t converging to $\arg\min_{X \succ 0} f(X)$, where
$f(X) = -\log\det X + \text{tr}(SX) + \lambda \|X\|_1$.

1 **for** $t = 0, 1, \ldots$ **do**
2 Compute $W_t = X_t^{-1}$.
3 Form the second order approximation $\bar{f}_{X_t}(\Delta) := \bar{g}_{X_t}(\Delta) + h(X_t + \Delta)$ to $f(X_t + \Delta)$.
4 Partition the variables into free and fixed sets based on the gradient, see Section 3.3.
5 Use coordinate descent to find the Newton direction $D_t = \arg\min_\Delta \bar{f}_{X_t}(X_t + \Delta)$ over the free variable set, see (6) and (9). (A *Lasso* problem.)
6 Use an *Armijo*-rule based step-size selection to get α s.t. $X_{t+1} = X_t + \alpha D_t$ is positive definite and the objective value sufficiently decreases, see (10).
7 **end**

4 Convergence Analysis

In this section, we show that our algorithm has strong convergence guarantees. Our first main result shows that our algorithm does converge to the optimum of (2). Our second result then shows that the asymptotic convergence rate is actually superlinear, specifically quadratic.

4.1 Convergence Guarantee

We build upon the convergence analysis in [17, 21] of the block coordinate gradient descent method applied to composite objectives. Specifically, [17, 21] consider iterative updates where at each

iteration t they update just a block of variables J_t. They then consider a Gauss-Seidel rule:

$$\bigcup_{j=0,\ldots,T-1} J_{t+j} \supseteq \mathcal{N} \ \ \forall t = 1, 2, \ldots, \tag{12}$$

where $\mathcal{N}$ is the set of all variables and T is a fixed number. Note that the condition (12) ensures that each block of variables will be updated at least once every T iterations. Our Newton steps with the free set modification is a special case of this framework: we set J_{2t}, J_{2t+1} to be the fixed and free sets respectively. As outlined in Section 3.3, our selection of the fixed sets ensures that a block update restricted to the fixed set would not change any values since these variables in fixed sets already satisfy the coordinatewise optimality condition. Thus, while our algorithm only explicitly updates the free set block, this is equivalent to updating variables in fixed and free blocks consecutively. We also have $J_{2t} \cup J_{2t+1} = \mathcal{N}$, implying the Gauss-Seidel rule with $T = 3$.

Further, the composite objectives in [17, 21] have the form $F(\mathbf{x}) = g(\mathbf{x}) + h(\mathbf{x})$, where $g(\mathbf{x})$ is smooth (continuously differentiable), and $h(\mathbf{x})$ is non-differentiable but separable. Note that in our case, the smooth component is the log-determinant function $g(X) = -\log\det X + \mathrm{tr}(SX)$, while the non-differentiable separable component is $h(\mathbf{x}) = \lambda\|\mathbf{x}\|_1$. However, [17, 21] impose the additional assumption that $g(\mathbf{x})$ is smooth over the domain R^n. In our case $g(\mathbf{x})$ is smooth over the restricted domain of the positive definite cone S^p_{++} . In the appendix, we extend the analysis so that convergence still holds under our setting. In particular, we prove the following theorem in Appendix 7.2:

Theorem 1. *In Algorithm 1, the sequence* $\{X_t\}$ *converges to the unique global optimum of* (2).

4.2 Asymptotic Convergence Rate

In addition to convergence, we further show that our algorithm has a quadratic asymptotic convergence rate.

Theorem 2. *Our algorithm* QUIC *converges quadratically, that is for some constant* $0 < \kappa < 1$*:*

$$\lim_{t\to\infty} \frac{\|X_{t+1} - X^*\|_F}{\|X_t - X^*\|_F^2} = \kappa.$$

The proof, given in Appendix 7.3, first shows that the step size as computed in Section 3.2 would eventually become equal to one, so that we would be eventually performing vanilla Newton updates. Further we use the fact that after a finite number of iterations, the sign pattern of the iterates converges to the sign pattern of the limit. From these two assertions, we build on the convergence rate result for constrained Newton methods in [6] to show that our method is quadratically convergent.

5 Experiments

In this section, we compare our method QUIC with other state-of-the-art methods on both synthetic and real datasets. We have implemented QUIC in C++, and all the experiments were executed on 2.83 GHz Xeon X5440 machines with 32G RAM and Linux OS.

We include the following algorithms in our comparisons:

- ALM: the Alternating Linearization Method proposed by [14]. We use their MATLAB source code for the experiments.
- GLASSO: the block coordinate descent method proposed by [8]. We used their Fortran code available from cran.r-project.org, version 1.3 released on 1/22/09.
- PSM: the Projected Subgradient Method proposed by [5]. We use the MATLAB source code available at `http://www.cs.ubc.ca/~schmidtm/Software/PQN.html`.
- SINCO: the greedy coordinate descent method proposed by [15]. The code can be downloaded from `https://projects.coin-or.org/OptiML/browser/trunk/sinco`.
- IPM: An inexact interior point method proposed by [11]. The source code can be downloaded from `http://www.math.nus.edu.sg/~mattohkc/Covsel-0.zip`.

Since some of the above implementations do not support the generalized regularization term $\|\lambda \circ X\|_1$, our comparisons use $\lambda\|X\|_1$ as the regularization term.

The GLASSO algorithm description in [8] does not clearly specify the stopping criterion for the Lasso iterations. Inspection of the available Fortran implementation has revealed that a separate

Table 1: The comparisons on synthetic datasets. p stands for dimension, $\|\Sigma^{-1}\|_0$ indicates the number of nonzeros in ground truth inverse covariance matrix, $\|X^*\|_0$ is the number of nonzeros in the solution, and ϵ is a specified relative error of objective value. $*$ indicates the run time exceeds our time limit 30,000 seconds (8.3 hours). The results show that QUIC is overwhelmingly faster than other methods, and is the only one which is able to scale up to solve problem where $p = 10000$.

Dataset setting			Parameter setting			Time (in seconds)					
pattern	p	$\|\Sigma^{-1}\|_0$	λ	$\|X^*\|_0$	ϵ	QUIC	ALM	Glasso	PSM	IPM	Sinco
chain	1000	2998	0.4	3028	10^{-2}	**0.30**	18.89	23.28	15.59	86.32	120.0
					10^{-6}	**2.26**	41.85	45.1	34.91	151.2	520.8
chain	4000	11998	0.4	11998	10^{-2}	**11.28**	922	1068	567.9	3458	5246
					10^{-6}	**53.51**	1734	2119	1258	5754	*
chain	10000	29998	0.4	29998	10^{-2}	**216.7**	13820	*	8450	*	*
					10^{-6}	**986.6**	28190	*	19251	*	*
random	1000	10758	0.12	10414	10^{-2}	**0.52**	42.34	10.31	20.16	71.62	60.75
					10^{-6}	**1.2**	28250	20.43	59.89	116.7	683.3
			0.075	55830	10^{-2}	**1.17**	65.64	17.96	23.53	78.27	576.0
					10^{-6}	**6.87**	*	60.61	91.7	145.8	4449
random	4000	41112	0.08	41910	10^{-2}	**23.25**	1429	1052	1479	4928	7375
					10^{-6}	**160.2**	*	2561	4232	8097	*
			0.05	247444	10^{-2}	**65.57**	*	3328	2963	5621	*
					10^{-6}	**478.8**	*	8356	9541	13650	*
random	10000	91410	0.08	89652	10^{-2}	**337.7**	26270	21298	*	*	*
					10^{-6}	**1125**	*	*	*	*	*
			0.04	392786	10^{-2}	**803.5**	*	*	*	*	*
					10^{-6}	**2951**	*	*	*	*	*

threshold is computed and is used for these inner iterations. We found that under certain conditions the threshold computed is smaller than the machine precision and as a result the overall algorithm occasionally displayed erratic convergence behavior and slow performance. We modified the Fortran implementation of GLASSO to correct this error.

5.1 Comparisons on synthetic datasets

We first compare the run times of the different methods on synthetic data. We generate the two following types of graph structures for the underlying Gaussian Markov Random Fields:

- Chain Graphs: The ground truth inverse covariance matrix Σ^{-1} is set to be $\Sigma^{-1}_{i,i-1} = -0.5$ and $\Sigma^{-1}_{i,i} = 1.25$.
- Graphs with Random Sparsity Structures: We use the procedure mentioned in Example 1 in [11] to generate inverse covariance matrices with random non-zero patterns. Specifically, we first generate a sparse matrix U with nonzero elements equal to ± 1, set Σ^{-1} to be $U^T U$ and then add a diagonal term to ensure Σ^{-1} is positive definite. We control the number of nonzeros in U so that the resulting Σ^{-1} has approximately $10p$ nonzero elements.

Given the inverse covariance matrix Σ^{-1}, we draw a limited number, $n = p/2$ i.i.d. samples, to simulate the high-dimensional setting, from the corresponding GMRF distribution. We then compare the algorithms listed above when run on these samples.

We can use the minimum-norm sub-gradient defined in Lemma 5 in Appendix 7.2 as the stopping condition, and computing it is easy because X^{-1} is available in QUIC. Table 1 shows the results for timing comparisons in the synthetic datasets. We vary the dimensionality from $1000, 4000$ to 10000 for each dataset. For chain graphs, we select λ so that the solution had the (approximately) correct number of nonzero elements. To test the performance of algorithms on different parameters (λ), for random sparse pattern we test the speed under two values of λ, one discovers correct number of nonzero elements, and one discovers 5 times the number of nonzero elements. We report the time for each algorithm to achieve ϵ-accurate solution defined by $f(X^k) - f(X^*) < \epsilon f(X^*)$. Table 1 shows the results for $\epsilon = 10^{-2}$ and 10^{-6}, where $\epsilon = 10^{-2}$ tests the ability for an algorithm to get a

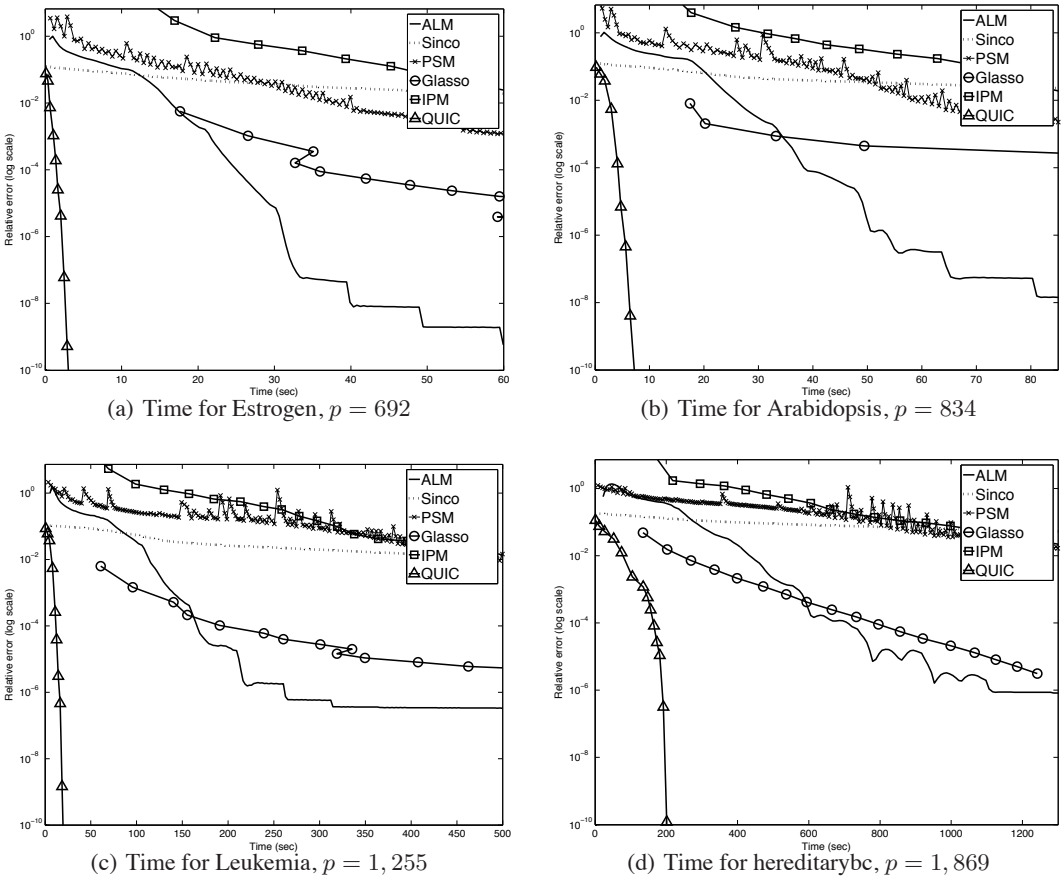
good initial guess (the nonzero structure), and $\epsilon = 10^{-6}$ tests whether an algorithm can achieve an accurate solution. Table 1 shows that QUIC is consistently and overwhelmingly faster than other methods, both initially with $\epsilon = 10^{-2}$, and at $\epsilon = 10^{-6}$. Moreover, for $p = 10000$ random pattern, there are $p^2 = 100$ million variables, the selection of fixed/free sets helps QUIC to focus only on very small part of variables, and can achieve an accurate solution in about 15 minutes, while other methods fails to even have an initial guess within 8 hours. Notice that our λ setting is smaller than [14] because here we focus on the λ which discovers true structure, therefore the comparison between ALM and PSM are different from [14].

5.2 Experiments on real datasets

We use the real world biology datasets preprocessed by [11] to compare the performance of our method with other state-of-the-art methods. The regularization parameter λ is set to 0.5 according to the experimental setting in [11]. Results on the following datasets are shown in Figure 1: Estrogen ($p = 692$), Arabidopsis ($p = 834$), Leukemia ($p = 1,225$), Hereditary ($p = 1,869$). We plot the relative error $(f(X_t) - f(X^*))/f(X^*)$ (on a log scale) against time in seconds. On these real datasets, QUIC can be seen to achieve super-linear convergence, while other methods have at most a linear convergence rate. Overall QUIC can be ten times faster than other methods, and even more faster when higher accuracy is desired.

6 Acknowledgements

We would like to thank Professor Kim-Chuan Toh for providing the data set and the IPM code. We would also like to thank Professor Katya Scheinberg and Shiqian Ma for providing the ALM implementation. This research was supported by NSF grant IIS-1018426 and CCF-0728879. ISD acknowledges support from the Moncrief Grand Challenge Award.



(a) Time for Estrogen, $p = 692$

(b) Time for Arabidopsis, $p = 834$

(c) Time for Leukemia, $p = 1,255$

(d) Time for hereditarybc, $p = 1,869$

Figure 1: Comparison of algorithms on real datasets. The results show QUIC converges faster than other methods.

References

[1] A. Agarwal, S. Negahban, and M. Wainwright. Convergence rates of gradient methods for high-dimensional statistical recovery. In *NIPS*, 2010.

[2] O. Banerjee, L. E. Ghaoui, and A. d'Aspremont. Model selection through sparse maximum likelihood estimation for multivariate Gaussian or binary data. *The Journal of Machine Learning Research*, 9, 6 2008.

[3] D. Bertsekas. *Nonlinear programming*. Athena Scientific, Belmont, MA, 1995.

[4] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 7th printing edition, 2009.

[5] J. Duchi, S. Gould, and D. Koller. Projected subgradient methods for learning sparse Gaussians. *UAI*, 2008.

[6] J. Dunn. Newton's method and the Goldstein step-length rule for constrained minimization problems. *SIAM J. Control and Optimization*, 18(6):659–674, 1980.

[7] J. Friedman, T. Hastie, H. Höfling, and R. Tibshirani. Pathwise coordinate optimization. *Annals of Applied Statistics*, 1(2):302–332, 2007.

[8] J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, July 2008.

[9] J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010.

[10] E. S. Levitin and B. T. Polyak. Constrained minimization methods. *U.S.S.R. Computational Math. and Math. Phys.*, 6:1–50, 1966.

[11] L. Li and K.-C. Toh. An inexact interior point method for l1-reguarlized sparse covariance selection. *Mathematical Programming Computation*, 2:291–315, 2010.

[12] L. Meier, S. Van de Geer, and P. Bühlmann. The group lasso for logistic regression. *Journal of the Royal Statistical Society, Series B*, 70:53–71, 2008.

[13] R. T. Rockafellar. *Convex Analysis*. Princeton University Press, Princeton, NJ, 1970.

[14] K. Scheinberg, S. Ma, and D. Glodfarb. Sparse inverse covariance selection via alternating linearization methods. *NIPS*, 2010.

[15] K. Scheinberg and I. Rish. Learning sparse Gaussian Markov networks using a greedy coordinate ascent approach. In J. Balczar, F. Bonchi, A. Gionis, and M. Sebag, editors, *Machine Learning and Knowledge Discovery in Databases*, volume 6323 of *Lecture Notes in Computer Science*, pages 196–212. Springer Berlin / Heidelberg, 2010.

[16] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society*, 58:267–288, 1996.

[17] P. Tseng and S. Yun. A coordinate gradient descent method for nonsmooth separable minimization. *Mathematical Programming*, 117:387–423, 2007.

[18] T. T. Wu and K. Lange. Coordinate descent algorithms for lasso penalized regression. *The Annals of Applied Statistics*, 2(1):224–244, 2008.

[19] G.-X. Yuan, K.-W. Chang, C.-J. Hsieh, and C.-J. Lin. A comparison of optimization methods and software for large-scale l1-regularized linear classification. *Journal of Machine Learning Research*, 11:3183–3234, 2010.

[20] M. Yuan and Y. Lin. Model selection and estimation in the gaussian graphical model. *Biometrika*, 94:19–35, 2007.

[21] S. Yun and K.-C. Toh. A coordinate gradient descent method for l1-regularized convex minimization. *Computational Optimizations and Applications*, 48(2):273–307, 2011.