Adaptive Estimation of Graphical Models under Total Positivity

***Abstract*—This document is a report of the passage Adaptive Estimation of Graphical Models under Total Positivity and my own try to recover the code in this paper.**

I. Introduction

The paper Adaptive Estimation of Graphical Models under Total Positivity illustrates a new method to estimate the graphical models under total positivity, using F-score to estimate the performance of the method. To be more exact, it proposes a new method to estimate the precision matrix of Gaussian graphical model, based on both M-matrix and diagonally dominant M-matrix by solving a weighted l_1 regularization problem. To solve the regularization, it uses the gradient projection method and provide the anlysis of the estimation error.

II. Problem notation and formulation

A. Problem notation

1) M matrix: An M matrix is a square matrix where all the off-diagonal elements are non-positive, and all the row sums are non-negative.

2) Precision matrix: The precision matrix is the inverse of the covariance matrix. The covariance matrix represents the covariances and variances between different random variables in a multivariate distribution. To be noted, precision matrix in Gaussian Graphical models is used to represent conditional independence relationships between variables in a graphical form, helping in visualizing and understanding the structure of the multivariate distribution.

3) Gaussian Graphical Model: A Gaussian Graphical Model is a type of continuous probabilistic graphical model. Suppose $X = (X_1, X_2, ..., X_n)$ which is an N-dimensional random vector following a multivariate Gaussian distribution, where μ represents the mean vector and Σ represents the covariance matrix. Let $G = (V, E)$ be an undirected graph consisting of the vertex set $V = (X_1, X_2, ..., X_n)$ and the edge set. A Gaussian graphical model corresponds to a multivariate Gaussian distribution.

4) Precision matrix in Gaussian Graphical Model: Suppose σ_{ij} stands for the ith row and jth column in the precision matrix. We can get the following settings: if $\sigma_{ij} = 0$, then X_i and X_j are conditionally independent given all other variables, and node i and j are not connected in the graph; if $\sigma_{ij} \neq 0$, then X_i and X_j are conditionally dependent given all other variables, and node i and j are connected in the graph.

B. Problem formulation

To estimate the precision matrix under MTP_2 Gaussian graphical model, the paper proposes the formulation of penalized Gaussian maximum likelihood estimation:

$$minimize_{\Theta\in S} - logdet(\Theta) + tr(\Theta\hat{\Sigma}) + \sum_{i\neq j} h_\lambda(|\Theta_{ij}|)$$

where h_λ is a sparsity penalty function, such as the l_1-norm
Proof:
Suppose $Z_1, Z_2, ..., Z_n - N_d(\mu, \Sigma)$, $\Sigma = \Theta^{-1}$
$f(Z) = (2\pi)^{-\frac{d}{2}}|\Sigma|^{-\frac{1}{2}}exp(-\frac{(Z-\mu)^T\Sigma^{-1}(Z-\mu)}{2})$, so we can get:
$f(Z_1, Z_2, ..., Z_n) = \Pi_{i=1}^n(2\pi)^{-\frac{d}{2}}|\Sigma|^{-\frac{1}{2}}exp(-\frac{(Z_i-\mu)^T\Sigma^{-1}(Z_i-\mu)}{2}) = \Pi_{i=1}^n(2\pi)^{-\frac{d}{2}}|\Theta^{-1}|^{-\frac{1}{2}}exp(-\frac{(Z_i-\mu)^T\Theta(Z_i-\mu)}{2})$
Its log-likelihood function is:
$L(\Theta) \propto \sum_{i=1}^n \frac{1}{2}log|\Theta| - exp(-\frac{(Z_i-\mu)^T\Theta(Z_i-\mu)}{2})$
$\propto -\frac{1}{n}[\sum_{i=1}^n(Z_i-\mu)^T\Theta(Z_i-\mu) - nlog|\Theta|]$
Adding the trace on the formulation, the minimization problem then becomes:
$min\frac{\sum_{i=1}^n tr((Z_i-\mu)^T\Theta(Z_i-\mu))}{n} - log|\Theta|$
Centralize the Z_i it becomes: $\frac{\sum_{i=1}^n tr(Z_iZ_i^T\Theta)}{n} - log|\Theta|$
After adding the penalty function it becomes the formulation above.

To solve the minimization problem, I derive the formulation and get: $\frac{1}{n}\sum_{i=1}^n Z_iZ_i^T - \Theta^{-1} = 0$. Solving the formulation: $\hat{\Theta}_{MLE} = (\frac{1}{n}\sum_{i=1}^n Z_iZ_i^T)^{-1}$ which is the reason how the sample covariance matrix comes from.

It also set two cases for S:

$$M^P = \{\Theta \in S^P_{++}|\Theta_{ij} = \Theta_{ji} \leq 0, \forall i \neq j\}$$

$$M^P_D = \{\Theta \in S^P_{++}|\Theta_{ij} = \Theta_{ji} \leq 0, \forall i \neq j, \Theta 1 \geq 0\}$$

M^P represents the set of all symmetric positive definite M-matrices, and M^P_D adds constraints $\Theta 1 \geq 0$ leading Θ to be a diagonally dominant M-matrix.

III. Proposed Method

A. Adaptive Estimation

In this section, the paper transforms the minimization function to an adaptive form where the model adjusts its parameters or structure dynamically based on the characteristics of the data it is processing. The minimization problem becomes as follows:

$$minimize_{\Theta\in S} - logdet(\Theta) + tr(\Theta\hat{\Sigma}) + \sum_{i\neq j} \lambda_{ij}^{(k)}|\Theta_{ij}|$$

where $\lambda_{ij}^{(k)} = p_\lambda(|\hat{\Theta}_{ij}^{(k-1)}|)$ with p_λ the weight-updating function and $\hat{\Theta}_{ij}^{(k-1)}$ is obtained at the (k-1) stage.

B. Gradient Projection Method

The gradient projection method is an approach for finding the optimal solution to constrained nonlinear optimization problems using the projection technique, particularly effective for solving nonlinear optimization problems with linear constraints. It begins with an initial feasible solution and utilizes the constraints to determine the projection of the gradient onto the boundary of the convex constraint set, which guides the determination of the next search direction and step length. After each search, a check is performed until the desired level of accuracy is achieved.

Because in M matrix, all the off-diagonal elements are non-positive, so the problem can be simplified to:

$$minimize_{\Theta \in S} - logdet(\Theta) + tr(\Theta\hat{\Sigma}) - \sum_{i \neq j} \lambda_{ij}^{(k)} \Theta_{ij}$$

The union S can be written as $S = S_d \cap S_{++}^p$ where the frontier one is closed set and the second is open set. So the updating formula can be written as:

$$\Theta_{t+1} = P_{S_d}(\Theta_t - \eta_t \nabla f(\Theta_t))$$

where P_{S_d} stands for the projection onto the set S_d with respect to the Frobenius norm.

Using the line search precedure, the paper try the step size $\eta_t \in \sigma\{\beta^0, \beta^1, \beta^2, ...\}$ with $\beta \in (0,1)$ and $\sigma > 0$ until we find the smallest $m \in N$ with $\eta_t = \sigma\beta^m$, satisfying:

$$f(\Theta_{t+1}) \leq f(\Theta_t) - \alpha\eta_t ||G_{\frac{1}{\eta_t}}(\Theta_t)||_F^2$$

where $\alpha \in (0,1)$, and $G_{\frac{1}{\eta_t}}(\Theta_t) = \frac{1}{\eta_t}(\Theta_t - \Theta_{t+1})$

C. Computation of Projections

In this section the paper compute the projection P_{S_d} for both cases of estimating M-matrices and diagonally dominant M-matrices.

1) Estimation of M matrices: In M matrices, the closed set S_d becomes:

$$S_d = \{\Theta \in R^{p \times p} | \Theta_{ij} = \Theta_{ji} \leq 0, \forall i \neq j\}$$

$$S_d = S_A \cap S_B$$

where $S_A = \{\Theta \in R^{p \times p} | \Theta_{ij} = \Theta_{ji}, \forall i \neq j\}$ and $S_B = \{\Theta \in R^{p \times p} | \Theta_{ij} \leq 0, \forall i \neq j\}$. It is obvious that if Θ_t is symmetric, then the $P_{S_B}(\Theta_t - \eta_t \nabla f(\Theta_t))$ is also symmetric. So the updating formula becomes:

$$\Theta_{t+1} = P_{S_B}(\Theta_t - \eta_t \nabla f(\Theta_t))$$

Then compute the P_{S_B}:

$$[P_{S_B}(X)]_{ij} = \begin{cases} min(X_{ij}, 0) & \text{if } i \neq j \\ X_{ij} & \text{if } i = j \end{cases} \qquad (1)$$

2) Estimation of diagonally dominant M matrices: In diagonally dominant M matrices, the closed set S_d becomes:

$$S_d = \{\Theta \in R^{p \times p} | \Theta_{ij} = \Theta_{ji} \leq 0, \forall i \neq j, \Theta 1 \geq 0\}$$

$$S_d = S_A \cap S_C$$

Because we can't guarantee symmetry, so the updating formula is still:

$$\Theta_{t+1} = P_{S_A \cap S_C}(\Theta_t - \eta_t \nabla f(\Theta_t))$$

Then the minimizer is: $P_{S_A \cap S_C} = argmin_{X \in S_A \cap S_C} ||X - Y||_F^2$

The paper uses Dykstra's projection algorithm to solve this minimization problem. Using the algorithm, set the $P_{S_A}(Y) = (Y + Y^T)/2$, $P_{S_C}(Y) = argmin_{X \in S_C} \frac{1}{2}||X - Y||_F^2$, the paper proves the optimal solution as follows:

- If $y_r \geq -\sum_{i \in [p] \backslash r} min(y_i, 0)$, then $\hat{x_r} = y_r$ and $\hat{x_i} = min(y_i, 0)$ for $i \neq r$
- If $y_r < -\sum_{i \in [p] \backslash r} min(y_i, 0)$, then $\hat{x_r} = y_r + \rho$ and $\hat{x_i} = min(y_i + \rho, 0)$ for $i \neq r$, where ρ satisfies $\rho + \sum_{i \in [p] \backslash r} min(y_i + \rho, 0) + y_r = 0$

To solve the ρ, the paper also proposes a function:

$$g(\rho) = \rho + \sum_{i \in [p] \backslash r} min(y_i + \rho, 0) + y_r = 0$$

To make the function zero, it follow the procedure as follows:

- Sort y_r and obtain the sorted version $\widetilde{y}_r$, where $[\widetilde{y}_r]_1 \leq [\widetilde{y}_r]_2 \leq ... \leq [\widetilde{y}_r]_{p-1}$
- Find $M := max_{1 \leq m \leq p-1}\{m | \frac{y_r + \sum_{i=1}^m [\widetilde{y}_r]_i}{m+1} > [\widetilde{y}_r]_m\}$
- Compute $\rho = \frac{-y_r - \sum_{m=1}^M [\widetilde{y}_r]_i}{M+1}$

IV. Analysis of Estimation error

In this section, the paper provides an analysis of the estimation error in the proposed adaptive multiple-stage estimation method. The estimation error between the estimated precision matrix and the underlying precision matrix can be upper bounded by two terms: optimization error and statistical error. The optimization error decreases at a linear rate across iterative stages, indicating the progressive refinement of the estimate. In contrast, the statistical error remains independent of the number of iterations and does not decrease, capturing the intrinsic statistical rate. The estimation error is primarily dominated by the statistical error. It also provides Theorem 4.4, which establishes a probabilistic upper bound on the estimation error in terms of the sample size and other parameters of the method.

V. Results

The paper presents a novel adaptive multiple-stage estimation method for estimating M-matrices as precision matrices in Gaussian graphical models. The proposed method outperforms state-of-the-art methods in terms of estimation accuracy and computational efficiency. It provides a detailed description of the problem of estimating M-matrices, the proposed method, and its theoretical analysis,also including numerical experiments on synthetic and real-world datasets to demonstrate the effectiveness of the proposed method.

VI. Code

We first review the experiment setting in the paper, and set the preparations for the algorithm:

How to set the experiment setting

1) Construct a graph structure randomly
2) Get a weight that is uniformly sampled from U(2,5), then set it to edge matrix. The edge matrix A is our weighted adjacency matrix. In my code, it is the function $generate\_weighted\_adjacency\_matrix()$
3) Using the formula $\Theta = \delta I - A$ where $\delta = 1.05\lambda_{max}A$ to get the based precision matrix. In my code,it is the function $generate\_precision\_matrix()$
4) Based on the formula $\Theta^* = E\Theta E$, I want to find matrix E making the $(\Theta^*)^{-1}$ has unit diagonal elements. In my code, it is the function get_Theta, in which exists a function get_E
5) Finally, based on the Θ^*, I can generate n independent and identically distributed observations conforming to the N(0, $(\Theta^*)^{-1}$), the sample covariance matrix's proof I have given in the before.

How to set the algorithm I follow the steps in Algorithm 1 and realize the algorithm.

1) The input of the algorithm is the sample covariance matrix $\hat{\Sigma}$,weight of all the variables of λ_{ij}:$\Lambda^{(k)}$, parameter σ,α, β
2) First compute the gradient function, it is computed as the formula $\nabla f(\Theta_t) = -\Theta_t^{-1} + \hat{\Sigma} - \Lambda^{(k)}$. In my code, it is the function $compute\_gradient()$
3) Set the m to 0 as the initial value, which represent the whole iteration number.
4) Then the main loop of the algorithm, update the Θ according to formula $\Theta_{t+1} = P_{S_d}(\Theta_t - \sigma\beta^m \nabla f(\Theta_t))$
5) Add m by 1, and repeat the step 3 and 4 until $\Theta_{t+1} \in S^p_{++}$ and $f(\Theta_{t+1}) \leq f(\Theta_t) - \alpha\eta_t ||G_{\frac{1}{\eta_t}}(\Theta_t)||_F^2$. In my code, it is the function $perform\_gradient\_projection()$

It is called line search method, used to determine an appropriate step size or learning rate along the search direction(negative gradient direction) at each iteration. In this algorithm it is set as the formula $\eta_t \in \sigma\{\beta^0, \beta^1, \beta^2, ...\}$ with $\beta \in (0, 1)$. The line search method aims to find the optimal step size that minimizes the objective function along the search direction. It performs a backtracking procedure, where it iteratively reduces the step size until it finds a step size that satisfies the desired conditions.

How to evaluate the algorithm

The paper use five evaluation to evaluate the performance of the algorithm, they are: true positive rate(TPR), false positive rate(FPR), true negative rate(TNR), false positive rate(FPR), F-score, estimation error and the modularity value.

Suppose TP, FP, TN and FN denote the number of true positives, false positives, true negatives and false negatives, respectively. Then, the TPR, FPR, TNR and FNR are defined as follows:

$$TPR = \frac{TP}{TP + FN}$$

$$FPR = \frac{FP}{FP + TN}$$

$$F - score = \frac{2TP}{2TP + FP + FN}$$

In my code I use the function $calculate\_evaluation\_metrics()$

Issues about penalty function

The paper uses the SCAD penalty function, but the definition of SCAD penalty function varies. And during the setting of the penalty function, it has two parameter named threshold and alpha which can restrict the penalty. The threshold parameter is used to determine the transition point between linear and nonlinear penalties. A smaller threshold value will result in more linear penalties, while a larger threshold value will result in more nonlinear penalties.

The hyperparameter α is used to balance the degree of linear and nonlinear penalties. A smaller value of α will lead to more nonlinear penalties while a larger one will result in penalties that are closer to linear.

So the adjusting of this two parameter will also affect the result a little, it still needs more trial to find the best parameter for this algorithm. I personally set a common value $\alpha = 3.7$ and threshold as 0.8.

Result I set the parameter $\sigma = 0.05$, $\alpha = 0.003$, $\beta = 0.5$. I find that with the change of the σ, α, the result will change to some extent. And the set of the parameter β can't be too small because in the loop function there is a β^m, when m is large it will become converging to zero, which will make the division become zero, leading to the math error. As I expected, I think some of my results are not good enough, adjusting the parameter may get a better result.

The result of the picture visualized are as follows: there are four pictures, 1st is the grid graph, 2nd is the line graph, 3rd is the barabasi-albert graph model, the 4th is the stochastic block model. Since they are generated randomly, each trial of running the code will result different pictures.

And my result of the code is as follows:

The estimation error of the grid graph is 0.44, the line graph is 0.48, the barabasi-albert graph model is 0.48, the stochastic block model is 0.30. I also compared with the SLTP method and GLASSO method. The TPR is 0.95, the FPR is 0.003. I find that with the iteration increasing the result will become better. In my code I only iterate 100 times and 10 times of the big loop, I think more will get better result. The code is given in the test.ipynb.

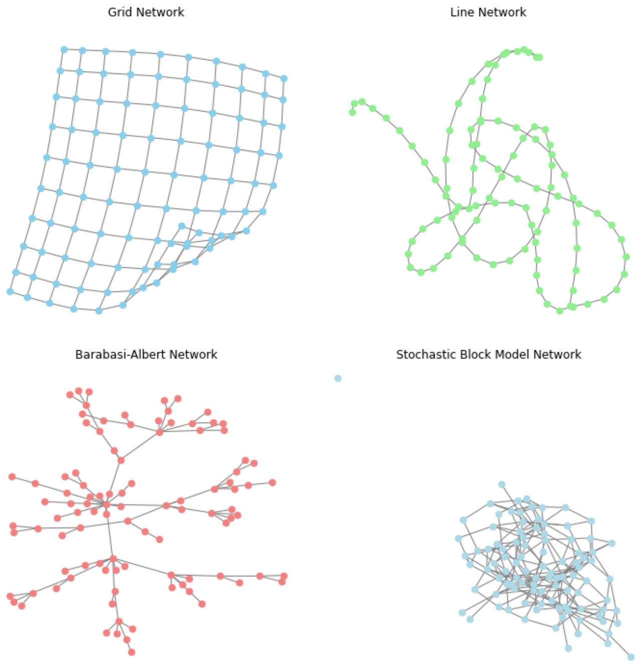

Fig. 1. The result of the picture