Notes on the Glasso Algorithm

T.K. Huang

2011/02/24

1 Preliminaries

1.1 Schur's complement

Let

$$W \;:=\; \begin{bmatrix} C & \mathbf{y} \\ \mathbf{y}^\top & a \end{bmatrix} \;\in\; \mathcal{R}^{p\times p},$$

be a symmetric matrix, where C is the upper-left sub-matrix of dimension $(p-1)$-by-$(p-1)$, $\mathbf{y}$ is a column vector of length $p-1$, and a is a real number. From Schur's complement, we have

$$W^{-1} \;=\; \begin{bmatrix} C^{-1} + \frac{C^{-1}\mathbf{y}\mathbf{y}^\top C^{-1}}{a-\mathbf{y}^\top C^{-1}\mathbf{y}} & -\frac{C^{-1}\mathbf{y}}{a-\mathbf{y}^\top C^{-1}\mathbf{y}} \\ -\frac{\mathbf{y}^\top C^{-1}}{a-\mathbf{y}^\top C^{-1}\mathbf{y}} & (a-\mathbf{y}^\top C^{-1}\mathbf{y})^{-1} \end{bmatrix}.$$

From Crammer's rule, we know

$$(W^{-1})_{pp} \;=\; (-1)^{2p}\frac{|C|}{|W|} \;\;\Longleftrightarrow\;\; |W| \;=\; \frac{|C|}{(W^{-1})_{pp}} \;=\; |C|(a-\mathbf{y}^\top C^{-1}\mathbf{y}),$$

where $|\cdot|$ denotes the determinant of a matrix. Therefore,

$$\log|W| \;=\; \log|C| + \log(a-\mathbf{y}^\top C^{-1}\mathbf{y}). \tag{1}$$

1.2 Dual norm of the l1 norm

Consider the following two types of norms[1]:

- l1 norm: $\|X\|_1 := \sum_{i,j}|X_{ij}|$.
- sup norm: $\|X\|_\infty := \max_{i,j}|X_{i,j}|$.

[1]These are *entrywise* norms, meaning that they treat matrices as vectors. There are also *induced* or *operator* norms that use the same notation but are defined in a different way.

These two norms are dual to each other, meaning that

$$\begin{aligned} \|\Omega\|_1 &= \max_{U:\|U\|_\infty \le 1} \operatorname{tr}(\Omega U), \\ \|U\|_\infty &= \max_{\Omega:\|\Omega\|_1 \le 1} \operatorname{tr}(\Omega U). \end{aligned}$$

2 The Glasso algorithm

Let $S \in \mathcal{R}^{p\times p}$ be a sample covariance matrix. We aim to solve the following regularized maximum likelihood problem:

$$\min_{\Omega \succ 0} \quad \mathcal{L}(\Omega) := \operatorname{tr}(\Omega S) - \log|\Omega| + \lambda\|\Omega\|_1. \tag{2}$$

This is a convex optimization problem: the l1 penalty is convex, the linear term is convex, the negative log determinant function is convex, and the set of positive semidefinite matrices is also convex. In particular, (2) is in the form of *Semidefinite Programming* (SDP) because the variable is constrained to be a PSD matrix. SDPs are known to be hard convex optimization problems. Although there exist algorithms for solving general SDPs, such as the interior point method, most of these algorithms/solvers usually become quite inefficient when the dimension of the variable matrix exceeds hundreds. Therefore, machine learning researchers have been focusing specifically on solving (2), and as a result developed the Glasso algorithm. The main idea of the Glasso algorithm consists of the following three ingredients:

- Instead of the regularized maximum likelihood problem (2), solve its dual problem.
- Decompose the dual problem of (2) into a series of sub-problems, and iteratively solve the sub-problems until convergence.
- For each sub-problem, solve its dual via the Lasso algorithm.

The first two ingredients are due to [1], while the third is due to [3]. The name "Glasso" was coined by [3]. In the following we describe the three ingredients in more detail.

2.1 Dual of the regularized maximum likelihood (2)

Re-writing the objective function using the dual of the sup norm, we get

$$\begin{aligned} \mathcal{L}(\Omega) &= \operatorname{tr}(\Omega S) - \log|\Omega| + \lambda \max_{U:\|U\|_\infty \le 1} \operatorname{tr}(\Omega U) \\ &= \operatorname{tr}(\Omega S) - \log|\Omega| + \max_{U:\|U\|_\infty \le \lambda} \operatorname{tr}(\Omega U) \\ &= -\log|\Omega| + \max_{U:\|U\|_\infty \le \lambda} \operatorname{tr}(\Omega(S+U)). \end{aligned}$$

Consider the following function:

$$h(\Omega, U) \ := \ -\log|\Omega| + \mathrm{tr}(\Omega(S+U))$$

with the domain $\{\Omega, U \mid \Omega \succ 0, \|U\|_\infty \leq \lambda\}$. Using strong duality, we have

$$\min_{\Omega \succ 0} \mathcal{L}(\Omega) = \min_{\Omega \succ 0} \max_{\|U\|_\infty \leq \lambda} h(\Omega, U) \ = \ \max_{\|U\|_\infty \leq \lambda} \min_{\Omega \succ 0} h(\Omega, U).$$

Under a fixed U, it is easy to see $\hat{\Omega} = (S+U)^{-1}$ minimizes $h(\Omega, U)$. Plugging $\hat{\Omega}$ back in h and doing a change of variable by defining $W := S + U$, we get the dual optimization problem:

$$\max_{W : \|W - S\|_\infty \leq \lambda, W^\top = W} \quad \mathcal{D}(W) \ := \ \log|W| \tag{3}$$

Let $\hat{\Sigma}$ be the solution to the dual (3). We obtain the estimated precision matrix by $\hat{\Omega} = \hat{\Sigma}^{-1}$. From (1), we can see that $\hat{\Sigma}_{ii} = S_{ii} + \lambda$, i.e., the i-th diagonal element of $\hat{\Sigma}$ is the i-th diagonal element of S plus λ. Therefore, in solving (3) we only need to consider off-diagonal elements.

2.2 Solving the dual problem by block coordinate descent

The strategy to solve (3) is to optimize one row/column of W (excluding the diagonal element) at a time, and then iterate over all rows/columns until convergence. By permuting rows and columns, we can always assume the last row/column of W is currently being optimized. Using Schur's complement, we re-write the objective function:

$$\log|W| = \log|C| + \log(S_{pp} + \lambda - \mathbf{y}^\top C^{-1} \mathbf{y}). \tag{4}$$

Since $\log(\cdot)$ is a monotonically increasing function and $S_{pp} + \lambda$ is a constant, maximizing (4) over $\mathbf{y}$ is equivalent to the following QP:

$$\begin{aligned} \min_{\mathbf{y} \in \mathcal{R}^{p-1}} \quad & \mathbf{y}^\top C^{-1} \mathbf{y}, \\ \text{s.t.} \quad & \|\mathbf{y} - S_p\|_\infty \leq \lambda, \end{aligned} \tag{5}$$

where S_p is the p-th column of S excluding S_{pp}.

2.3 Solving each block coordinate descent by Lasso

Solving (5) is not a good idea because we need to do matrix inversion, whose time complexity grows cubically with p. ([1] points out that when solving the sub-problems iteratively, one can compute the matrix inversion in quadratic time using rank-one updates). The Glasso algorithm avoids this difficulty by looking at the dual of the sub-problem:

$$\min_{\mathbf{x}} \quad \frac{1}{2} \mathbf{x}^\top C \mathbf{x} - S_p^\top \mathbf{x} + \lambda \|\mathbf{x}\|_1, \tag{6}$$

where the relation between the primal and dual variables is

$$\mathbf{y} = C\mathbf{x}. \tag{7}$$

To verify that (6) is the dual of (5), we first re-write the constraint in (5) as a set of bound constraints:

$$\|\mathbf{y} - S_p\|_\infty \leq \lambda \iff -\lambda + (S_p)_i \leq \mathbf{y}_i \leq \lambda + (S_p)_i, \quad i = 1, \ldots, p,$$

and then the Lagrangian:

$$l(\mathbf{y}, \boldsymbol{\alpha}, \boldsymbol{\beta}) \;=\; \mathbf{y}^\top C^{-1}\mathbf{y} - \boldsymbol{\alpha}^\top(\lambda\mathbf{e} + S_p - \mathbf{y}) - \boldsymbol{\beta}^\top(\mathbf{y} + \lambda\mathbf{e} - S_p),$$

where $\boldsymbol{\alpha}_i \geq 0, \boldsymbol{\beta}_i \geq 0,\ i = 1, \ldots, p$, are the Lagrange multipliers for the inequality constraints and $\mathbf{e}$ is a column vector of ones. Maximizing $l(\mathbf{y}, \boldsymbol{\alpha}, \boldsymbol{\beta})$ w.r.t $\mathbf{y}$, we get

$$\mathbf{y} = C\mathbf{x}, \quad \text{where } \mathbf{x} := \left(\frac{\boldsymbol{\beta} - \boldsymbol{\alpha}}{2}\right). \tag{8}$$

Using the complementary slackness condition, we have that the optimal Lagrange multipliers must satisfy

$$\boldsymbol{\alpha}_i\boldsymbol{\beta}_i \;=\; 0,\ i = 1, \ldots, p,$$

implying that $\|\mathbf{x}\|_1 = \frac{(\boldsymbol{\alpha}+\boldsymbol{\beta})^\top\mathbf{e}}{2}$. Using this fact and plugging (8) back in the Lagrangian, we get the dual problem (6).

To show that (6) is actually a Lasso problem, we let $Q := C^{1/2}$, $\mathbf{b} := \frac{1}{2}Q^{-1}S_p$, and re-write (6) as

$$\min_{\mathbf{x}} \quad \frac{1}{2}\|Q\mathbf{x} - \mathbf{b}\|_2^2 + \lambda\|\mathbf{x}\|_1.$$

Note that to use the Lasso algorithm we do not need to compute Q and $\mathbf{b}$. In fact, the usual Lasso algorithm first computes the Hessian and the first-order term from Q and $\mathbf{b}$, but here we already have them.

Putting everything together, we give a summary of the Glasso algorithm in the following:

1. Initialize $W := S + \lambda I$.
2. Repeat until convergence:
   For $i = 1$ to p,
   - Run Lasso to solve the sub-problem (6) with C being the sub-matrix of the current W excluding the i-th row and column.
   - Update the i-th row and column of W by (7)
3. Return the solution $\hat{W}$ and $\hat{\Omega} = \hat{W}^{-1}$. The latter is computed by Schur's complement:
   For $i = 1$ to p,

- Let $\hat{\mathbf{x}}$ be the solution to (6) for the i-th row/column and $\hat{\mathbf{y}} = C\hat{\mathbf{x}}$.
- Compute the i-th row/column of $\hat{\Omega}$ excluding the diagonal element by

$$\hat{\Omega}_i \;=\; -\frac{\hat{\mathbf{x}}}{\lambda + S_{ii} - \hat{y}^\top \hat{\mathbf{x}}},$$

  and the i-th diagonal element by

$$\hat{\Omega}_{ii} \;=\; -\frac{1}{\lambda + S_{ii} - \hat{y}^\top \hat{\mathbf{x}}}.$$

2.4 Convergence

Theorem 3 of [1] proves the convergence of this particular block coordinate descent procedure using the convergence result for general block coordinate descent algorithms given by [2]. However, that result requires blocks of variables to be disjoint from one another, which is not the case here. Nevertheless, [4] gives convergence proofs for a row-by-row type of block coordinate descent algorithms for solving a class of SDPs, which includes the Glasso algorithm as a special case. In practice, the Glasso algorithm is observed to converge usually within a few sweeps over all columns/rows. See [3] for more numerical experiments and running time results.

References

[1] O. Banerjee, L. E. 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, 2008.

[2] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, MA 02178-9998, second edition, 1999.

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

[4] Z. Wen, D. Goldfarb, S. Ma, and K. Scheinberg. Row by row methods for semidefinite programming. Technical report, Dept of IEOR, Columbia University, 2009.