Maximum likelihood estimation over multiple undirected graphs

Yunzhang Zhu, Xiaotong Shen and Wei Pan

University of Minnesota

December 13, 2011

1. Introduction
   - Estimation of multiple precision matrices
   - Motivating examples
2. Computation
   - DC programming
   - Blockwise minimization
3. Theory
4. Numerical examples

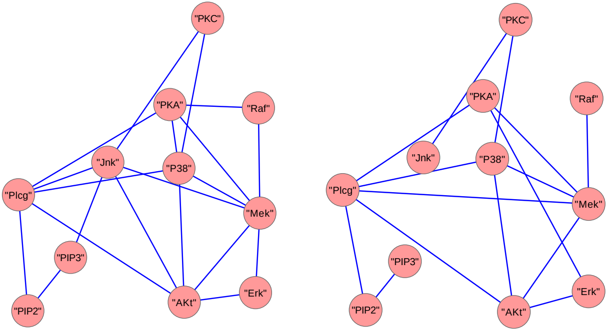
Estimation of multiple precision matrices

- Multiple Gaussian graphical models:
  - Data: $\boldsymbol{Y}_1^{(l)}, \cdots, \boldsymbol{Y}_{n_l}^{(l)} \sim N(\boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l); l = 1, \cdots, L$. and Σ_l is $p \times p$ covariance matrix.
  - Parameter of interest: $\boldsymbol{X}_l = \boldsymbol{\Sigma}_l^{-1}; l = 1, \cdots, L$.
- High-dimensional cases:
  - Network size: p could be large.
  - # networks: L could be large.
  - # of parameters: $\frac{p(p-1)}{2} L >> n = \sum_{l=1}^{L} n_l$).

Why multiple Gaussian graphical models?

- Existing research has focused on single precision matrix estimation through sparseness–Glasso,...
- Networks sharing sparsity pattern: (Guo et al 2011)
  - Entries of matrices tend to be simultaneously zero or not.
- Time-varying networks: (Kolar et al 2009), (Zhou et al 2010), ...
  - Network structures change over time: Adjacent matrices tend to be similar.
- Our goal: Exploit sparse structure within matrices and cluster structure among matrices.

Motivating examples



(a) condition 1 (b) condition 2

Estimation of multiple precision matrices

- Regularization:

$$S(\boldsymbol{X}) = \sum_{l=1}^{L} \Big(-\log det(\boldsymbol{X}_l) + tr(\boldsymbol{S}_l \boldsymbol{X}_l) \Big) + J\big(\{x_{jkl} : k \neq j\}\big),$$

Precision matrices: $\boldsymbol{X} = \big(\boldsymbol{X}_1, \cdots, \boldsymbol{X}_L\big)$; Sample covariances: $\boldsymbol{S}^{(1)}, \cdots, \boldsymbol{S}_L$; $J\big(\{x_{jkl} : k \neq j\}\big)$ over on off-diagonal elements.

- Idea: Sparseness pursuit within matrices and clustering over matrices.
- Penalty: $J\big(\{x_{jkl} : k \neq j\}\big) = \lambda_1 \sum_{(l,j,k):k\neq j} J_\tau(|x_{jkl}|) + \lambda_2 \sum_{(j,k,l_1,l_2):|l_1-l_2|=1,k\neq j} J_\tau(|x_{jkl_1} - x_{jkl_2}|)$
  - $J_\tau(x) = \min(\frac{x}{\tau}, 1) \to \mathbb{I}(x \neq 0)$ as $\tau \to 0$ (computational surrogate for ℓ_0 penalty).

Multiple precision matrices estimation

- Parameter estimator:
  - $p(p+1)L/2$ unknown parameters.
  - Existing methods such Glasso for single matrix estimation break down.
- Our strategy: Difference convex programming $+$ block-wise minimization by sweeping each column of $\boldsymbol{X}_l$ across $l = 1, \cdots, L$.
  - DC programming: treat non-convex minimization by solving a sequence of convex relaxations.
  - Blockwise minimization: each convex relaxation problem is solved by blockwise coordinate descent method.

Difference convex programming

- DC decomposition: a difference of two convex functions:

$$J_\tau(|x|) = |x| - \max(|x| - \tau, 0) \tag{1}$$

- Convex relaxation: given current solution, linearize $\max(|x| - \tau, 0)$ in (1).
- Convex subproblem:

$$\sum_{l=1}^{L} \Big(-\log det(\boldsymbol{X}_l) + tr(\boldsymbol{S}_l \boldsymbol{X}_l) \Big) + J^{(m)}_{\text{con}}\big(\{x_{jkl}; k \neq j\}\big), \tag{2}$$

$$J^{(m)}_{\text{con}}\big(\{x_{jkl}; k \neq j\}\big) = \sum_{j=1}^{p} \sum_{k \neq j} \Big(\lambda_1 \sum_{l=1}^{L} \mathbb{I}\big(|x^{(m)}_{jkl}| \leq \tau\big)|x_{jkl}| + \lambda_2 \sum_{l=2}^{L} \mathbb{I}\big(|x^{(m)}_{jk(l-1)} - x^{(m)}_{jkl}| \leq \tau\big)|x_{jk(l-1)} - x_{jkl}| \Big).$$

- Blockwise coordinate descent: For (2), blockwise coordinate descent is used to sweep over one column at a time.

Blockwise minimization

- Notation: decompose the matrix in terms of jth column

$$\boldsymbol{X}_l = \begin{pmatrix} \boldsymbol{X}_{-jl} & \boldsymbol{x}_{-jl} \\ \boldsymbol{x}_{-jl}^T & x_{jjl} \end{pmatrix}, \quad \boldsymbol{S}_l = \begin{pmatrix} \boldsymbol{S}_{-jl} & \boldsymbol{s}_{-jl} \\ \boldsymbol{s}_{-jl}^T & s_{jjl} \end{pmatrix}$$

$$\boldsymbol{x}_{-j} = (\boldsymbol{x}_{-j1}, \cdots, \boldsymbol{x}_{-jL}), \quad J_{\text{con}}^{(m)}(\{x_{jkl} : k \neq j\}) = \sum_{j=1}^{p} J_j^{(m)}(\boldsymbol{x}_{-j})$$

with

$$J_j^{(m)}(\boldsymbol{x}_{-j}) = \sum_{k \neq j} \Big(\lambda_1 \sum_{l=1}^{L} \mathbb{I}(|x_{jkl}^{(m)}| \leq \tau)|x_{jkl}| + \lambda_2 \sum_{l=2}^{L} \mathbb{I}(|x_{jk(l-1)}^{(m)} - x_{jkl}^{(m)}| \leq \tau)|x_{jk(l-1)} - x_{jkl}| \Big)$$

Blockwise minimization

- One-column problem: fixing $\boldsymbol{X}_{-j1}, \cdots, \boldsymbol{X}_{-jL}$ and using

$$det(\boldsymbol{X}_l) = det\Big(\boldsymbol{X}_{-jl}(\boldsymbol{x}_{jjl} - (\boldsymbol{x}_{-jl})^T(\boldsymbol{X}_{-jl})^{-1}\boldsymbol{x}_{-jl})\Big),$$

- Negative log-likelihood:

$$\sum_{l=1}^{L}\left(-\log\Big(x_{jjl}-(\boldsymbol{x}_{-jl})^T(\boldsymbol{X}_{-jl})^{-1}\boldsymbol{x}_{-jl})\Big)+s_{jjl}x_{jjl}+2(\boldsymbol{s}_{-jl})^T\boldsymbol{x}_{-jl}\right).$$

- Optimizing out x_{jjl}: Plugging in the optimal $x_{jjl} = 1/s_{jjl} + (\boldsymbol{x}_{-jl})^T(\boldsymbol{X}_{-jl})^{-1}\boldsymbol{x}_{-jl}$, negative log-likelihood becomes

$$\sum_{l=1}^{L}\Big(s_{jjl}(\boldsymbol{x}_{-jl})^T(\boldsymbol{X}_{-jl})^{-1}(\boldsymbol{x}_{-jl}) + 2(\boldsymbol{s}_{-jl})^T\boldsymbol{x}_{-jl}\Big),$$

which is quadratic in $\boldsymbol{x}_{-j}$ (if fixing $\boldsymbol{X}_{-jl}$)!

Algorithm

Algorithm 1:

- Initialization: Set $X_l^{(0)} = \mathbb{I}_{p\times p}$; $l = 1, \cdots, L$.
- **Step 1:** Iterate over m, compute $(\boldsymbol{X}_1^{(m+1)}, \cdots, \boldsymbol{X}_L^{(m+1)})$ by solving (2) based on $(\boldsymbol{X}_1^{(m)}, \cdots, \boldsymbol{X}_L^{(m)})$ using **Step 2.** Terminate if $(\boldsymbol{X}_1^{(m+1)}, \cdots, \boldsymbol{X}_L^{(m+1)}) = (\boldsymbol{X}_1^{(m)}, \cdots, \boldsymbol{X}_L^{(m)})$.
- **Step 2:** Iterate over $j = 1, \cdots, p$, update $\boldsymbol{x}_{-j}$ by solving

$$\min_{\boldsymbol{x}_{-j}} \Big(\sum_{l=1}^{L} \Big(s_{jjl} (\boldsymbol{x}_{-jl})^T (\boldsymbol{X}_{-jl})^{-1} (\boldsymbol{x}_{-jl}) + 2 (\boldsymbol{s}_{-jl})^T \boldsymbol{x}_{-jl} \Big) + J_j^{(m)} (\boldsymbol{x}_{-j}) \Big),$$

and update x_{jjl} according to

$$x_{jjl} \leftarrow 1/s_{jjl} + (\boldsymbol{x}_{-jl})^T (\boldsymbol{X}_{-jl})^{-1} \boldsymbol{x}_{-jl}.$$

Terminate when certain pre-specified stopping criterion is met.

Matrix inverse problem

Remark about Algorithm 1:

- Matrix inverse: every iteration requires inverting L $(p-1)\times(p-1)$ sub-matrices $\boldsymbol{X}_{-jl}$.
- Efficient update for inverses:
  - efficient updates requiring only $O(p^2)$ operation for each inverse (matrix-vector multiplication and matrix-matrix addition)
  - this is possible because most elements of $\boldsymbol{X}_l$ are unchanged within each iteration
- Efficient algorithm for the sub-problem: fixing $\boldsymbol{X}_{-jl}$, the sub-problem is easy to solve.
- Generalization: applicable to general form of penalty as long as it is "separable" in column.

Exponential probability error bound for oracle recovery

- Definition of oracle estimator:
  - MLE given the true sparsity pattern and cluster.
- Simultaneous recovery probability:

$$\mathbb{P}(\text{oracle recovery}) \geq 1 - \exp\left\{-n\left(c_1 C_{min}(\boldsymbol{\theta}^0) - 2\frac{\log d}{n} - \frac{\log(S^\star)}{n}\right)\right\}$$

under some regularity conditions.

  - $C_{min}(\boldsymbol{\theta}^0)$: quantity reflecting the level of difficulty of recovery.
  - d: dimension of parameter of interest.
  - $S^\star$: quantity related to the graph in the clustering penalty and the true graph.

The multiple precision matrix case

- Lower bound on entropy:

$$C_{min}(\boldsymbol{\theta}^0) \geq c_2 \min_{1 \leq l \leq L} c_{min}(H_l)\eta_{min}^2,$$

where

$$\eta_{min} = \min\left(\min_{(i,i',l):x^0_{ii'l} \neq 0} \left|x^0_{ii'l}\right|, \frac{1}{\sqrt{2}} \min_{(i,i',l):x^0_{ii'l} - x^0_{ii'(l+1)} \neq 0} \left|x^0_{ii'l} - x^0_{ii'(l+1)}\right| \right)$$

- Upper bound on complexity:

$$\log S^\star \leq 2g_0 \max(\log(d_0/g_0), 1),$$

where $g_0 \equiv \sum_{l=1}^{L-1} \sum_{i>i'} \mathbb{I}\big(x^0_{ii'l} \neq x^0_{ii'(l+1)}\big)$ is the number of break points among these clusters.

The multiple precision matrix case

- Sufficient condition for simultaneous pursuit of sparseness and clustering:

$$\text{If } \min_{1 \le l \le L} c_{min}(H_l)\eta_{min}^2 \ge c_3 \frac{\log(p^2 L) - g_0 \max(\log(d_0/g_0), 1)}{n}$$

for some constant $c_3 > 0$, then

$$\mathbb{P}\left(\hat{\boldsymbol{X}}^{\ell_0} \neq \hat{\boldsymbol{X}}^{o}\right) \text{ and } \mathbb{P}\left(\hat{\boldsymbol{X}}^{g} \neq \hat{\boldsymbol{X}}^{o}\right) \to 0 \text{ as } n, d \to +\infty.$$

Evaluation metrics

- The following metrics are considered:
  - Average entropy loss (EL):

$$EL = \frac{1}{L}\sum_{l=1}^{L}\left(tr\left((\boldsymbol{X})_l^{-1}\hat{\boldsymbol{X}}_l\right) - \log\det\left((\boldsymbol{X}_l)^{-1}\hat{\boldsymbol{X}}_l\right)\right)$$

  - Average quadratic loss (QL):

$$QL = \frac{1}{L}\sum_{l=1}^{L} tr\left[\left((\boldsymbol{X}_l)^{-1}\hat{\boldsymbol{X}}_l - \boldsymbol{I}\right)^2\right]$$

Tuning

- Cross-validation:

$$CV(\boldsymbol{\lambda}) = \frac{1}{L}\sum_{l=1}^{L}\Big(-\log\det\big(\hat{\boldsymbol{X}}_l(\boldsymbol{\lambda})\big) + tr\big(\boldsymbol{S}_l^{\text{test}}\hat{\boldsymbol{X}}_l(\boldsymbol{\lambda})\big)\Big),$$

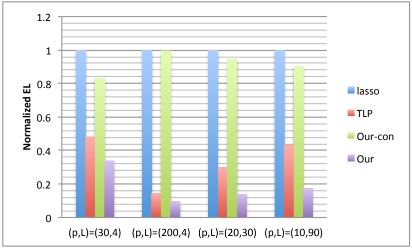
where S_l^{test} is the sample covariance matrix for the testing set; $l = 1, \cdots, L$.

- Grid search: get the tuning parameter by minimizing $CV(\boldsymbol{\lambda})$ through a grid search.

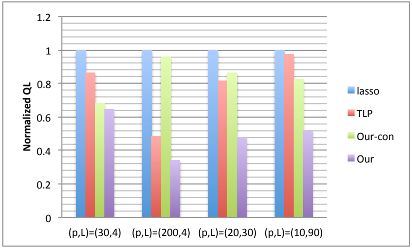
Example1

- Chain Networks: $\boldsymbol{X}_l^{-1} = \boldsymbol{\Sigma}_l$ is AR(1)-structured
  - ij-element: $\sigma_{ijl} = \exp(-|s_{il} - s_{jl}|/2)$ with $s_{1l} < s_{2l} < \cdots < s_{pl}$, $s_{il} - s_{(i-1)l} \sim$ Unif(0.5, 1).
- Scenarios:
  - $(n, p) = (120, 30)$, $(n, p, L) = (120, 200)$ and $\boldsymbol{X}_1 = \boldsymbol{X}_1, \boldsymbol{X}_3 = \boldsymbol{X}_4$;
  - $(n, p, L) = (120, 20, 30)$, $(n, p, L) = (120, 10, 90)$ and $\boldsymbol{X}_1 = \cdots = \boldsymbol{X}_{L/3}$, $\boldsymbol{X}^{(1+L/3)} = \cdots = \boldsymbol{X}_{2L/3}$ and $\boldsymbol{X}_{1+2L/3} = \cdots = \boldsymbol{X}_L$.

Example1



(c) example 1: entropy loss



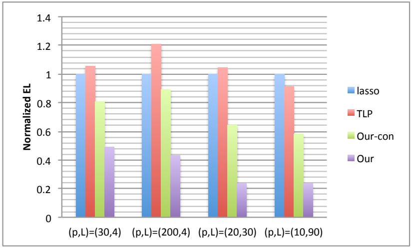
(d) example 1: quadratic loss

- The proposed method outperforms its convex counterpart.
- Simultaneous sparsity and clustering pursuit outperforms that with only sparsity pursuit, and clustering alone.

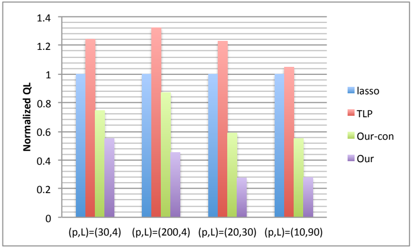
Example2

- Nearest Neighbor Networks: detailed generating scheme in (Li and Gui 2006)
- Scenarios:
  - $(n, p, L) = (300, 30, 4)$ and $(n, p, L) = (300, 200, 4)$, where $\boldsymbol{X}_1 = \boldsymbol{X}_2$, $\boldsymbol{X}_3 = \boldsymbol{X}_4$ with the second cluster of matrices $(\boldsymbol{X}_3, \boldsymbol{X}_4)$ deleting one edge for each node.
  - $(n, p, L) = (300, 20, 30)$ and $(n, p, L) = (300, 10, 90)$, where $\boldsymbol{X}_1 = \cdots = \boldsymbol{X}_{L/3}, \boldsymbol{X}_{1+L/3} = \cdots = \boldsymbol{X}_{2L/3}, \boldsymbol{X}_{1+2L/3} = \cdots = \boldsymbol{X}_l$ with the second cluster of matrices $(\boldsymbol{X}_3, \boldsymbol{X}_4)$ deleting one edge for each node and the third cluster of matrices $(\boldsymbol{X}_{1+2L/3}, \cdots, \boldsymbol{X}_L)$ adding an edge.

Example2



(e) Example 2: entropy loss

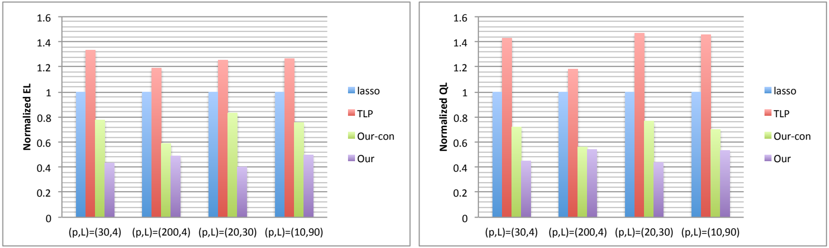


(f) Example 2: quadratic loss

Example3

- Exponential Decay Matrix:
  - ij-element: $x_{ijl} = \exp\left(a_l|i-j|\right)$ with a_l sampled uniformly over $[1, 2]$.
- Scenarios:
  - $(p, L) = (30, 4)$, $(p, L) = (200, 4)$ and $n = 120$ or 300 with $\boldsymbol{X}_1 = \boldsymbol{X}_2$, $\boldsymbol{X}_3 = \boldsymbol{X}_4$
  - $(p, L) = (20, 30)$, $(p, L) = (10, 90)$ and the sample size $n = 120$ or 300 with $\boldsymbol{X}_1 = \cdots = \boldsymbol{X}_{L/3}, \boldsymbol{X}_{1+L/3} = \cdots = \boldsymbol{X}_{2L/3}, \boldsymbol{X}_{1+2L/3} = \cdots = \boldsymbol{X}_L$.

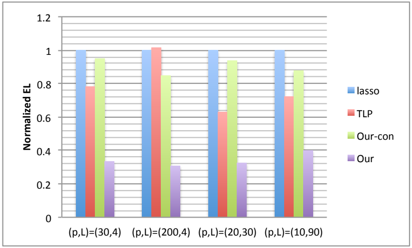
Example3: $n = 120$



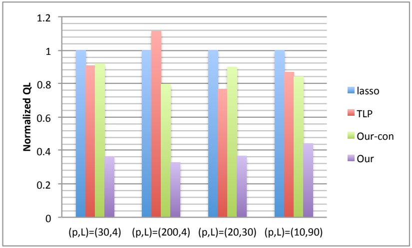
(g) example 3: entropy loss

(h) example 3: quadratic loss

Example3: $n = 300$



(i) example 3: entropy loss



(j) example 3: quadratic loss

Proteins data

- Multiple protein network: protein networks under different experimental conditions