A Multilevel Framework for Sparse Inverse Covariance Estimation

Eran Treister*
Computer Science, Technion, Israel
and Earth and Ocean Sciences, UBC
Vancouver, BC, V6T 1Z2, Canada
eran@cs.technion.ac.il

Javier S. Turek, Irad Yavneh
Department of Computer Science
Technion, Israel Institute of Technology
Technion City, Haifa 32000, Israel
{javiert,irad}@cs.technion.ac.il

Abstract

We consider the problem of estimating the inverse of a covariance matrix of a normal distribution, assuming that it is sparse. To this end, an l_1 regularized log-determinant optimization problem is solved. We present a multilevel framework for accelerating existing solvers of this problem. Taking advantage of the sparseness of the matrix, we create a multilevel hierarchy of similar problems, which are traversed in order to accelerate the optimization process. Our numerical experiments demonstrate the efficiency of the multilevel framework for solving both medium and large scale instances of this problem.

1 Introduction

Estimating the parameters of a multivariate Gaussian (Normal) distribution is a common problem in many applications in machine learning, computational biology, and others [1]. Given a set of samples $\{\mathbf{x}_i\}_{i=1}^m \in \mathbb{R}^n$ where $\mathbf{x}_i \sim \mathcal{N}(\mu, \Sigma)$, the objective is to estimate the mean $\mu \in \mathbb{R}^n$, and either the covariance matrix $\Sigma \in \mathbb{R}^{n\times n}$ or its inverse Σ^{-1}. Both the mean μ and the covariance Σ are often estimated using the maximum likelihood estimator (MLE). The MLE is obtained by maximizing the Gaussian probability density function, which immediately leads to the estimation $\hat{\mu} = \frac{1}{m}\sum_{i=0}^m \mathbf{x}_i$ for the mean. Following this, the estimation for Σ^{-1} is obtained by solving

$$\min_{A\succ 0} f(A) = \min_{A\succ 0} -\log(\det A) + \mathrm{tr}(SA), \tag{1}$$

where $S = \frac{1}{m}\sum_{i=0}^m (\mathbf{x}_i - \hat{\mu})(\mathbf{x}_i - \hat{\mu})^T$. This leads to the covariance MLE estimation $\hat{\Sigma}^{\text{MLE}} = S$, which is also called the empirical covariance matrix.

However, if the number of samples is smaller than the problem dimension, i.e., $m < n$, then S in (1) is rank deficient, whereas the true $\mathbf{\Sigma}$ is assumed to be positive definite, hence full-rank. Still, one can reasonably estimate $\mathbf{\Sigma}^{-1}$ assuming that it is sparse [4]. For this purpose, we follow [2, 1, 3], and minimize (1) with a sparsity-promoting l_1 prior:

$$\min_{A\succ 0}\ F(A) = \min_{A\succ 0}\ f(A) + \lambda\|A\|_1, \tag{2}$$

where $f(A)$ is the smooth and convex MLE functional in (1), $\|A\|_1 \equiv \sum_{i,j}|a_{ij}|$, and $\lambda > 0$ is a regularization parameter. The function $F(A)$ in (2) is non-smooth and convex. Many methods were recently developed for solving (2)—see [2, 3, 5, 6, 7, 8, 9, 11, 12, 15] and references therein.

In this work we introduce a multilevel framework for accelerating existing solvers for (2). Our approach extends the approach of [14] which introduced this idea for the solution of the LASSO

*Eran Treister is grateful to the Azrieli Foundation for the award of an Azrieli Fellowship. This research was funded (in part) by the Intel Collaborative Research Institute for Computational Intelligence (ICRI-CI).

problem (l_1 regularized least squares minimization). In this approach, the convergence of common iterative methods is accelerated using a nested hierarchy of smaller versions of the problem. Using the sparsity of the matrix A, the dimension of the problem (2) is reduced by ignoring ostensibly irrelevant variables in the matrix, which are kept zero. That is, each reduced problem is defined by (2), restricted to a specially chosen subset of variables, resulting in a nested hierarchy of problems. The algorithm then performs sub-space correcting iterations using existing methods over each of the low dimensional problems in turn, that aim to activate the variables that comprise the support of a true minimizer. Under suitable conditions, this algorithm converges to the global minimizer of (2).

2 Iterative proximal Newton methods for sparse inverse covariance estimation

The current state-of-the-art, [7, 8, 9, 12, 15], involve a "proximal Newton" approach [16], where only the smooth part $f(\mathbf{A})$ in (2) is replaced by its quadratic approximation to obtain the Newton descent direction. The non-smooth l_1 term remains intact and the resulting Newton problem is an l_1 regularized quadratic minimization, also known as LASSO [13]. To obtain this quadratic approximation, the gradient and Hessian of $f(A)$ are needed, and are given by

$$\nabla f(A) = S - A^{-1}, \qquad \nabla^2 f(A) = A^{-1} \otimes A^{-1}, \tag{3}$$

where $\otimes$ is the Kronecker product. Given an iterate matrix $A^{(k)}$, the Newton direction Δ problem is given by

$$\min_{\Delta \in \mathbb{R}^{n \times n}} f(A^{(k)}) + \text{tr}((S - W)\Delta) + \frac{1}{2}\text{tr}(\Delta W \Delta W) + \lambda \|A^{(k)} + \Delta\|_1, \tag{4}$$

where $W = (A^{(k)})^{-1}$ is the inverse of the k-th iterate. This problem may be solved using an iterative "shrinkage" method for the LASSO problem [17]. After obtaining a solution to (4), the iterated solution is updated using a step-length obtained by linesearch. This guarantees the positive definiteness of $A^{(k+1)}$ and ensures the sufficient reduction in the objective of (2).

Restricting to an active set. In addition, a crucial step is to restrict the minimization of (4) at each iteration to an "active set" of variables and keep the rest as zeros [9, 12]. The active set of a matrix A is defined as

$$active(A) = \left\{(i,j) : A_{ij} \neq 0 \vee |(S - A^{-1})_{ij}| > \lambda\right\}. \tag{5}$$

If one solves (4) with respect only to the variables outside this active set, they all remain zero, suggesting that it is worthwhile to (temporarily) restrict (4) only to the variables in this active set [9]. This reduces the computational complexity of the LASSO solvers: given the matrix W, the Hessian term in (4) can be calculated in $O(Kn)$ operations instead of $O(n^3)$, where $K = |active\left(A^{(k)}\right)|$. This holds for any method for solving the LASSO problem. Algorithm 1 summarizes a generic proximal Newton method for solving (2). We note that this algorithm is only suitable for small-scale problems, because it assumes that the inverse of a sparse matrix, which is typically a dense $n \times n$ matrix, fits into memory. Out of all existing methods, only [8, 15] are suitable for large-scale problems. [8] generally follows Algorithm 1, while [15] is different, but follows related ideas.

Algorithm: $A^{(k+1)} \leftarrow$ **ProxNewtonInvCov**$(A^{(k)})$
Invert the matrix $W = (A^k)^{-1}$, and define $active(A^{(k)})$.
Calculate the direction Δ by solving (4) restricted to $active(A^{(k)})$.
Define $A^{(k+1)} = A^{(k)} + \alpha\Delta$ by linesearch

Algorithm 1: Proximal Newton iteration for the sparse inverse covariance estimation.

3 A Multilevel acceleration for sparse inverse covariance estimation

Given an iterate $A^{(k)}$, Algorithm 1 solves the Newton problem (4) while considering only the variables in $active(A^{(k)})$. This saves a significant amount of computations. However, in [9] it is shown

that if $A^{(k)}$ is far from the optimal solution A^*, then $|active(A^{(k)})|$ may be several times larger than number of non-zeros in the solution A^*, since the entries of the gradient of $A^{(k)}$ are typically large. This may lead to a rather dense $A^{(k+1)}$, which imposes a larger and more difficult Newton problem, requiring extensive computations. This phenomenon of rather dense iterates is typical for most methods. As the iterates progress, the number of non zeros in the matrices reduces, until they converge to that of A^*. If we knew the non-zeros of A^*, solving (2) would require less computations.

We next describe our new multilevel framework for solving (2), which aims to save computations by incrementally choosing the non-zeros of the iterates and preventing dense matrices. At each multilevel iteration, called a "ML-cycle", we define a hierarchy of reduced problems, referred to as *coarse* problems. Each coarse problem is defined by (2), restricted to a specially chosen subset of variables. In each ML-cycle we traverse the entire hierarchy of levels, from the coarsest to the finest, applying an iteration of some method similar to Algorithm 1 on each of the reduced problems. We henceforth refer to such an iteration as "relaxation". We iteratively repeat these ML-cycles, monotonically reducing the objective of (2), until some convergence criterion is satisfied.

Define the reduced problem by limiting problem (2) to a subset of entries, denoted by $\mathcal{C}$

$$\min_{A\succ 0} \; F_{\mathcal{C}}(A) = \min_{A\succ 0,\; supp(A)\in\mathcal{C}} \; f(A) + \lambda\|A\|_1, \tag{6}$$

where $supp(A) = \{(i,j) : (A)_{ij} \neq 0\}$ is the support of A—the set of non-zero entries of A. To solve problem (6) using Algorithm 1, for example, one may restrict the Newton direction to $active(A) \cap \mathcal{C}$ instead of $active(A)$.

At any iteration k, in order for $A^{(k)}$ to be a feasible point of (6), we require that $supp(A^{(k)}) \subseteq \mathcal{C}$, which also implies $F_{\mathcal{C}}(A^{(k)}) = F(A^{(k)})$. Since Algorithm 1 limits the Newton direction to the active set, we impose $\mathcal{C} \subseteq active(A^{(k)})$. Following these guidelines, we define a hierarchy of subsets $\{\mathcal{C}_l\}_{l=0}^{L}$

$$active(A^{(k)}) = \mathcal{C}_0 \supseteq \mathcal{C}_1 \supseteq ... \supseteq \mathcal{C}_L = supp(A^{(k)}). \tag{7}$$

In the ML-cycle, we treat the levels from L to 1 by applying one relaxation for the reduced problem (6) corresponding to each subset $\mathcal{C}_l$[1]. Because the subsets $\{\mathcal{C}_l\}_{l=0}^{L}$ are nested, the variables on the coarser levels will overall undergo more relaxations in the ML-cycle. Therefore, in addition to $supp(A^{(k)})$, for the coarser levels we select the variables that can reduce the objective $f(A)$ in (2) the most. These correspond to the variables (i,j) with the largest $|(\nabla f(A^{(k)}))_{ij}|$. The size of each level is chosen to be $|\mathcal{C}_{l+1}| = \lceil \frac{1}{2}|\mathcal{C}_l| \rceil$ for $0 \leq l < L-1$. The ratio $1/2$ implies that the cost of a ML-cycle is similar to that of two relaxations. A precise description of the ML-cycle is defined in Algorithm 2. Any method of the form of Algorithm 1 as well as other similar methods such as [15], can be incorporated into this multilevel framework.

Algorithm: $A^{k+1} \leftarrow$ **ML-cycle**$(A^{(k)}, Relax(\cdot))$
% Relax(): a relaxation method.
Calculate $\nabla f(A^{(k)})$, $active(A^{(k)})$ and define the hierarchy $\{\mathcal{C}_l\}_{l=0}^{L}$ in (7).
Set $A \leftarrow A^{(k)}$
For $l = L...1$
    Apply $A \leftarrow Relax(A)$ iteration for the coarse problem $F_{\mathcal{C}_l}$ defined in (6).
Apply $A \leftarrow Relax(A)$ iteration for the whole problem F in (2).
Set $A^{(k+1)} \leftarrow A$

Algorithm 2: ML-cycle for sparse inverse covariance estimation.

4 Numerical Results

In this section we compare the performance of several known methods to our multilevel framework, for solving both small-scale and large-scale instances of (2). The methods that we show in our

[1]For level 0, we reconsider all variables again, since a cost of a relaxation on $\mathcal{C}_0$ is similar to that of a relaxation that includes all variables.

Table 1: Small-scale gene-expression analysis results

Problem	n	m	λ	$\|\Sigma^{-1^*}\|_0$	QUIC	ML(QUIC)	DC-QUIC	GISTA
Lymph	587	147	0.25	17161	3.1 (10)	**2.1** (19)	6.4 (15)	5.69 (314)
ER	692	157	0.35	14940	2.9 (11)	**1.7** (14)	1.8 (12)	8.07 (338)
Arabidopsis	834	117	0.40	24442	3.9 (8)	**3.4** (16)	7.4 (16)	11.8 (351)
Leukemia	1255	71	0.50	34777	11.1 (11)	5.3 (13)	**4.5** (11)	9.4 (109)
Hereditary bc	1869	21	0.55	66285	92.9 (14)	**24.5** (18)	36.2 (17)	34.6 (178)

Table 2: Large-scale gene-expression analysis results

Code name	n	m	λ	$\|\Sigma^{-1^*}\|_0$	BCD-IC	ML(BCD)	BIG-QUIC
GSE1898	21794	182	0.70	293845	788.3 (7)	**287.9** (8)	5080 (12)
GSE20194	22283	278	0.70	197953	452.9 (8)	**170.9** (7)	2811 (10)
GSE17951	54675	154	0.78	558929	1622 (6)	**812.8** (7)	8230 (9)
GSE14322	104702	76	0.90	4973476	55314 (9)	**14164** (9)	127199 (14)

comparison include QUIC [9], DC-QUIC [7] and GISTA [6] for small-scale experiments, and BIG-QUIC [8], BCD-IC [15] for large-scale experiments. Within our multilevel framework we use QUIC and BCD-IC, which are denoted by ML(QUIC) and ML(BCD), respectively. For QUIC, BIG-QUIC, GISTA, and BCD-IC we used the default parameters suggested in the original papers and adapted the authors' software for our tests. As a stopping criterion for all methods, we follow [8] and use the condition: $\|grad^S F(A^{(k)})\|_1 < 0.01\|A^{(k)}\|_1$, where $grad^S F(\cdot)$ is the minimal norm subgradient. All solutions achieved by all algorithms correspond to functional values $F(A^*)$ which are identical up to several significant digits and have an essentially identical support size. All the experiments were run on a machine with 2 Intel Xeon E-2650 2.0GHz processors with 16 cores, 64GB RAM and Windows 7 OS.

For the small-scale experiments, we use the gene expression data sets reported in [10]. On the other hand, we use gene expression data sets from [15] for the large-scale experiments. In this case the values of λ were chosen as in [15]. These small and large-scale data sets have many variables and very few samples ($m \ll n$). The size of these data sets is large, and thus could be treated only by BCD-IC [15] and BIG-QUIC [8].

Table 1 presents the timing results (in seconds) and the number of iterations in parentheses (for ML, we count the relaxations). The ML(QUIC) algorithm shows the best performance in almost all the tests. The multilevel framework method ML(QUIC) improves the performance of the QUIC method in all the cases, although the degree of improvement is related to the amount of time spent computing the Newton direction. In the bigger tests, the performance gap is bigger as the number of non-zeros in the active set is higher and the Newton direction computation is more expensive.

Table 2 shows the results for these large data sets. Here, the advantage of the multilevel framework is more evident, as ML(BCD) clearly outperforms the other two options by significant factors. The first reason for this is the smaller Newton direction problems as in the small-scale case. Moreover, BCD-IC (and BIG-QUIC) require many iterative solutions of linear systems—these involve many matrix-vector multiplications, whose cost is directly controlled by the number of non-zeros in the matrices. By limiting the sparsity of the solution, the proposed multilevel framework saves significant computations and improves runtime substantially.

5 Conclusions

In this work, we presented a multilevel framework for the sparse inverse covariance estimation problem. Variables are treated as part of an hierarchy of coarse problems. This hierarchy enables an incremental construction of the solution in the number of non-zeros and avoids dense iterates. The framework applied to distinct methods improves performance in small and large-scale problems.

Acknowledgement: The research leading to these results has received funding from the European Union's - Seventh Framework Programme (FP7/2007-2013) under grant agreement no 623212 MC Multiscale Inversion.

References

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

[2] O. Banerjee, L. El Ghaoui, A. d'Aspremont, and G. Natsoulis. Convex optimization techniques for fitting sparse gaussian graphical models. In *Proceedings of the 23rd ICML*, pages 89–96. ACM, 2006.

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

[4] A. P. Dempster. Covariance selection. *Biometrics*, pages 157–175, 1972.

[5] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008.

[6] D. Guillot, B. Rajaratnam, B. T. Rolfs, A. Maleki, and I. Wong. Iterative thresholding algorithm for sparse inverse covariance estimation. *NIPS, Lake Tahoe CA*, 2012.

[7] Cho-Jui Hsieh, Inderjit Dhillon, Pradeep Ravikumar, and Arindam Banerjee. A divide-and-conquer method for sparse inverse covariance estimation. In *NIPS 25*, pages 2339–2347, 2012.

[8] Cho-Jui Hsieh, Matyas A Sustik, Inderjit Dhillon, Pradeep Ravikumar, and Russell Poldrack. Big & Quic: Sparse inverse covariance estimation for a million variables. In *NIPS 26*, pages 3165–3173, 2013.

[9] Cho-Jui Hsieh, Matyas A Sustik, Inderjit S Dhillon, and Pradeep D Ravikumar. Sparse inverse covariance matrix estimation using quadratic approximation. In *NIPS 24*, pages 2330–2338, 2011.

[10] Lu Li and Kim-Chuan Toh. An inexact interior point method for l-1 regularized sparse covariance selection. *Mathematical Programming Computation*, 2(3-4):291–315, 2010.

[11] R. Mazumder and T. Hastie. Exact covariance thresholding into connected components for large-scale graphical lasso. *The Journal of Machine Learning Research*, 13:781–794, 2012.

[12] Peder A Olsen, Figen öztoprak, Jorge Nocedal, and Steven J Rennie. Newton-like methods for sparse inverse covariance estimation. In *NIPS 25*, pages 764–772, 2012.

[13] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[14] E. Treister and I. Yavneh. A multilevel iterated-shrinkage approach to l_1 penalized least-squares minimization. *Signal Processing, IEEE Transactions on*, 60(12):6319–6329, 2012.

[15] Eran Treister and Javier S Turek. A block-coordinate descent approach for large-scale sparse inverse covariance estimation. In *NIPS*, 2014.

[16] Paul Tseng and Sangwoon Yun. A coordinate gradient descent method for nonsmooth separable minimization. *Mathematical Programming*, 117(1-2):387–423, 2009.

[17] M. Zibulevsky and M. Elad. L1-l2 optimization in signal and image processing. *Signal Processing Magazine, IEEE*, 27(3):76–88, May 2010.