Learning Sparse Gaussian Markov Networks using a Greedy Coordinate Ascent Approach

Katya Scheinberg[1] and Irina Rish[2]

[1] IEOR Department, Columbia University, New York, NY
[2] IBM T.J. Watson Research Center, Yorktown Heights, NY

Abstract. In this paper, we introduce a simple but efficient greedy algorithm, called *SINCO*, for the Sparse INverse COvariance selection problem, which is equivalent to learning a sparse Gaussian Markov Network, and empirically investigate the structure-recovery properties of the algorithm. Our approach is based on a coordinate ascent method which naturally preserves the sparsity of the network structure. We show that SINCO is often comparable to, and, in various cases, outperforms commonly used approaches such as *glasso* [7] and COVSEL [1], in terms of both structure-reconstruction error (particularly, false positive error) and computational time. Moreover, our method has the advantage of being easily parallelizable. Finally, we show that SINCO's greedy nature allows reproduction of the regularization path behavior by applying the method to one (sufficiently small) instance of the regularization parameter λ only; thus, SINCO can obtain a desired number of network links directly, without having to tune the λ parameter. We evaluate our method empirically on various simulated networks and real-life data from biological and neuroimaging applications.

1 Introduction

In many practical applications of statistical learning the objective is not simply to construct an accurate predictive model but rather to discover meaningful interactions among the variables. For example, in applications such as reverse-engineering of gene networks, discovery of functional brain connectivity patterns, or analysis of social interactions, the main focus is on reconstructing the network structure representing dependencies among multiple variables, such as genes, brain areas, or individuals. Probabilistic graphical models, such as Markov networks, provide a statistical tool that captures such variable interactions explicitly in a form of a graph.

Herein, we focus on learning sparse Markov Networks over Gaussian random variables (also called Gaussian Markov Random Fields, or GMRFs), which is equivalent to reconstructing the inverse covariance (concentration, or precision) matrix C, assuming the data are centered to have zero mean. Following the parsimony principle, our objective is to choose the simplest model, i.e. the sparsest network (matrix) that adequately explains the data. This sparsity requirement not only improves the interpretability of the model, but also serves as a regularizer that helps to avoid overfitting.

The sparse inverse covariance selection problem, first introduced in [4], is to find the maximum-likelihood model with a constraint on the number of parameters (i.e.,

small l_0-norm of C). In general, this is an intractable combinatorial problem. Early approaches used greedy forward or backward search that required $O(p^2)$ maximum-likelihood-estimation (MLE) fits for different models in order to add (delete) an edge [8], where p is the number of variables. This approach does not scale well with the number of variables[3]; moreover, the existence of MLE for C is not even guaranteed when the number of variables exceeds the number of observations [3].

Recently, however, an alternative approximation approach to the above problem was suggested in [21, 1] that replaces the intractable l_0 constraint with its l_1-relaxation, known to enforce sparsity, and yields a convex optimization problem that can be solved efficiently. A variety of algorithms for solving this problem were proposed in the past few years [21, 1, 7, 14, 5, 15, 9][4].

In this paper, we introduce a very simple algorithm for solving the above l_1-regularized maximum-likelihood problem, and provide a convergence proof. Our algorithm, called SINCO (for Sparse INverse COvariance), solves the primal problem, unlike most of its predecessors that focus on the dual (e.g. COVSEL [1], *glasso* [7], as well as [5]). SINCO uses coordinate ascent, in a greedy manner, optimizing one diagonal or two symmetric off-diagonal elements of C at each step, unlike, for example, COVSEL or *glasso* which optimize one row (column) of the dual matrix. Thus, SINCO naturally preserves the sparsity of the solution and tends to avoid introducing unnecessary (small) nonzero elements, which appears to be beneficial when the "ground-truth" structure is sufficiently sparse.

Note that, although the current state-of-art algorithms for the above problem are converging to the same optimal solution in the limit, the near-optimal solutions obtained after any fixed number of iterations can be different structure-wise, even though they reach similar precision in the objective function reconstruction. Indeed, it is well-known that similar likelihoods can be obtained by two distributions with quite different structures due to multiple (sufficiently) weak links. As to the l_1-norm regularization, although it often tends to enforce solution sparsity, it is still only an approximation to l_0 (i.e. a sparse solution may have same l_1-norm as a much denser one). Adding l_1-norm penalty is only guaranteed to recover the "ground-truth" model under certain condition on the data (that are not always satisfied in practice) and for certain asymptotic growth regimes of the regularization parameter, with growing number of samples n and dimensions p (with unknown constant). So the optimal solution, as well as near-solutions at given precision, could possibly include false positives, and one optimization method can potentially choose sparser near-solutions (at same precision) than another method.

Thus, especially in case of sufficiently sparse ground-truth models, a method such as SINCO may be preferable, since it is more "cautious" about adding nonzero elements than its competitors (i.e., it adds at most two nonzero elements at a time, which are also providing the maximum improvement in the objective function - i.e., the method selects, in a sense, the "most important" edges first). Indeed, as demonstrated by our empirical results, SINCO has a better capability of reducing the false-positive error rate (while

[3] E.g., [11] reported difficulties running "the forward selection MLE for more than thirty nodes in the graph".

[4] Moreover, recent extensions of this approach impose additional structure on the graph, allowing, for example, to learn blockwise-sparse models [5, 15, 10].

maintaining a similar true positive rate) when compared to *glasso*, a commonly used method that we choose as a baseline here (together with the similar but less efficient *COVSEL* method), since it is the only other method that maintains the initial sparsity of solution in a controlled manner.

Another property of SINCO is that evaluating each candidate edge can be performed very efficiently, in constant time, by solving a quadratic equation. In terms of the overall computational time, while *glasso* is comparable to, or faster than SINCO for a relatively small number of variables p, SINCO appears to have much better scaling when p increases (e.g., gets closer to 1000 variables), and can significantly outperform *glasso* (and, of course, COVSEL). Moreover, SINCO has the advantage of being easily parallelizable due to the nature of its greedy steps. While we are not claiming SINCO's computational superiority to all state-of-art methods in the *sequential setting* (it is known that the recently proposed projected gradient [5] and smooth optimization [9] methods outperform *glasso* which is comparable to SINCO), we must underscore that straightforward massive parallelization appears to be SINCO's unique property, as none of its competitors seem to be parallelizable, at least not in such an easy way. In particular, *glasso* solves a sequence of Lasso problems, each of which is solved using sequential coordinate descent, which does not gain from parallelization. The gradient-based methods of [5] and [9] require an eigenvalue factorization or a matrix inverse. These operations, while parallelizable, do not scale as efficiently as simple arithmetic operations involved in SINCO's computations.

Next, we investigate empirically the "path-building" property of SINCO. Note that the structure reconstruction accuracy is known to be quite sensitive to the choice of the regularization parameter λ, and the problem of selecting the "best" value of this parameter in practical settings remains open. (As mentioned before, recent theoretical work has focused mainly on asymptotic consistency results [11, 21, 1, 18, 13].) Thus, we explore SINCO vs *glasso* behavior in several regimes of λ. What we observe is that SINCO's greedy approach introduces "important" nonzero elements in a manner similar to the path-construction process based on sequentially reducing the value of λ. SINCO can reproduce the regularization path behavior without actually varying the value of the regularization parameter, following instead the "greedy solution path", i.e. sequentially introducing non-zero elements. We observe such behavior on both synthetic problems and real-life biological networks, such as E.coli transcriptional network from the DREAM-07 challenge [16]. This behavior is somewhat similar to LARS [6] for Lasso, however, unlike LARS, SINCO updates the coordinates which provide the best optimal function value improvement, rather than the largest gradient component.

Finally, experiments on real-life brain imaging (fMRI) data demonstrate that SINCO reconstructs Markov Networks that achieve the same or better classification accuracy than its competitors while using much smaller fraction of edges (non-zero entries of the inverse-covariance matrix). In summary, the advantages of SINCO include (1) simplicity, (2) natural tendency to preserve sparsity (beneficial on sufficiently sparse problems), (3) efficiency and a relatively straightforward massive parallelization, as well as (4) an interesting property associated with its solution path.

2 Problem Formulation

We consider a multivariate Gaussian probability density function over a set of p random variables $X = \{X_1, ..., X_p\}$ with the covariance matrix Σ and zero mean. A Markov network (also called a Markov Random Field, or MRF) represents the conditional independence structure of a joint distribution, where a missing edge (i, j) implies conditional independence between X_i and X_j given all remaining variables [8]. In Gaussian MRFs, missing edges correspond to zero entries in the inverse covariance (concentration) matrix $C = \Sigma^{-1}$, and vice versa [8]. Thus, learning the structure of a Gaussian MRF is equivalent to recovering the zero-pattern of the corresponding inverse-covariance matrix. Note that the straightforward approach of just taking the inverse of the empirical covariance matrix $A = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i^T \mathbf{x}_i$, where $\mathbf{x}_i$ is the i-th sample, $i = 1, ..., n$ (i.e., the inverse of the maximum-likelihood estimate of the covariance matrix Σ), does not produce the desired result. Indeed, even if the inverse exists (which is not necessarily the case when $p >> n$), it does not typically contain any elements that are exactly zero. Therefore, an explicit sparsity-enforcing constraint needs to be added to the maximum-likelihood formulation.

A common approach to enforcing sparsity of C is to include as a penalty the (vector) l_1-norm of C, which is equivalent to imposing Laplace priors on the elements of C in the maximum-likelihood framework [21, 7, 1, 5, 15] (see [21] for the derivation details). The standard approach assumes that all entries of C follow the same Laplace distribution with a common parameter λ, i.e. $p(C_{ij}) = \frac{\lambda_{ij}}{2} e^{-\lambda_{ij}|C_{ij}|}$, yielding the following penalized log-likelihood maximization problem [21, 1, 7].

$$\max_{C \succ 0} \frac{n}{2}[\ln\det(C) - \mathrm{tr}(AC)] - \lambda||C||_1. \tag{1}$$

Herein, we make a more general assumption about $p(C)$, allowing different elements of C to have different parameters λ_{ij} (as, for example, in [5]). Hence we consider the following formulation

$$\max_{C \succ 0} \frac{n}{2}[\ln\det(C) - \mathrm{tr}(AC)] - \|C\|_S. \tag{2}$$

Here by $\|C\|_S$ we denote the sum of absolute values of the elements of the matrix $S \cdot C$, where $\cdot$ denotes the element-wise product. For example, if S is a product of $\rho = \frac{n}{2}\lambda$ and the matrix of all ones, then the problem reduces to the standard problem in the eq. 1. The dual of this problem is

$$\max_{W \succ 0}\{\frac{n}{2}\ln\det(W) - np/2 : \text{ s.t. } -S \leq \frac{n}{2}(W - A) \leq S\}, \tag{3}$$

where the inequalities involving matrices W, A and S are element-wise. The optimality conditions for this pair of primal and dual problems imply that $W = C^{-1}$ and that $(n/2)W_{ij} - A_{ij} = S_{ij}$ if $C_{ij} > 0$ and $(n/2)W_{ij} - A_{ij} = -S_{ij}$ if $C_{ij} < 0$.

3 The SINCO Method

3.1 Relation to Prior Art

Problem (2) is a special case of a semidefinite programming problem (SDP) [20], which can be solved in polynomial time by interior point methods (IPM). However, each iteration requires $O(p^6)$ time and $O(p^4)$ space, which is very costly. Another reason

why using IPMs is less desirable for the structure recovery problem is that the sparsity pattern is recovered only in the limit, i.e., the solution does not typically include exact zeros, and thus numerical inaccuracy can potentially interfere with the structure recovery.

As an alternative to IPMs, several more efficient approaches were developed recently for problem (2). Most of those approaches are primarily focused on solving the dual problem in (3). For example, [1] and [7] apply the block-coordinate descent method to the dual formulation, [9] uses a first-order optimal gradient ascent approach, and [5] uses a projected gradient approach.

Herein, we propose a novel algorithm, called SINCO, for Sparse INverse COvariance problem. SINCO solves the primal problem directly and uses coordinate ascent, which naturally preserves the sparsity of the solution. Unlike, for example, COVSEL and *glasso* that optimize one row (and the corresponding symmetric column) of the dual matrix is at each step, SINCO only optimizes one diagonal or two (symmetric) off-diagonal entries of the matrix C at each step. The advantage of our approach is that the solution to each subproblem is available in closed form as a root of a quadratic equation. Computation at each step requires a constant number of arithmetic operations, independent of p. Hence, in $O(p^2)$ operations a potential step can be computed for *all pairs* of symmetric elements (i.e., for all pairs (i, j)). Then the step which provides the *best* function value improvement can be chosen, which is the essence of the greedy nature of our approach. Once the step is taken, the update of the gradient information requires $O(p^2)$ operations. Hence, overall, each iteration takes $O(p^2)$ operations. As we will see later, each step is also suitable for massive parallelization.

In comparison, *glasso* and COVSEL require solving a quadratic programming problem when updating a row (column)[5], and its theoretical and empirical complexity varies depending on the method used, but always exceeds $O(p^2)$: it is $O(p^4)$ for COVSEL and $O(p^3)$ for *glasso*. Also, the methods of [14] and [5] iterate through the columns and require $O(p^4)$ and $O(p^3)$ time per iteration, respectively (see [5] for detailed discussion). Note, however, that the overall number of iterations can be potentially lower than in the case of SINCO, since the above methods update each row (column) at once. We will mainly focus on comparing our method with *glasso* as a representative state-of-the art technique, particularly since it is the only other method that maintains the initial sparsity of the solution in a controlled manner. (In some cases, when we only compare the accuracy of the solution (Section 4.4), we perform experiments with COVSEL, a similar but less efficient implementation of the same approach as *glasso*).

As we will show in our numerical experiments, SINCO, in a serial mode, is comparable to or faster than *glasso*, which is orders of magnitude faster than COVSEL [7]. Also, SINCO often reaches lower false-positive error than *glasso* since it introduces nonzero elements greedily. Perhaps the most interesting consequence of SINCO's greedy nature is that it reproduces the regularization path behavior while using only one value of the regularization parameter λ (see Section 4.1). Another important feature of SINCO is the ability to efficiently utilize warm starts in various modes. For instance, it

[5] COVSEL solves the subproblems via an interior point approach (as second order cone quadratic problems (SOCP)), while *glasso* poses the subproblem as a dual of the Lasso problem [17], which is solved by coordinate descent method.

is easy to compute a range of solutions for various values of λ, which defines matrix S.

3.2 Algorithm Description

The main idea of the method is the following: at each iteration, the matrix C is updated by changing one element on the diagonal or two symmetric off-diagonal elements. This implies that the updated C can be written at $C + \theta(e_i e_j^T + e_j e_i^T)$, where i and j are the indices corresponding to the elements that are being changed. We can therefore rewrite the objective function of the problem (2) as a function of θ (denoted $f(\theta)$ below). The key observation is that, given the matrix $W = C^{-1}$, the exact line search that optimizes $f(\theta)$ along the direction $e_i e_j^T + e_j e_i^T$ reduces to a solution of a quadratic equation. Hence each such line search takes a constant number of operations. Moreover, given the starting objective value, the new function value on each step can be computed in a constant number of steps. This means that we can perform such line search for all (i, j) pairs in $O(p^2)$ time, which is linear in the number of unknown variables C_{ij}. We then can choose the step that gives the best improvement in the value of the objective function. After the step is chosen, the dual matrix $W = C^{-1}$ and, hence, the objective function gradient, are updated in $O(p^2)$ operations[6].

We now present the method. First, we can reformulate the problem (2) as:

$$\begin{aligned} &\max_{C', C''} \ \frac{n}{2}[\ln\det(C' - C'') - \mathrm{tr}(A(C' - C''))] - ||C' - C''||_S, \\ &\text{s. t.} \ \ C' \geq 0,\ C'' \geq 0,\ C' - C'' \succ 0 \end{aligned}$$

Note that $||C' - C''||_S = tr(S(C' + C''))$ if C' and C'' have non-overlapping nonzero structure.

For a fixed pair (i, j), we consider the update of C' given by $C'(\theta) = C' + \theta(e_i e_j^T + e_j e_i^T)$, such that $C' \geq 0$. Then we can write the objective as the function of θ:

$$f'(\theta) = \frac{n}{2}(\ln\det(C + \theta e_i e_j^T + \theta e_j e_i^T) - \mathrm{tr}(A(C + \theta e_i e_j^T + \theta e_j e_i^T)) - ||C + \theta e_i e_j^T + \theta e_j e_i^T||_S$$

Similarly, if we consider the update of the form $C''(\theta) = C'' + \theta(e_i e_j^T + e_j e_i^T)$ such that $C'' > 0$, the objective function becomes

$$f''(\theta) = \frac{n}{2}(\ln\det(C - \theta e_i e_j^T - \theta e_j e_i^T) - \mathrm{tr}(A(C - \theta e_i e_j^T - \theta e_j e_i^T)) - ||C - \theta e_i e_j^T - \theta e_j e_i^T||_S$$

The method we propose works as follows:

[6] Note that there is no need to enforce the posdef constraint explicitly, as $\ln\det(C)$ goes to negative infinity when C approaches singularity. At each step, we maximize the objective along the direction of increase until the local maximum is reached. Hence, it is impossible for the method to move past the point where the objective is negative infinity.

Algorithm 1

0. Initialize $C' = I,\ C'' = 0,\ W = I$
1. Form the gradient $G' = \frac{n}{2}(W - A) - S$ and $G'' = -S - \frac{n}{2}(W + A)$
2. For each pair (i, j) such that
   $(i) G'_{ij} > 0, C''_{ij} = 0,$ compute the maximum of $f'(\theta)$ for $\theta > 0$.
   $(ii) G'_{ij} < 0, C'_{ij} > 0,$ compute the maximum of $f'(\theta)$ for $\theta < 0$ subject to $\mathrm{C}' \geq 0$.
   $(iii) G''_{ij} > 0, C'_{ij} = 0,$ compute the maximum of $f''(\theta)$ for $\theta > 0$.
   $(iv) G''_{ij} < 0, C''_{ij} > 0,$ compute the maximum of $f''(\theta)$ for $\theta < 0$ subject to $\mathrm{C}'' \geq 0$.
3. Choose the step which provides the maximum function improvement.
   If relative function improvement is below tolerance, then Exit.
4. Update W^{-1} and the function value and repeat.

The inverse W, then, is updated, according to the Sherman-Morrison-Woodbury formula $(X + ab^T)^{-1} = X^{-1} - X^{-1}a(1 + b^T X^{-1}a)^{-1}b^T X^{-1}$ in $O(p^2)$ operations. The following theorem is the result of the analysis presented in Appendix.

Theorem 1. *The steps of SINCO algorithm are well-defined (that is, the quadratic equation always yields the maximum of $f(C)$ along the chosen direction). The algorithm converges to the unique optimal solution of (2).*

Note that the algorithm lends itself readily to massive parallelization. Indeed, at each iteration of the algorithm the step computation for each (i, j) pair can be parallelized and the procedure that updates W involves simply adding to each element of W a function that involves only two rows of W (see Appendix for details). Hence the updates can be also done in parallel and in very large scale cases the matrix W can also be stored in a distributed manner. The same is true for the storage of matrices A and S (assuming that S needs to be stored, that is not all elements of S are the same), while the best way to store C' and C'' matrices may be in sparse form.

4 Empirical Evaluation

In order to test structure-reconstruction accuracy, we first performed experiments on several types of synthetic problems. Note that, unlike prediction of an observed variable, structure reconstruction accuracy is harder to test on "real" data since (1) the "true" structure may not be available and (2) known links in "real" networks (e.g., known gene networks) may not necessarily correspond to links in the underlying Markov net. We generated uniform-random, as well as semi-realistic, structured "scale-free" networks, that follow a power-law degree distribution; such networks are known to model well various biological, ecological, social, and other real-life networks [2]. The scale-free (SF) networks were generated using the preferential attachment (Barabasi-Albert) model [2] [7]

[7] We used the open-source Matlab code available at small http://www.mathworks.com/matlabcentral/fileexchange/11947.

We generated networks with various density, measured by the $\%$ of non-zero off-diagonal entries. For each density level, we generated the networks over p variables, that defined the structure of the "ground-truth" inverse covariance matrix, and for each of them, we generate matrices with random covariances corresponding to the non-diagonal non-zero entries (while maintaining positive definiteness of the resulting covariance matrix). We then sampled n instances, with the value of n depending on the experiment, from the corresponding multivariate Gaussian distribution over p variables.

We also experimented with several real-life datasets, including (a) microarray data for the genome-scale transcriptional network of E.coli (a DREAM-2007 challenge [16]), and (b) the brain activity data from a set of fMRI experiments described in [12][8].

We used $\epsilon = 10^{-6}$ threshold on the improvement in the objective function as a stopping criterion.

4.1 Regularization Path

One of the main challenges in sparse inverse covariance selection is the proper choice of the weight matrix S in (2). Typically the matrix S is chosen to be a multiple of the matrix of all ones. The multiplying coefficient is denoted by λ and is called the "regularization parameter". Hence the norm $\|C\|_S$ in (2) reduces to $\lambda\|C\|_1$ (in the vector-norm sense) as in ([1]). Clearly, for large values of λ as $\lambda \to \infty$ the solution to (2) is likely to be very sparse and eventually diagonal, which means that no structure recovery is achieved. On the other hand, if λ is small as $\lambda \to 0$, the solution C is likely to be dense and eventually approach A^{-1}, and, again, no structure recovery occurs. Hence exploration of a regularization path is an integral part of the sparse inverse covariance selection.

The SINCO method is very well-suited for the efficient regularization path computation, since it directly exploits warm starts. When λ is relatively large, a very sparse solution can be obtained quickly. This solution can be used as a warm start to the problem with a smaller value of λ and, if the new value of λ is not much smaller than the previous value, then the new solution is typically obtained in just a few iterations, because the new solution has only a few extra nonzero elements. Warm starts can also be used to initiate different subproblems for the leave-one-out validation approach, where the structure learning is performed on n subsets of the data (one sample being left out each time), so that the stability of the solution can be evaluated. Since each leave-one-out subproblem differs from another one by a rank-two update of matrix A, and since the resulting nonzero pattern is expected to be not very different, the solution to one subproblem can be an efficient warm start for another subproblem.

Typically the output of the regularization path is evaluated via the ROC curves showing the trade-off between the number of true positive (TP) element recovered and the number of false positive (FP) elements. Producing better curves (where the number of TPs rises fast relative to FPs) is usually an objective of any method that does not focus on specific λ selection. An interesting property of SINCO is that it introduces nonzero entries to the matrix C as it progresses. Hence, if we use looser tolerance and stop the algorithm early, then we will observe fewer nonzero entries, hence a sparse solution for any specific value of λ. What we observe, as seen in Figure 1, is that if we apply

[8] For more details, see the StarPlus website http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-81/www/.

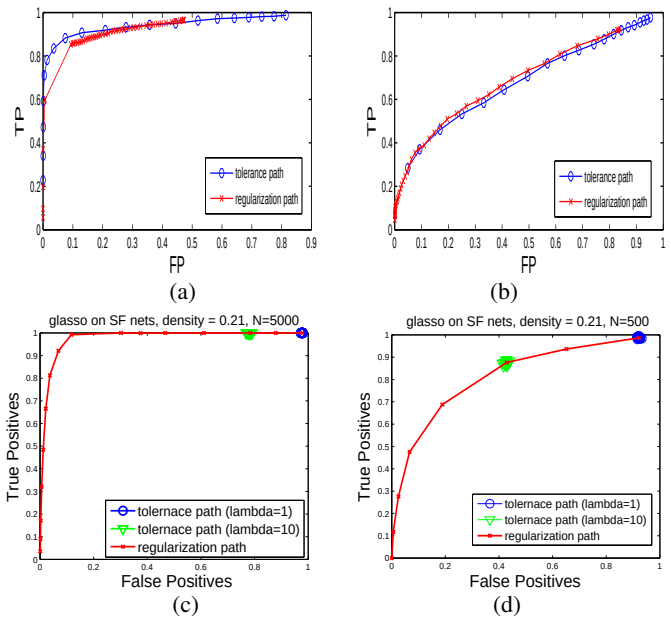
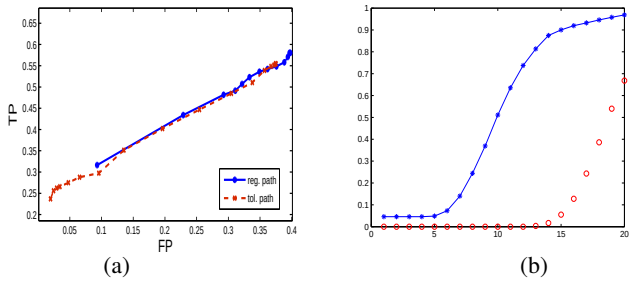


Fig. 1. Scale-free networks: SINCO and *glasso* paths when varying tolerance (tolerance path - blue 'o') and λ (lambda, or regularization, path - red 'x'). (a) and (b) show SINCO paths, on problems with (a) $p = 100$, $n = 5000$ and (b) $p = 100$, $n = 500$, respectively; (c) and (d) show *glasso* paths on the same problems.

SINCO to problem (2) with ever tighter tolerance (equivalent to observing the path of intermediate solutions) then the ROC curves obtained from the tolerance solution path match the ROC curves obtained from the regularization path. Here we show examples of the matching ROC curves for various networks with which we experimented. We use a randomly generated structured (scale-free) network that is 21% dense and a randomly generated unstructured network, 3% dense (due to space restriction, we only show the results for scale-free networks; random unstructured networks produce very similar results). We use $p = 100$ and two instances: $n = 500$ and $n = 5000$. We applied SINCO to one instance of problem (2) with $\lambda = 0.01$ (very small regularization) with a range of stopping tolerances from 10^{-4} to 10^{-7}. The ROC curve of that path is presented by a line with "o"s. We also applied SINCO with fixed tolerance of 10^{-6} to a range of λ values from 300 to 0.01. The corresponding ROC curves are denoted by lines with "x"s. We can see that the ROC curve of the regularization path for the given range of values of λ is somewhat less steep than that of the tolerance path, but the curves are still very close in the area where they overlap. For baseline we also present the ROC curve of the regularization path computed by *glasso*, which is very similar to the SINCO's ROC curves. Note that changing tolerance does not have the same effect on *glasso* as it does on SINCO. The number of TP and FP does not change noticeably with increasing tolerance. This is due to the fact that the algorithm in *glasso* updates a whole row and a



(a) (b)

Fig. 2. SINCO's tolerance path vs regularization path: (a) two path for the E.coli subnetwork; (b) comparing positives on scale-free networks.

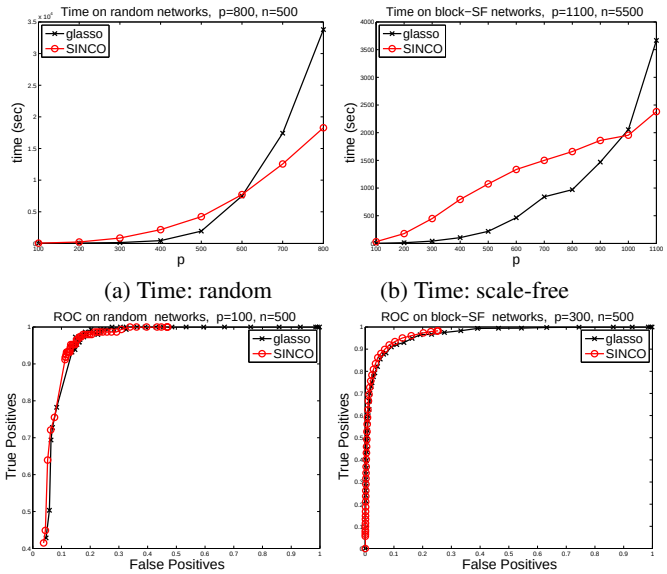
column of C at each iteration while it cycles through the rows and columns, rather than selecting the updates in a greedy manner.

Figure 2a shows a very similar behavior when comparing the two paths for the DREAM-07 challenge problem of E.coli transcriptional network reconstruction, for $n = 300$ microarray samples and a subset of $p = 133$ transcription factors that form a connected component in the graph.

Note, however, that it is not always the case that SINCO's solution path (tolerance path) is actually the same as the regularization path. In Figure 2b, the lower curve shows the percentage of the positives (nonzero entries in C) in solutions from SINCO's tolerance path which are *not* present in the solution on the regularization path. The higher curve represents the percentage of true positives; the x axis of the figure represent the points along the tolerance and regularization paths, which are matched to each other. We observe that the SINCO and the regularization paths largely coincide until the TP reach its maximum and further nonzeros are in the FP category and hence, in a way, are random noise.

Our observations imply that SINCO can be used to greedily select the elements of graphical model until the desired trade-off between FPs and TPs or the desired number of nonzero elements is achieved or the allocated CPU time is exhausted. In the limit SINCO solves the same problem as *glasso* and hence the limit number of the true and false positives is dictated by the choice of λ. But since the real goal is to recover the true nonzero structure of the covariance matrix, it is not necessary to solve problem (2) accurately. For the purpose of recovering a good TP/FP ratio one can apply the SINCO method, without the adjustments to λ.

We should note that computing the regularization path presented in our experiments is typically more efficient in terms of CPU time than computing the tolerance path; the largest computational cost lies in computing the tail of the path for smaller tolerances. On the other hand, the tolerance path appears to be more precise and exhaustive, in terms of possible FP/TP tradeoffs. It is also important to note that the entire tolerance path is automatically produced as a sequence of iterates produced by SINCO, while the regularization path can only be computed as a sequence of solutions for a given set of values of λ.



(a) Time: random (b) Time: scale-free

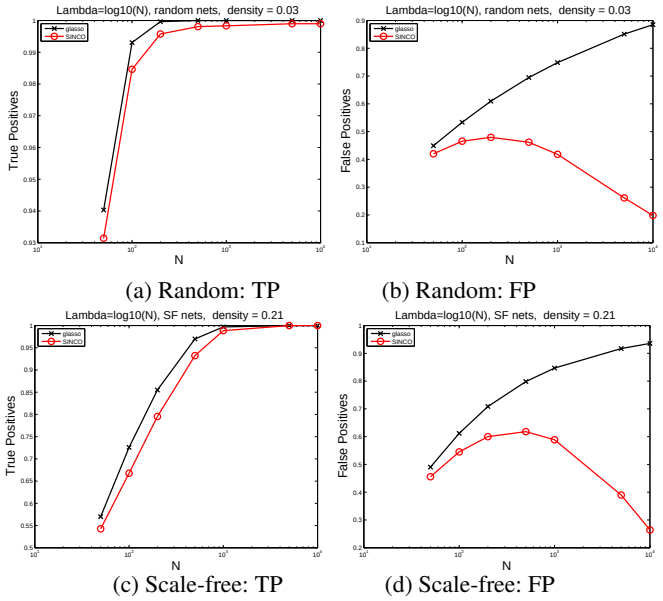
(c) ROC: random (d) ROC: scale-free

Fig. 3. CPU time: SINCO vs *glasso* on (a) random networks ($n = 500$, fixed range of λ) and (b) scale-free networks (density 21%, n and λ scaled by the same factor with p, $n = 500$ for $p = 100$). ROC curves: SINCO vs *glasso* on (c) random networks ($p = 100$, $n = 500$, fixed range of λ) and (d) scale-free networks ($p = 300$, $n = 1500$, density 21%, n and λ scaled by the same factor with p, starting with $n = 500$ for $p = 100$).

4.2 Empirical Complexity

Here we will discuss the empirical dependence of the runtime of the SINCO algorithm on the choice of stopping tolerance and the problem size p. We also investigate the effect increasing p has on the results produced by SINCO and *glasso*. Both methods were executed on Intel Core 2Duo T7700 processor (2.40GHz); note, however, that *glasso* is based on well-tuned Fortran code with an R interface, while SINCO a straight-forward C++ implementation of the algorithm in Section 3 with Matlab interface.

We consider the situation when p increases. If together with p the number of nonzeros in the true inverse covariance also increases, then to obtain a comparable problem we need to increase n accordingly. Increasing n, in turn, affects the contribution of λ, since the problem scaling changes. Here we chose to consider the following two simple settings, where we can account for these effects. In the first setting, we increase p while keeping the number of the off-diagonal nonzero elements in the randomly generated unstructured network constant (around 300). We do not, therefore, increase n or λ. The CPU time necessary to compute the entire path for λ ranging from 300 to 0.01 is plotted for $p = 100, 200, 300, 500, 800$ and 1000 in Figure 3a. In the second case, we generated block-diagonal matrices of sizes $p = 100, 200, 300, 500, 800, 1000$, with 100×100 diagonal blocks, each of which equals the inverse covariance matrix of a 21%-dense structured (scale-free) network from the previous section. Since the number of nonzero



(a) Random: TP (b) Random: FP

(c) Scale-free: TP (d) Scale-free: FP

Fig. 4. SINCO and *glasso* accuracy with growing n: (a) and (b) show the results averaged over 20 random networks ($p = 100$, density 3%), (c) and (d) show similar results averaged over 25 scale-free networks ($p = 100$, density 21%).

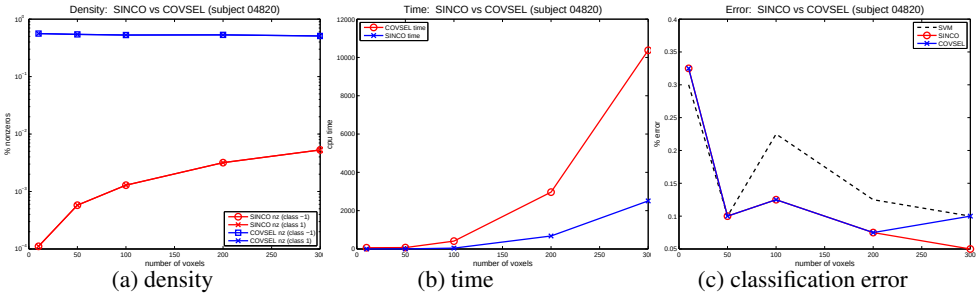
elements grows linearly with p, we increased n and the appropriate range of λ linearly as well. The CPU time for this case is shown in the last plot of Figure 3b.

The first two plots in Figure 3 shows that the CPU time (in seconds) for SINCO scales up more slowly than that of *glasso*, with increasing number of variables, p. The reason for the difference in scaling rates is evident in the ROC curves shown in Figures 3c and 3d, which demonstrate that, for similarly high true-positive rate, *glasso* tends to have much higher false-positive rate than SINCO, thus producing a less sparse solution, overall.

4.3 Asymptotic behavior with increasing λ

Finally, we investigate the behavior of SINCO for a fixed value of p as n grows. In this setting, we expect to obtain larger TP values and smaller FP error with increasing n. The consistency result in [21] suggests that for our formulation, to obtain an accurate asymptotic structure recovery, we should pick λ that grows with n, but so that its growth is slower than $\sqrt{n}$.

Here we use $\lambda = \log_{10}(n)$. We again apply our algorithm and *glasso* to the 21%-dense scale-free networks with $p = 100$. In Figure 4 we show the how the value of TP and FP returned by the two algorithms changes with growing n (note that λ is kept fixed for each value of n). We observe that SINCO achieves in the limit nearly 0% false-positive error and nearly 100% true-positive rate, while *glasso*'s FP error grows with increasing n. This result is, again, a consequence of the greedy approach utilized by SINCO.



(a) density (b) time (c) classification error

Fig. 5. SINCO vs. COVSEL on fMRI data.

4.4 Application to fMRI analysis

Here we describe the results of applying SINCO to a real-life data, where the "ground truth" network structure was not available; thus, we could not measure the structure reconstruction accuracy, and instead evaluated the prediction accuracy of the resulting Markov networks. In this section, we compare SINCO with COVSEL [1] rather than *glasso*, since COVSEL is the other available Matlab implementation solving the same dual problem as *glasso* (as opposed to SINCO solving the primal one), and, although SINCO was shown to be slower than *glasso*[7], the objective here was rather to compare the prediction accuracy of the two approaches and the density of solutions.

We used fMRI data for the mind-state prediction problem described in [12][9]. The data consists of a series of trials in which the subject is being shown either a picture $(+1)$ or a sentence (-1). Our dataset consists of 1700 to 2200 features, dependent on a particular subject, and 40 samples, where half of the samples correspond to the picture stimulus (+1) and the remaining half correspond to sentence stimulus (-1). (One sample corresponds to the averaged fMRI image over 6 scans sequentially taken while a subject is presented with a particular stimulus). We used leave-one-out cross-validation, and report average results over 40 cross-validation folds, each corresponding to one sample left out for testing, and the remaining 39 used for training.

For each class $Y = \{-1, 1\}$, we learn a sparse Markov Net model that provides us with an estimate of the Gaussian conditional density $p(\mathbf{x}|y)$, where $\mathbf{x}$ is the feature (voxel) vector; on the test data, we choose the most-likely class label $\arg\max_y p(\mathbf{x}|y)P(y)$ for each unlabeled test sample $\mathbf{x}$.

Figure 5 show the results of comparing SINCO versus COVSEL for one of the subjects in the above study (similar results were obtained for two more subjects). We observe that SINCO produces classifiers that are equally (or more) accurate that those produced by COVSEL (Figure 5c), but is much faster (Figure 5b) and uses much sparser Markov Net models (Figure 5a), which suggests that COVSEL (and hence *glasso*, since they are different implementations of the same approach) learns many links that are not essential for the discriminative ability of the classifier[10].

[9] For more details, see the StarPlus website http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-81/www/.

[10] It is also interesting to note that both Markov Net classifiers are competitive with, and often more accurate than the (linear) SVM classifier (Figure 5c). Herein, we used the SVM code by A. Schwaighofer available at http://ida.first.fraunhofer.de/~anton/software.html.

5 Discussion

We proposes a novel approach, called SINCO, for solving the sparse inverse-covariance selection problem, which is equivalent to learning the structure (and parameters) of a Gaussian MRF. Our method is very simple; it uses greedy coordinate ascent, efficiently performing each evaluation step in a constant time, by solving a quadratic equation. We also provide a convergence proof. The method we present has two major advantages: (1) natural tendency to preserve the sparsity of solution, leading to better true-positive vs. false-positive error rate trade-off, especially on sparse problems, and (2) potential for a straightforward massive parallelization that could provide a significant $O(p^2)$ speedup at each iteration. Also, our method has interesting (and useful) property of replicating the regularization path *behavior* (although not necessarily replicating the actual regularization path) by applying the method to one (sufficiently small) instance of the regularization parameter λ only. Thus, a desired number of network links can be obtained directly from the greedy solution path, without having to tune the λ parameter. SINCO properties are evaluated on a range of randomly generated problems, as well as on two real-life applications including gene-network reconstruction and neuroimaging. A important direction for future work is a more detailed investigation of the near-solution space of the sparse inverse covariance problem considered herein, and a better characterization of the relation between the objective function near its optimum and the variance in the structure of potential solutions.

Appendix

Herein we present the derivation of the SINCO algorithm.

As mentioned in Section 3.2, the maximum of the one-dimensional function in Step 3 of SINCO is available in closed form. Indeed, consider the step $\bar{C}' = C' + \theta(e_i e_j^T + e_j e_i^T)$. Let us assume that $\theta > 0$ and that $C'_{ij} = 0$, which implies that we can write the step as $\bar{C} = C + \theta(e_i e_j^T + e_j e_i^T)$, since the (i,j) and (j,i) elements do not become zero for any such step.

The inverse W, then, is updated, according to the Sherman-Morrison-Woodbury formula $(X + ab^T)^{-1} = X^{-1} - X^{-1}a(1 + b^T X^{-1}a)^{-1}b^T X^{-1}$, as follows:

$$\bar{W} = W - \theta(\kappa_1 W_i W_j^T + \kappa_2 W_i W_i^T + \kappa_3 W_j W_j^T + \kappa_1 W_j W_i^T),$$

$$\kappa_1 = -(1 + \theta W_{ij})/\kappa, \;\; \kappa_2 = \theta W_{jj}/\kappa, \;\; \kappa_3 = \theta W_{ii}/\kappa,$$

$$\kappa = \theta^2(W_{ii} * W_{jj} - W_{ij}^2) - 1 - 2\theta W_{ij}.$$

Let us now compute the objective function as the function of θ:

$$f(\theta) = \tfrac{n}{2}(\ln\det(C + \theta e_i e_j^T + \theta e_j e_i^T) - \mathrm{tr}(A(C + \theta e_i e_j^T + \theta e_j e_i^T)) - ||C + \theta e_i e_j^T + \theta e_j e_i^T||_S.$$

We use the following property of the determinant:

$$\det(X + ab^T) = det(X)(1 + b^T X^{-1}a)$$

and the Scherman-Morisson-Woodbury formula. We have

$$\det(C + \theta e_i e_j^T + \theta e_j e_i^T) = \det(C + \theta e_j e_i^T)(1 + \theta e_j^T (C + \theta e_j e_i^T)^{-1} e_i) =$$

$$\det(C)(1 + \theta e_i^T C^{-1} e_j)(1 + \theta e_j^T C^{-1} e_i - \theta^2 e_j^T C^{-1} e_j (1 + \theta e_i^T C^{-1} e_j)^{-1} e_i^T C^{-1} e_i) =$$

$$\det(C)(1 + 2\theta e_i^T C^{-1} e_j + (\theta e_j^T C^{-1} e_i)^2 - \theta^2 e_i^T C^{-1} e_i e_j^T C^{-1} e_j).$$

Given the dual solution $W = C^{-1}$, and recalling that W and A are symmetric, but S is not necessarily so, we can

write the above as

$$\det(C + \theta e_i e_j^T + \theta e_j e_i^T) = \det(C)(1 + 2\theta W_{ij} + \theta^2(W_{ij}^2 - W_{ii}W_{jj})).$$

Then the change in the objective function is

$$f(\theta) - f = \frac{n}{2}(\ln(1 + 2\theta W_{ij} + \theta^2(W_{ij}^2 - W_{ii}W_{jj})) - 2A_{ij}\theta) - S_{ij}\theta - S_{ji}\theta,$$

the last term being derived from the fact that $C_{ij} + \theta$ and $C_{ji} + \theta$ remain positive. Let us now consider the derivative of the objective function with respect to θ

$$\frac{df(\theta)}{d\theta} = \frac{nW_{ij} + n\theta(W_{ij}^2 - W_{ii}W_{jj})}{\theta^2(W_{ij}^2 - W_{ii}W_{jj}) + 1 + 2\theta W_{ij}} - nA_{ij} - S_{ij} - S_{ji}.$$

To find the maximum of $f(\theta)$ we need to find $\theta > 0$ for which $\frac{df(\theta)}{d\theta} = 0$. Letting a denote $W_{ii}W_{jj} - W_{ij}^2$, this condition can be written as:

$$nW_{ij} - nA_{ij} - S_{ij} - S_{ji} - (na + 2W_{ij}(nA_{ij} + S_{ij} + S_{ji})\theta + a(nA_{ij} + S_{ij} + S_{ji})\theta^2 = 0.$$

To find the value of θ for which the derivative of the objective function equals zero we need to solve the above quadratic equation

$$ab\theta^2 - (na + 2W_{ij}b)\theta + nW_{ij} - b = 0, \tag{4}$$

where $a = W_{ii}W_{jj} - W_{ij}^2$ and $b = nA_{ij} + S_{ij} + S_{ji}$. Notice that a is always nonnegative, because matrix W is positive definite, and it equals zero only when $i = j$. We know that at $\theta = 0$ $\frac{df(\theta)}{d\theta} > 0$. Let us investigate what happens when θ grows. The discriminant of the quadratic equation is

$$D = (na + 2W_{ij}b)^2 - 4ab(nW_{ij} - b) = (na)^2 + 4nW_{ij}ab + 4W_{ij}^2b^2 - 4abnW_{ij} + 4ab^2$$

$$= (na)^2 + 4b^2W_{ii}W_{jj} > 0,$$

hence the quadratic equation always has a solution. At $\theta = 0$ the quadratic function equals

$$nW_{ij} - nA_{ij} - S_{ij} - S_{ji} = G'_{ij} + G'_{ji} > 0.$$

Now let us again consider the derivative $\frac{df(\theta)}{d\theta}$. At $\theta = 0$ we know that the derivative is positive. We also know that the denominator

$$\theta^2(W_{ij}^2 - W_{ii}W_{jj}) + 1 + 2\theta W_{ij} = (1 + \theta W_{ij})^2 - \theta^2 W_{ii}W_{jj}$$

is positive when $\theta = 0$ and is equal to zero when $\theta = \theta_{max} = 1/(\sqrt{W_{ii}W_{jj}} - W_{ij}) > 0$. The function $f(\theta)$ approaches negative infinity when $\theta \to \theta_{max}$, hence so does $\frac{df(\theta)}{d\theta}$. This implies that $\frac{df(\theta)}{d\theta}$ has to reach the value zero for some $\theta \in (0, \theta_{max})$. Hence the quadratic equation (4) has one positive solution in this interval. This solution gives us the maximum of $f(\theta)$ and hence the length of the step along the direction $e_i e_j^T + e_j e_i^T$.

The objective function value is easy to update using the formula

$$det(C' - C'' + \theta(e_i e_j^T + e_j e_i^T)) = det(C' - C'')(1 + 2\theta W_{ij} - \theta^2 a)).$$

Let us consider the negative step along the direction $e_i e_j^T + e_j e_i^T$ when $C'_{ij} > 0$. The derivations are exactly as above, except for we are now looking for solution $\theta < 0$. As discussed above, the term under the logarithm

$$\theta^2(W_{ij}^2 - W_{ii}W_{jj}) + 1 + 2\theta W_{ij} = (1 + \theta W_{ij})^2 - \theta^2 W_{ii}W_{jj}$$

is positive when $\theta = 0$ and is also equal to zero when $\theta = \theta_{min} = -1/(\sqrt{W_{ii}W_{jj}} + W_{ij}) < 0$. The derivative of $f(\theta)$ at $\theta = 0$ is negative, this time (which is why we are considering a negative step, in the first place), which means that there exists a $\theta \in (\theta_{min}, 0)$ for which this derivative is zero, hence the quadratic equation (4) has a negative solution. This negative solution $\theta_- < 0$ determines the length of the step in the direction $-e_i e_j^T - e_j e_i^T$. It is important to note that the length of the step cannot exceed the value C'_{ij}, hence the final step length is computed as $\max(\theta_-, -C'_{ij})$.

The other two possible steps listed in Step 3 can be analyzed analogously, the main difference being the sign before the terms nA_{ij}, S_{ij} and S_{ji} in the case of the step that updates C''.

Each step can be computed by a constant number of arithmetic operations, hence to find the step that provides the largest function value improvement it takes $O(p^2)$ operations - the same amount of work (up to a constant) that it takes to update W and the gradient after one iteration. Hence the overall per-iteration complexity is $O(p^2)$. Moreover, this algorithms lends itself readily to massive parallelization, as discussed earlier in Section 3.2.

The convergence of the method follows from the convergence of a block-coordinate descent method on a strictly convex objective function. The only constraints are box constraints (nonnegativity) and they do not hinder the convergence. In fact

we can view our method as a special case of the row by row (RBR) method for SDP described in [19]. In the case of SINCO we extensively use the fact that each coordinate descent step is cheap and, unlike the RBR algorithm, we select the next step based on the best function value improvement. On the other hand, we maintain the inverse matrix W, which RBR method does not. However, none of these differences prevent the convergence result for RBR in [19] to apply to our method. Hence the convergence to the optimal solution holds for SINCO.

References

1. O. Banerjee, L. El Ghaoui, and A. d'Aspremont. Model selection through sparse maximum likelihood estimation for multivariate gaussian or binary data. *Journal of Machine Learning Research*, 9:485–516, March 2008.
2. A.-L. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.
3. S.L. Buhl. On the existence of maximum likelihood estimators for graphical gaussian models. *Scandinavian Journal of Statistics*, 20(3):263–270, 1993.
4. A. P. Dempster. Covariance selection. *Biometrics*, 28(1):157–175, March 1972.
5. J. Duchi, S. Gould, and D. Koller. Projected subgradient methods for learning sparse gaussians. In *Proc. of UAI-08*, 2008.
6. B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Ann. Statist.*, 32(1):407–499, 2004.
7. J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 2007.
8. S. Lauritzen. *Graphical Models*. Oxford University Press, 1996.
9. Z. Lu. Smooth optimization approach for sparse covariance selection. *SIAM Journal on Optimization*, (19(4)):1807–1827, 2009.
10. B. Marlin and K. Murphy. Sparse gaussian graphical models with unknown block structure. In *Proc. of ICML-09*, 2009.
11. N. Meinshausen and P. Buhlmann. High dimensional graphs and variable selection with the Lasso. *Annals of Statistics*, 34(3):1436–1462, 2006.
12. T.M. Mitchell, R. Hutchinson, R.S. Niculescu, F. Pereira, X. Wang, M. Just, and S. Newman. Learning to decode cognitive states from brain images. *Machine Learning*, 57:145–175, 2004.
13. P. Ravikumar, M. Wainwright, G. Raskutti, and B. Yu. Model selection in Gaussian graphical models: High-dimensional consistency of l1-regularized MLE. In *NIPS-08*. 2008.
14. A. Rothman, P. Bickel, E. Levina, and J. Zhu. Sparse permutation invariant covariance estimation. *Electronic Journal of Statistics*, (2):494–515, 2008.
15. M. Schmidt, E. van den Berg, M. Friedlander, and K. Murphy. Optimizing costly functions with simple constraints: A limited-memory projected quasi-newton algorithm. In *Proc. of AISTATS-09*, 2009.
16. G. Stolovitzky, R.J. Prill, and A. Califano. Lessons from the dream2 challenges. *Annals of the New York Academy of Sciences*, (1158):159–95, 2009.
17. R. Tibshirani. Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society, Series B*, 58(1):267–288, 1996.
18. M. Wainwright, P. Ravikumar, and J. Lafferty. High-Dimensional Graphical Model Selection Using ℓ_1-Regularized Logistic Regression. In *NIPS 19*, pages 1465–1472. 2007.
19. Z. Wen, D. Goldfarb, S. Ma, and K. Scheinberg. Row by row methods for semidefinite programming. Technical report, Department of IEOR, Columbia University, 2009.
20. H. Wolkowicz, R. Saigal, and eds. L. Vanenberghe. *Handbook of Semidefinite Programming*. Kluwer Academic Publoshers, 2000.
21. M. Yuan and Y. Lin. Model Selection and Estimation in the Gaussian Graphical Model. *Biometrika*, 94(1):19–35, 2007.