GLASSOFAST: An efficient GLASSO implementation

Mátyás A. Sustik
Department of Computer Sciences
University of Texas at Austin
Austin, TX 78712, USA

Ben Calderhead
CoMPLEX
University College London
London, WC1E 6BT, UK

The University of Texas at Austin
UTCS Technical Report TR-12-29

November 6, 2012

***Abstract*— The GLASSO algorithm has been proposed by Friedman, Hastie and Tibshirani in 2008 to solve the ℓ_1 regularized inverse covariance matrix estimation problem. The conditional dependency structure which is captured by the inverse of the covariance matrix is of interest in numerous applications and the publication of GLASSO has spurred the development of several other algorithms aimed to solve the same optimization problem. In this paper, we present GLASSOFAST, our efficient implementation of GLASSO and demonstrate via numerical experiments that it is a magnitude faster than the original implementation.**

1. Background

The *graphical lasso* algorithm, GLASSO [5] solves the problem of estimating a sparse dependency graph via ℓ_1 regularization applied to the inverse covariance matrix, see also [1] and [4]. Even when the true inverse covariance matrix is sparse, the empirical covariance matrix

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

which is the maximum likelihood estimate, is normally dense. The addition of an ℓ_1 regularization term promotes sparsity, see [12]. We arrive to the following optimization problem:

$$\arg\min_{X\succ 0}\Big\{-\log\det X + \operatorname{tr}(SX) + \lambda\sum_{i,j=1}^{p}|x_{ij}|\Big\}, \quad (2)$$

where x_{ij} denotes the (i,j)-th element of matrix X. The ith columns and rows of matrix X are denoted by $\mathbf{x}_i$, and $\mathbf{x}_{i\cdot}$.

In [5] the authors derive a block coordinate descent method that solves (2). Their implementation is distributed as an R package called GLASSO[1].

In this paper we present a careful implementation of the original GLASSO algorithm. We call our implementation GLASSOFAST and we demonstrate in numerical experiments that it outperforms the implementation available from [5] (GLASSO R package). Our software is available from http://www.cs.utexas.edu/users/sustik/glassofast.

[1]The latest available version of GLASSO is 1.7 as of this writing.

In addition to faster execution we also fixed a problem in the GLASSO implementation that could result in an infinite loop. The performance gains of our algorithm should be interesting to the numerous applications which are using GLASSO or algorithms developed subsequently which solve the inverse covariance matrix estimation problem. These applications include structural bioinformatics [7] and certain Markov Chain Monte Carlo methodology [2] and many others.

2. Related Work

Several other algorithms emerged to find the solution of (2). The method PSM [4] applies projected subgradients, while the authors of VSM [9] propose an accelerated gradient descent method. The method ALM [11] uses an augmented Lagrangian method, while SINCO [10] uses a greedy coordinate descent and IPM [8] is an inexact interior point method. The QUIC [6] software is based on Newton's method, but also employs a quadratic approximation and coordinate descent, while exploiting sparsity.

According to comparisons in [6] the two most efficient algorithms are QUIC and GLASSO, and QUIC holds a strong lead for sparse and very sparse problems. We will compare our GLASSOFAST to GLASSO and QUIC in Section 4.

The R package implementation of GLASSO by [5] suffers from serious inefficiencies. These can be attributed to the apparent fact that the FORTRAN code is automatically generated from the true source code which is not made available[2]. It was not our primary goal to reverse engineer this implementation, but some inspection of the GLASSO code indicates that extensive and unnecessary copying is responsible for the inefficiency. We also found and mitigated a problem related to a convergence threshold which we observed to cause non-termination.

3. Algorithm implementation

We solve a generalized form of the optimization problem (2), where the regularization term is of the form $\sum_{ij}\lambda_{ij}|x_{ij}|$.

[2]The FORTRAN source for GLASSO lacks indentation and uses GOTO and CONTINUE statements to implement all the loops.

The GLASSO algorithm consists of four nested loops; the first (outermost) loop is controlled by convergence on the whole matrix, the second loop iterates through the rows of the matrix, while the two most inner loops solve a lasso problem for a given row using coordinate descent. An update in the coordinate descent step relies on the soft-thresholding operator, see [3]. For additional details on the GLASSO algorithm we refer the reader to [5].

The loop termination conditions are controlled by a user specified threshold τ. The outer loop terminates when the total change in the matrix is less than $\sigma = \tau \sum_{i \neq j} |s_{ij}|/(n-1)$, while the inner lasso loop uses σ/n as a threshold. This choice can lead to non-termination[3] when σ/n is smaller than the machine epsilon. Note, that the diagonal entries are excluded when computing σ and therefore σ can become too small even for reasonable values of τ. In the GLASSOFAST implementation we made an effort to make the code more robust in this regard by making sure that the thresholds are never too small.

We also note that GLASSO computes an approximation X that is not symmetric, while the exact solution of (2) is symmetric. In GLASSOFAST we always return a symmetric matrix. We present our GLASSOFAST implementation as Algorithm 1.

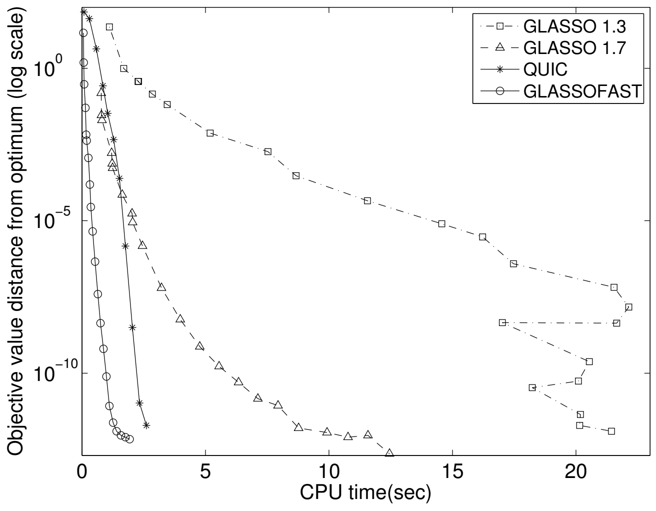
Algorithm 1: GLASSOFAST

```
Input  : $S$ $n \times n$ empirical covariance matrix, $\Lambda$
         regularization matrix, $\tau$ convergence threshold
Output: $X$ the $\ell_1$ regularized covariance matrix estimate
Set $X = 0$, $W = S$
Set $d_i = s_{ii} + \lambda_{ii}$ for $i = 1, \dots n$.
repeat
    for j = 1, n do
        $v = WX\mathbf{e}_j$, (column $j$ of $WX$)
        repeat
            for i = 1, n do
                $c = S(s_{ij} - v_i + d_i x_{ij}, \lambda_{ij})/d_i$
                $\delta = c - x_{ij}$
                if $\delta \neq 0$ then
                    $x_{ij} = c$
                    $v \leftarrow v + \delta \mathbf{w}_i$
        until Lasso($\tau$) threshold reached
        $v \leftarrow v + \mathbf{w}_j$
        $\mathbf{w}_j = v$, $\mathbf{w}_{j\cdot} = v^T$,
        $w_{jj} = d_j$
until $\tau$ threshold reached
for i = 1, n do
    $a = \mathbf{w}_i^T \mathbf{x}_i$
    $\mathbf{x}_i \leftarrow \mathbf{x}_i / a$
$X \leftarrow (X + X^T)/2$
```

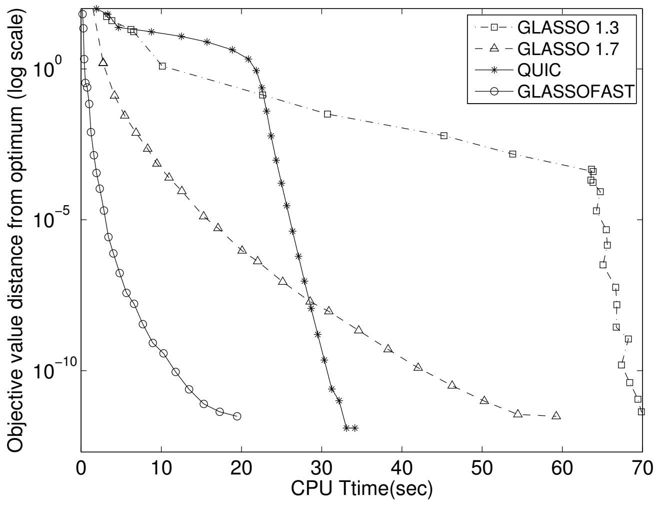
[3]Limiting the number of iterations is an option for the outer loop in GLASSO, but not for the lasso loops. The simple solution that relies on an iteration limit on the lasso loop would be inefficient.

4. Experiments

As provided, GLASSO is using single precision computations and therefore we recompiled it in order to properly compare with GLASSOFAST which defaults to double precision. We ran the experiment on a computer with an INTEL X3460 CPU running at 2.8GHz utilizing 8MB of cache. We used single threaded mode and there were no other programs running. In the experiments depicted in Figure1 we compare GLASSOFAST to versions 1.3 and 1.7 of GLASSO and to QUIC.



(a) Regularization parameter $\lambda = 0.5$ resulting in 2.2% non-zero elements.



(b) Regularization parameter $\lambda = 0.2$ resulting in 4.5% non-zero elements.

Fig. 1. Comparing GLASSO versions and GLASSOFAST on the ER dataset, $n = 692$. We observe that GLASSOFAST outperforms the GLASSO versions and QUIC.

Version 1.7 of GLASSO applies the original coordinate descent algorithm to diagonal blocks and appears to be faster than version 1.3 of GLASSO. The latter also exhibits some surprising behavior as the accuracy increases, namely, the decrease in the convergence tolerance threshold does not increase the computation time as one would expect. The explanation for this behavior is that the inner lasso computation tolerance threshold is also controlled by the user specified threshold value. The higher accuracy solution of the lasso problem though more costly, results in fewer outer loop iterations. We see that GLASSOFAST handily outperforms both versions.

While the algorithm QUIC shows faster asymptotic convergence, GLASSOFAST still runs faster in practice. We also observe that the regularization parameter can have a large impact on performance.

REFERENCES

[1] Onureena Banerjee, Laurent El Ghaoui, and Alexandre d'Aspremont. Model selection through sparse maximum likelihood estimation for multivariate Gaussian or binary data. *The Journal of Machine Learning Research*, 9, 6 2008.

[2] Ben Calderhead and Matyas A. Sustik. Sparse approximate manifolds for differential geometric MCMC. In *Advances in Neural Information Processing Systems (NIPS 2012)*, 2012. to appear.

[3] D. Donoho and I. Johnstone. Ideal spacial adaptation by wavelet shrinkage. *Biometrika*, 81:425–455, 1994.

[4] John Duchi, Stephen Gould, and Daphne Koller. Projected subgradient methods for learning sparse Gaussians. In *Conference on Uncertainty in Artificial Intelligence*, 2008.

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

[6] Cho-Jui Hsieh, Inderjit Sustik, Mátyás A. and Dhillon, and Pradeep Ravikumar. Sparse inverse covariance matrix estimation using quadratic approximation. In J. Shawe-Taylor, R.S. Zemel, P. Bartlett, F.C.N. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 2330–2338. 2011.

[7] D.T. Jones, D.W. Buchan, D. Cozzetto, and M. Pontil. PSICOV: precise structural contact prediction using sparse inverse covariance estimation on large multiple sequence alignments. *Bioinformatics*, 28:184–90, Jan 2012.

[8] Lu Li and Kim-Chuan Toh. An inexact interior point method for l1-reguarlized sparse covariance selection. *Mathematical Programming Computation*, 2:291–315, 2010.

[9] Z. Lu. Smooth optimization approach for sparse covariance selection. *SIAM Journal of Optimization*, 19:1807–1827, 2009.

[10] K. Scheinberg and I. Rish. Sinco – a greedy coordinate ascent method for sparse inverse covariance selection problem. *Technical report*, 7 2009.

[11] Katya Scheinberg, Shiqian Ma, and Donald Glodfarb. Sparse inverse covariance selection via alternating linearization methods. *NIPS*, 2010.

[12] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society B*, pages 267–288, 1996.