Sparse Estimation of a Covariance Matrix (2011)

Jacob Bien
Robert Tibshirani

May 13, 2014
Presented by Sen Zhao

Last Time

- Goal: Simultaneously estimate
  - the location of zero entries in a covariance matrix;
  - the value of nonzero entries in a covariance matrix.

Last Time

- Goal: Simultaneously estimate
  - the location of zero entries in a covariance matrix;
  - the value of nonzero entries in a covariance matrix.
- Method: Maximize penalized likelihood with Lasso (ℓ_1) penalty.

$$\hat{\mathbf{\Sigma}} = \textit{arg}\min_{\mathbf{\Sigma}}(\log\det(\mathbf{\Sigma}) + \textit{tr}(\mathbf{\Sigma}^{-1}\boldsymbol{S}) + \lambda\|\boldsymbol{P} * \mathbf{\Sigma}\|_1)$$

  - $\boldsymbol{S}$: sample covariance matrix;
  - λ: tuning parameter;
  - $\boldsymbol{P}$ penalty matrix of our choice.

Last Time

- Goal: Simultaneously estimate
  - the location of zero entries in a covariance matrix;
  - the value of nonzero entries in a covariance matrix.
- Method: Maximize penalized likelihood with Lasso (ℓ_1) penalty.

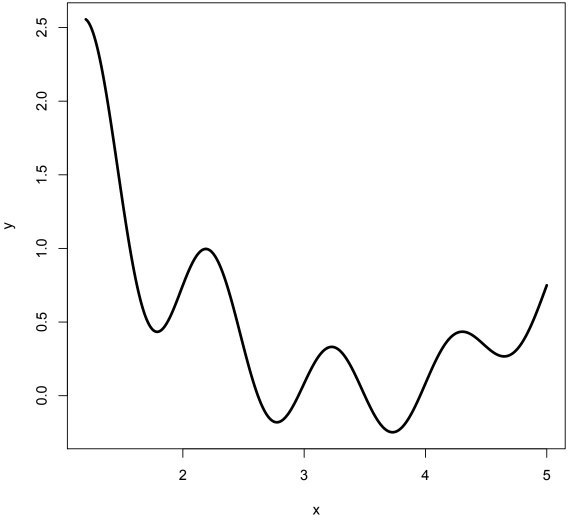
$$\hat{\boldsymbol{\Sigma}} = \textit{arg}\min_{\boldsymbol{\Sigma}}(\log\det(\boldsymbol{\Sigma}) + \textit{tr}(\boldsymbol{\Sigma}^{-1}\boldsymbol{S}) + \lambda\|\boldsymbol{P} * \boldsymbol{\Sigma}\|_1)$$

  - $\boldsymbol{S}$: sample covariance matrix;
  - λ: tuning parameter;
  - $\boldsymbol{P}$ penalty matrix of our choice.
- Difficulty: The optimization problem is not convex

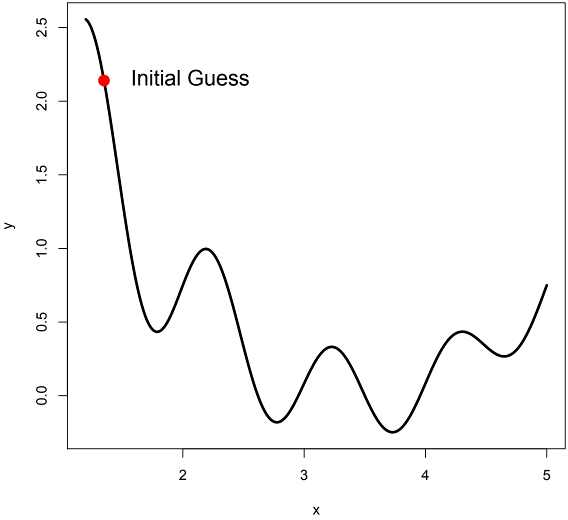
This Time

- Method to solve this non-convex optimization problem
- Some simulation results

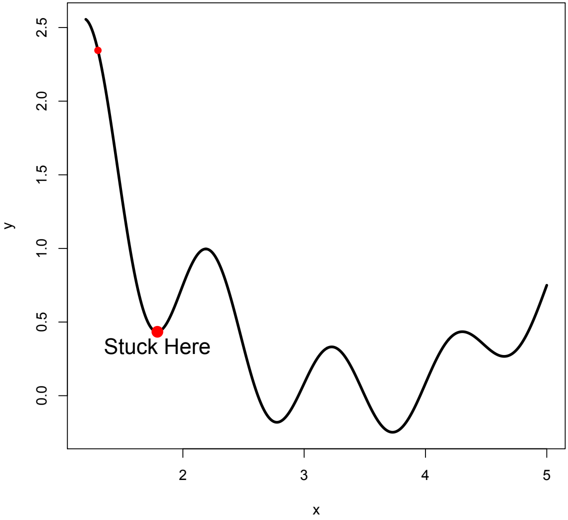
Majorize-Minimization (MM) Algorithm (Lange, 2004, Hunter and Li, 2005)



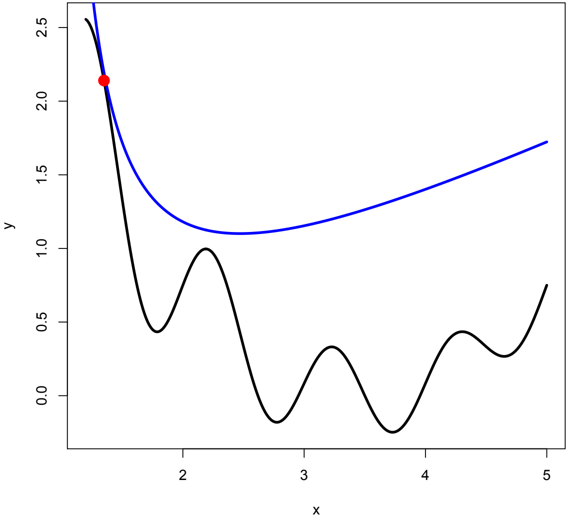
Majorize-Minimization (MM) Algorithm (Lange, 2004, Hunter and Li, 2005)



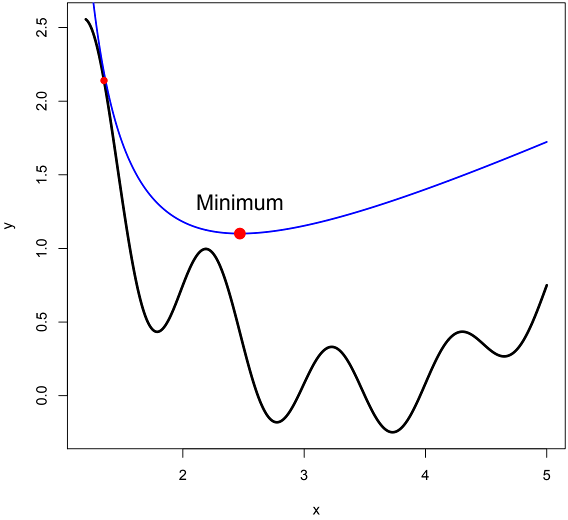
Majorize-Minimization (MM) Algorithm (Lange, 2004, Hunter and Li, 2005)



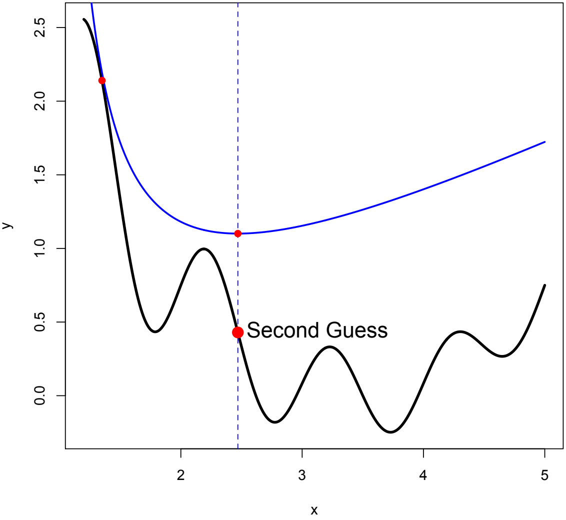
Majorize-Minimization (MM) Algorithm (Lange, 2004, Hunter and Li, 2005)



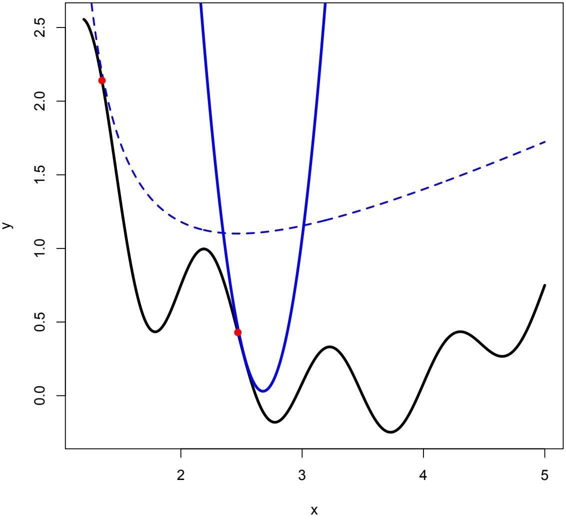
Majorize-Minimization (MM) Algorithm (Lange, 2004, Hunter and Li, 2005)



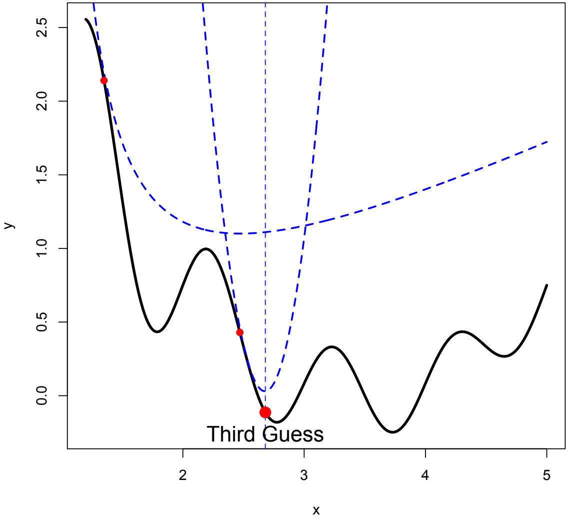
Majorize-Minimization (MM) Algorithm (Lange, 2004, Hunter and Li, 2005)



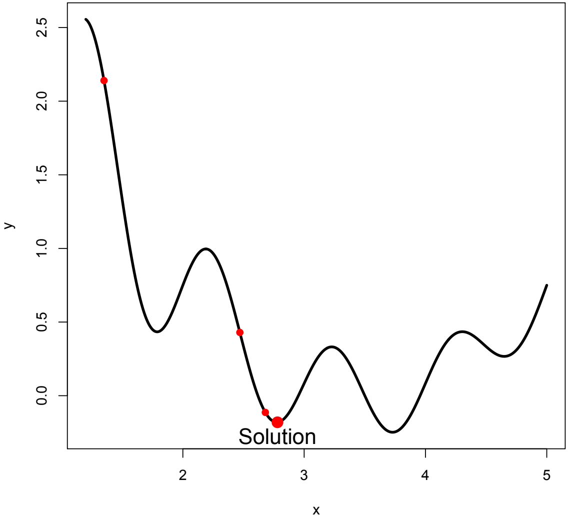
Majorize-Minimization (MM) Algorithm (Lange, 2004, Hunter and Li, 2005)



Majorize-Minimization (MM) Algorithm (Lange, 2004, Hunter and Li, 2005)



Majorize-Minimization (MM) Algorithm (Lange, 2004, Hunter and Li, 2005)



Majorize-Minimization (MM) Algorithm (Lange, 2004, Hunter and Li, 2005)

- Suppose $g(x)$ is the (non-convex) function we want to minimize.

1. Pick an initial guess point x_0.
2. Find a convex surrogate function $f(x)$ such that:
   - $f(x) \geq g(x)$ for any x;
   - $f(x_0) = g(x_0)$.
3. Update of $x_0 = arg \min_x f(x)$.
4. Repeat 2-3 until convergence.

Things to Consider

- How to find the convex surrogate function.
  - Difference of Convex Functions (DC) Programming (An and Tao, 2005)
- How to minimize the convex surrogate function.
  - Generalized Gradient Descent (Beck and Teboulle, 2009)
  - Alternating Direction Method of Multipliers (ADMM) (Boyd, 2011)

Difference of Convex Functions Programming (An and Tao, 2005)

- Suppose we want to minimize $g(x) = a(x) - b(x)$.
  - $a(x)$ and $b(x)$ are convex functions.
- Suppose $b'_{x_0}(x)$ is the tangent line of $b(x)$ at x_0.
  - $b'_{x_0}(x_0) = b(x_0)$;
  - $b'_{x_0}(x) \leq b(x)$ for all x.

Difference of Convex Functions Programming (An and Tao, 2005)

- Suppose we want to minimize $g(x) = a(x) - b(x)$.
  - $a(x)$ and $b(x)$ are convex functions.
- Suppose $b'_{x_0}(x)$ is the tangent line of $b(x)$ at x_0.
  - $b'_{x_0}(x_0) = b(x_0)$;
  - $b'_{x_0}(x) \leq b(x)$ for all x.
- $f(x) = a(x) - b'_{x_0}(x)$ is the convex surrogate function.
  - $f(x)$ is convex;
  - $f(x_0) = a(x_0) - b'_{x_0}(x_0) = a(x_0) - b(x_0) = g(x_0)$;
  - $f(x) = a(x) - b'_{x_0}(x) \geq a(x) - b(x) = g(x)$ for all x.
- The convex surrogate function in this case:

$$f(\boldsymbol{\Sigma}) = \log(\det(\boldsymbol{\Sigma}_0)) + tr(\boldsymbol{\Sigma}_0^{-1}\boldsymbol{\Sigma}) - p + tr(\boldsymbol{\Sigma}^{-1}\boldsymbol{S}) + \lambda\|\boldsymbol{P} * \boldsymbol{\Sigma}\|_1$$

What's Wrong with the Newton-Raphson Method?

- The convex surrogate function is not differentiable.
- There is an implicit constraint that $\mathbf{\Sigma}$ is positive semi-definite.

Simulation Setup

- Three methods to consider:
  - Soft-thresholded sample covariance matrix. (Rothman et al., 2009)
    - Off-diagonal entries are shrunken towards 0 by an additive factor c, until they reach 0.
  - Proposed method with the weight matrix $\boldsymbol{P}_{ij} = 1$ for $i \neq j$, $\boldsymbol{P}_{ii} = 0$
    - Equal penalties for all off-diagonal entries.
  - Proposed method with the weight matrix $\boldsymbol{P}_{ij} = \boldsymbol{S}_{ij}^{-1}$ for $i \neq j$, $\boldsymbol{P}_{ii} = 0$
    - Stronger penalties for entries with small sample covariances.

Simulation Setup

- Three methods to consider:
  - Soft-thresholded sample covariance matrix. (Rothman et al., 2009)
    - Off-diagonal entries are shrunken towards 0 by an additive factor c, until they reach 0.
  - Proposed method with the weight matrix $\boldsymbol{P}_{ij} = 1$ for $i \neq j$, $\boldsymbol{P}_{ii} = 0$
    - Equal penalties for all off-diagonal entries.
  - Proposed method with the weight matrix $\boldsymbol{P}_{ij} = \boldsymbol{S}_{ij}^{-1}$ for $i \neq j$, $\boldsymbol{P}_{ii} = 0$
    - Stronger penalties for entries with small sample covariances.
- 5 different structures of $\boldsymbol{\Sigma}$.
- $n = 200$, $p = 100$.
- 10 repetitions (this is not a typo!)

Runtime

- Intel 4^{th} generation Core i7 processor (2013), 2.0GHz.
- Parallel computing using 2 cores.

Runtime

- Intel 4^{th} generation Core i7 processor (2013), 2.0GHz.
- Parallel computing using 2 cores.
- 50 candidate shrinkage/tuning parameters.
- Use 10-fold CV to choose the shrinkage/tuning parameters.

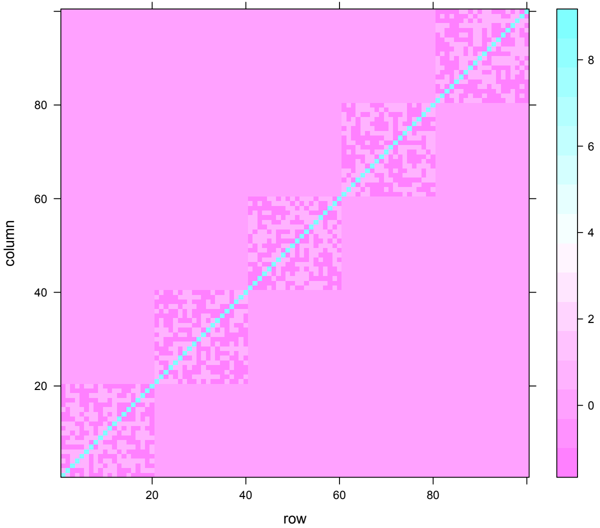
Runtime

- Intel 4^{th} generation Core i7 processor (2013), 2.0GHz.
- Parallel computing using 2 cores.
- 50 candidate shrinkage/tuning parameters.
- Use 10-fold CV to choose the shrinkage/tuning parameters.
- Time for 1 model (500 model fits):

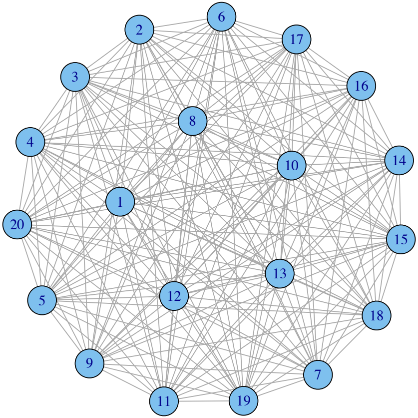
Method	Runtime
Thresholding of Sample Covariance	2 sec
Maximum ℓ_1-Penalized Likelihood	90 min

- The whole simulation will take around 150 hours.

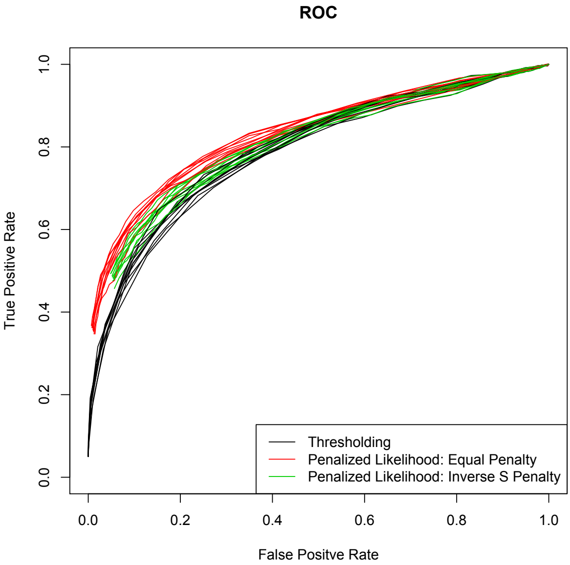
Cliques Covariance Structure



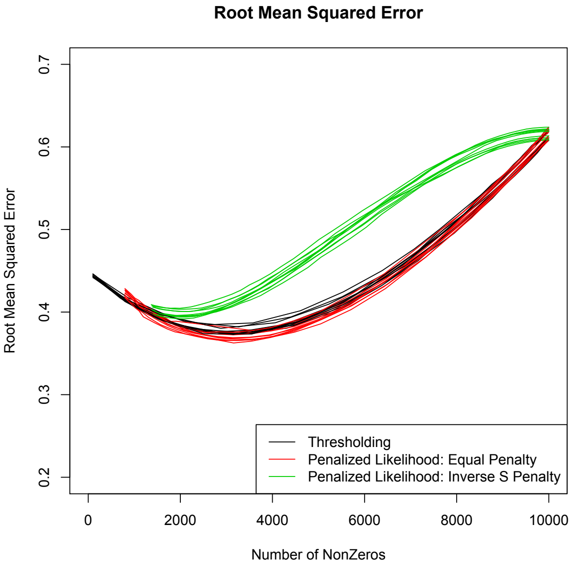
Cliques Covariance Structure



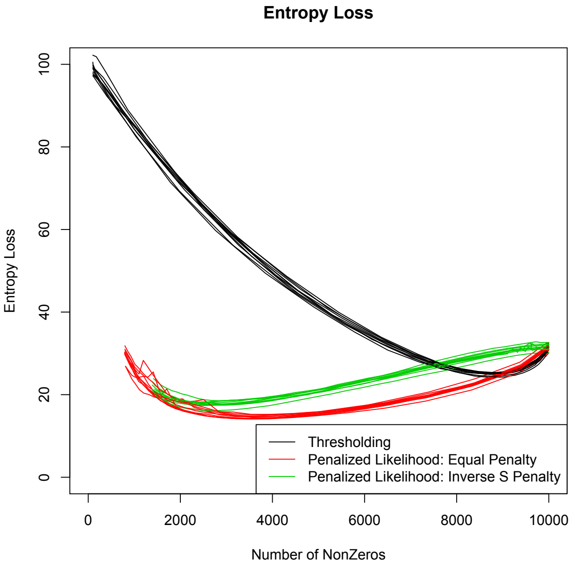
Cliques Covariance Structure



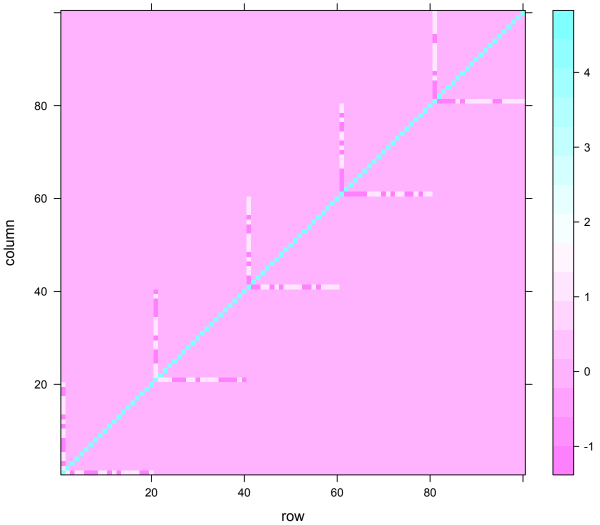
Cliques Covariance Structure



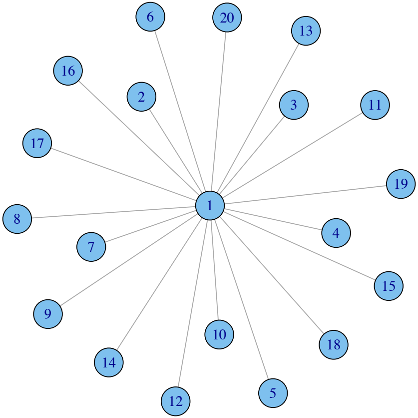
Cliques Covariance Structure



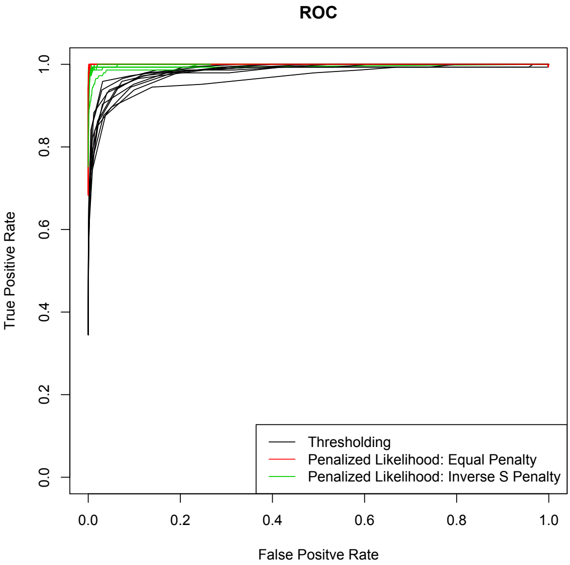
Hubs Covariance Structure



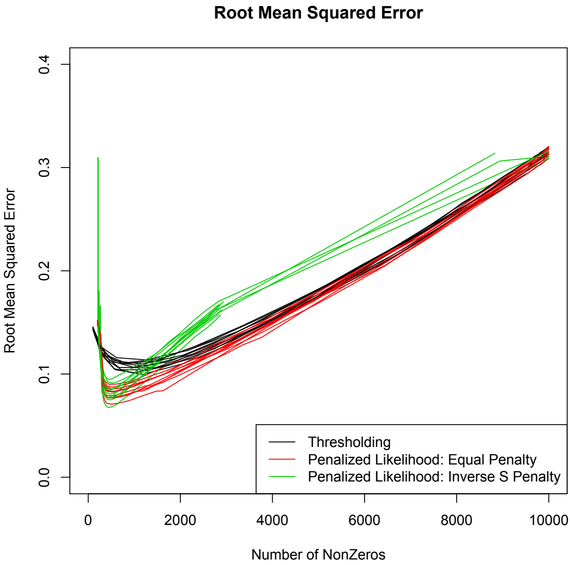
Hubs Covariance Structure



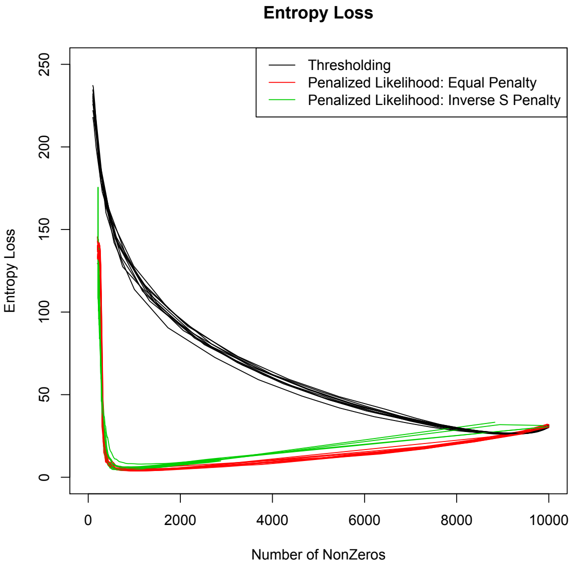
Hubs Covariance Structure



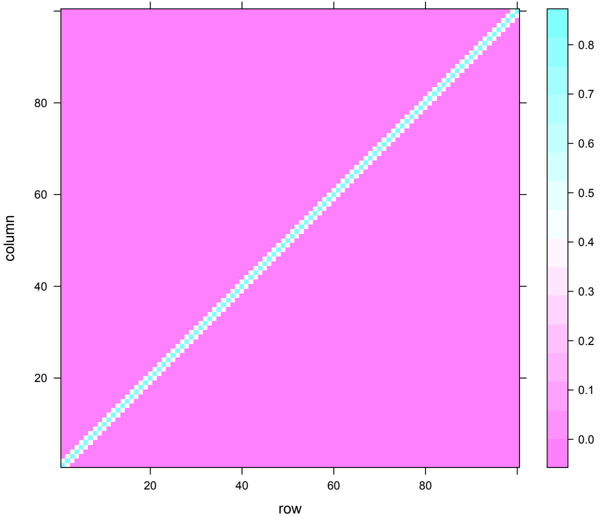
Hubs Covariance Structure



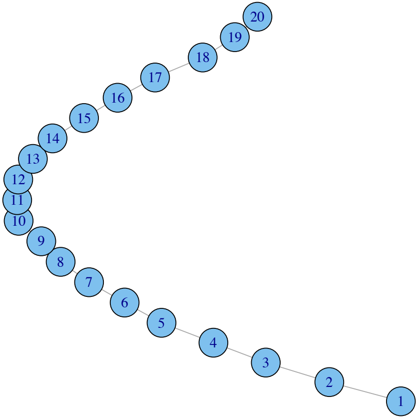
Hubs Covariance Structure



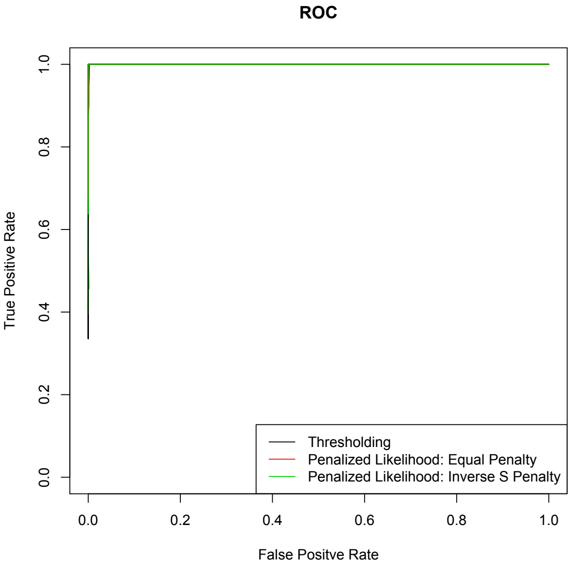
MA(1) Covariance Structure



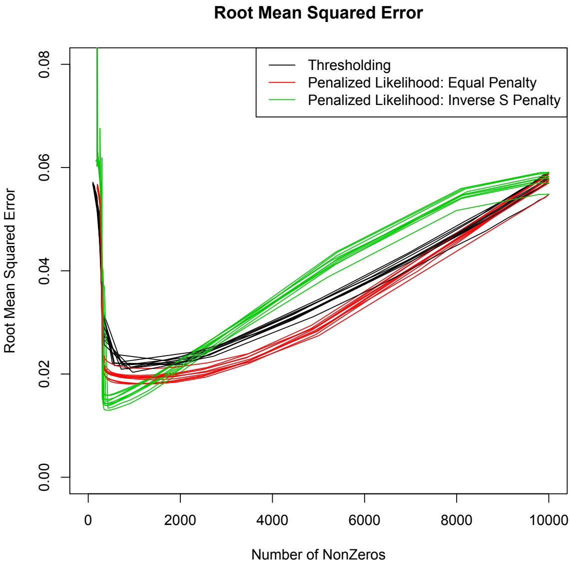
MA(1) Covariance Structure



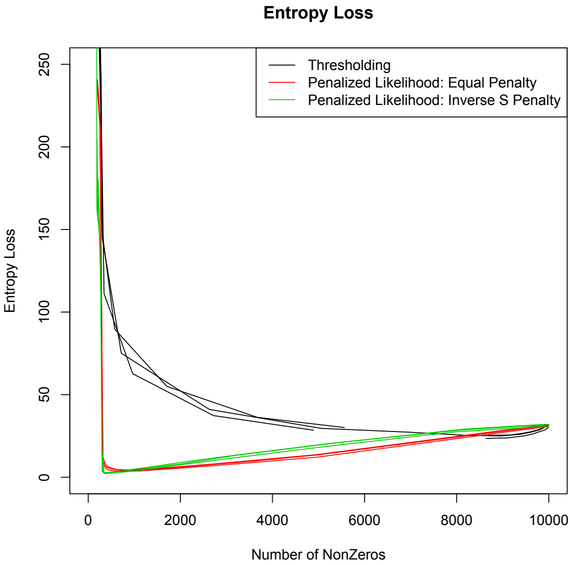
MA(1) Covariance Structure



MA(1) Covariance Structure



MA(1) Covariance Structure



Summary

- Majorize-Minimization (MM) Algorithm
- Difference of Convex Functions (DC) Programming
- The runtime of the proposed method is discouraging
- The method may work well when the covariance matrix is very sparse