Graph Learning Algorithms: Work Plan I

Convex Group, HKUST

August 8, 2018

1 Problem Statement

The graph learning problem can be viewed from a probabilistic perspective under the assumption that the observed samples obey multivariate Gaussian distribution. Specifically, suppose we have k samples independently drawn from a n-variate Gaussian distribution: $\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(k)} \sim N(\mathbf{0}, \mathbf{\Sigma})$, where $\mathbf{\Sigma}$ is the covariance matrix, and $\mathbf{y}^{(i)} \in \mathbb{R}^n, i \in \{1, 2, \ldots k\}$. Let $\mathbf{S}$ denote the second moment matrix:

$$\mathbf{S} := \frac{1}{k} \sum_{i=1}^{k} \mathbf{y}^{(i)} \mathbf{y}^{(i)T}. \tag{1}$$

The graph estimation can be formulated into precision matrix (i.e. inverse covariance matrix $\mathbf{\Sigma}$) estimation from $\mathbf{S}$ by the following optimization [1–3]

$$\min_{\mathbf{\Theta} \succ \mathbf{0}} - \log\det(\mathbf{\Theta}) + \operatorname{tr}(\mathbf{\Theta}\mathbf{S}) + \alpha \left\|\mathbf{\Theta}\right\|_1, \tag{2}$$

where $\mathbf{\Theta} \in \mathbb{R}^{n \times n}$ and corresponds to the maximum likelihood estimation of precision matrix for multivariate Gaussian distribution. The above problem has been extensively studied in detail. Recently, the authors in [2] have solved the above problem (2) under Laplacian constraint, which implies that the target precision matrix is positive semi-definite and singular, with off-diagonal entries non-positive:

$$\mathcal{S}_{\mathbf{\Theta}} = \left\{ \mathbf{\Theta} | \mathbf{\Theta} \succeq 0, \mathbf{\Theta}_{ij} = \mathbf{\Theta}_{ji} \leq 0 \text{ for } i \neq j, \mathbf{\Theta} \cdot \mathbf{1} = \mathbf{0} \right\}, \tag{3}$$

where $\mathbf{0}$ and $\mathbf{1}$ denote the constant zero and one vectors.

On a high level, within this framework we associate each variable $\{\mathbf{y}_i\}_{i=1}^n$ to a node, and based on the data statistics we try to infer whether the variables are related to each other or not. The association of these variables is realized as some form of matrix often called as graph matrix. If the entry in the ij-th element of the graph matrix is non-zero then this implies the nodes are connected(variables are related); otherwise they are disconnected(independent).

1.1 Graph Learning with K-components

In this project, our objective is to learn the graph of Kcomponents. This will provide a framework for applications like (clustering, graph clustering, sparse graph clustering, sparse graph, sparsely connected graph). We introduce a linear operator $\mathcal{L}$ which transform a vector $\mathbf{w} \in \mathbb{R}^{\frac{n(n-1)}{2}}$ to a matrix $\mathcal{L}\mathbf{w}$ which satisfies $[\mathcal{L}\mathbf{w}]_{ij} = [\mathcal{L}\mathbf{w}]_{ji}, \text{for}\, i \neq j$ and $[\mathcal{L}\mathbf{w}] \cdot \mathbf{1} = \mathbf{0}$. We denote the adjoint and Moore-Penrose pseudoinverse of $\mathcal{L}$ by $\mathcal{L}^\star$ and $\mathcal{L}^\dagger$, respectively, which satisfy $\langle \mathcal{L}\mathbf{w}, \mathbf{Y} \rangle = \langle \mathbf{w}, \mathcal{L}^\star \mathbf{Y} \rangle$ for $\forall \mathbf{w} \in \mathbb{R}^{\frac{n(n-1)}{2}}$ and $\mathbf{Y} \in \mathbb{R}^{n \times n}$, and $\mathcal{L}^\dagger \mathcal{L} = \mathcal{I}$. (An example about $\mathcal{L}$ is given in the Appendix **??** for better understanding). Define $\mathbf{H} = \alpha(2\mathbf{I} - \mathbf{1}\mathbf{1}^T)$ and $\mathbf{K} := \mathbf{H} + \mathbf{S}$

Since the sign of $\boldsymbol{\Theta}$ is fixed by the constraints $\boldsymbol{\Theta}_{ji} \leq 0\, \text{for}\, i \neq j$ and $\boldsymbol{\Theta}_{ji} \geq 0\, \text{for}\, i = j$, the regularization term $\alpha \left\|\boldsymbol{\Theta}\right\|_1$ can be written by $\text{tr}\left(\boldsymbol{\Theta}\mathbf{H}\right)$, where $\mathbf{H} = \alpha(2\mathbf{I} - \mathbf{1}\mathbf{1}^T)$. With the definitions defined above, now the problem of graph learning with K-components can be expressed as follows: Mathematically, our objective is to solve the following optimization problem

$$\begin{array}{ll} \underset{\mathbf{w}, \boldsymbol{\Lambda}, \mathbf{U}}{\mathsf{minimize}} & -\log\det(\boldsymbol{\Lambda}) + \text{tr}\left(\mathbf{K}\mathcal{L}\mathbf{w}\right) + \frac{\beta}{2}\|\mathcal{L}\mathbf{w} - \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^T\|_F^2, \\ \mathsf{subject\ to} & \mathbf{w} \geq \mathbf{0},\ \boldsymbol{\Lambda} \in \mathcal{S}_{\boldsymbol{\Lambda}},\ \mathbf{U}^T\mathbf{U} = \mathbf{I}. \end{array} \tag{4}$$

where we can continuously increase β to get better approximation between $\mathcal{L}\mathbf{w}$ and $\mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^T$; $\boldsymbol{\Lambda} \in \mathbb{R}^{(n-K)\times(n-K)}$ is a diagonal matrix in which K is the number of components in topology.

Definition of $\mathcal{L}$

In this subsection, we will give an example of how to define the operator $\mathcal{L}$. Consider a vector $\mathbf{x} \in \mathbb{R}^{\frac{n(n-1)}{2}}$. For $n = 4$, $\mathbf{x} \in \mathbb{R}^6$ which contains element as $\mathbf{w} = [x_1, x_2, x_3, x_4, x_5, x_6]$. Now the operation of $\mathcal{L}$ on the weight vector $\mathbf{x}$ will lead to a matrix $\Theta := \mathcal{L}\mathbf{w} \in \mathbb{R}^{N \times N}$ as follows:

$$\mathcal{L}\mathbf{w} = \begin{bmatrix} \sum_{i=1,2,3} x_i & -x_1 & -x_2 & -x_3 \\ -x_1 & \sum_{i=1,4,5} x_i & -x_4 & -x_5 \\ -x_2 & -x_4 & \sum_{i=2,4,6} x_i & -x_6 \\ -x_3 & -x_5 & -x_6 & \sum_{i=3,5,6} x_i \end{bmatrix}. \tag{5}$$

Definition of $\mathcal{L}^\star$

Let us understand this with an example for $n = 4$, consider a matrix $\mathbf{Y} \in \mathbb{R}^{n\times n}$ with elements as follows,

$$\mathbf{Y} = \begin{bmatrix} y_{11} & y_{12} & y_{13} & y_{14} \\ y_{21} & y_{22} & y_{23} & y_{24} \\ y_{31} & y_{32} & y_{33} & y_{34} \\ y_{41} & y_{42} & y_{43} & y_{44} \end{bmatrix}. \tag{6}$$

The operator $\mathcal{L}^\star$ when applied over a matrix $\mathbf{Y} \in \mathbb{R}^{n\times n}$, returns a vector $\tilde{\mathbf{x}} := \mathcal{L}^*\mathbf{Y} \in \mathbb{R}^{\frac{n(n-1)}{2}}$, with elements as $\tilde{\mathbf{x}} = [\tilde{x}_1, \tilde{x}_2, \ldots, \ldots, \tilde{x}_{\frac{n(n-1)}{2}}]$. Where each elements of the vector can be expressed as follows:

$$\tilde{x}_k := [\mathcal{L}^*\mathbf{Y}]_k = -Y_{i,i} + Y_{i,j} + Y_{j,i} - Y_{j,j},$$

where $k = (j-1)n + i - j - \sum_{s=0}^{j-1} s$, for any $i > j$ with $i \in \{2, 3, \cdots, n\}$ and $j \in \{1, 2, \cdots, n-1\}$. In this example, we have

$$\mathcal{L}^*\mathbf{Y} = \begin{bmatrix} -y_{11} + y_{21} + y_{12} - y_{22} \\ -y_{11} + y_{31} + y_{13} - y_{33} \\ -y_{11} + y_{41} + y_{14} - y_{44} \\ -y_{22} + y_{32} + y_{23} - y_{33} \\ -y_{22} + y_{42} + y_{24} - y_{44} \\ -y_{33} + y_{43} + y_{34} - y_{44} \end{bmatrix}. \tag{7}$$

1.2 Algorithm Design

We propose a block-coordinate descent algorithm in Alg. 1 to solve the optimization (4). In each iteration, we update one variable at a time with the others fixed.

1.3 w-update

Define

$$f(\mathbf{w}) := \frac{1}{2}\mathbf{w}^T\mathbf{M}\mathbf{w} - \mathbf{c}^T\mathbf{w} \tag{8}$$

Where, $\mathbf{c} = \mathcal{L}^\star(\mathbf{U}\mathbf{\Lambda}\mathbf{U}^T - \beta^{-1}\mathbf{K})$ and $\mathbf{M}$ is a constant matrix which satisfies $\mathcal{L}^\star\mathcal{L}\mathbf{b} = \mathbf{M}\mathbf{b}$ for any $\mathbf{b} \in \mathbb{R}^{\frac{n(n-1)}{2}}$. Denote, $\mathbf{D} = \lambda_{\max}(\mathbf{M})\mathbf{I}$ is a diagonal matrix. With the above definition, the closed form update for $\mathbf{w}^{(k+1)} = \max(a_i, 0)$ can be obtained as:

$$\left[\mathbf{w}^{(k+1)}\right]_i = \left(\left[\mathbf{w}^{(k)} - \frac{1}{\lambda_{\max}(\mathbf{M})}\nabla f(\mathbf{w}^{(k)})\right]_i\right)^+ \tag{9}$$

Where, $[\mathbf{w}]_i = w_i$, $\forall\, i = 1, \dots, n(n-1)/2$ and $(a)^+ := \max(a, 0)$.

1.3.1 Update U

To update $\mathbf{U}$, we solve the following sub-problem,

$$\underset{\mathbf{U}}{\mathsf{minimize}} \qquad \frac{\beta}{2}\|\mathcal{L}\mathbf{w} - \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T\|_F^2, \quad \mathsf{subject\ to}\ \mathbf{U}^T\mathbf{U} = \mathbf{I}. \tag{10}$$

The problem (10) is a least square optimization on the Stiefel manifold. The problem (10) is equivalent to

$$\underset{\mathbf{U}}{\mathsf{maximize}} \qquad \mathrm{tr}(\mathbf{U}\mathbf{\Lambda}\mathbf{U}^T\mathcal{L}\mathbf{w}), \quad \mathsf{subject\ to} \quad \mathbf{U}^T\mathbf{U} = \mathbf{I}. \tag{11}$$

where $\mathbf{\Lambda}$ and $\mathcal{L}\mathbf{w}$ are positive semidefinite. The closed form solution of the problem (11) is the principle eigen vectors of the matrix $\mathcal{L}\mathbf{w}$, i.e., $\mathbf{U}^* = \bar{\mathbf{U}}_{(:,K+1:n)}$, where $\bar{\mathbf{U}}$ admits eigenvalue decomposition $\mathcal{L}\mathbf{w} = \bar{\mathbf{U}}\bar{\mathbf{\Lambda}}\bar{\mathbf{U}}^T$. Here $\mathbf{U}^* \in \mathbb{R}^{n\times n-K}$ and $\mathcal{L}\mathbf{w} \in \mathbb{R}^{n\times n}$.

1.3.2 Update Λ

$$\underset{\mathbf{\Lambda}\in\mathcal{S}_{\mathbf{\Lambda}}}{\mathsf{minimize}} \quad -\log\det(\mathbf{\Lambda}) + \frac{\beta}{2}\|\mathcal{L}\mathbf{w} - \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T\|_F^2. \tag{12}$$

The optimization (12) can be rewritten as

$$\underset{\mathbf{\Lambda}\in\mathcal{S}_{\mathbf{\Lambda}}}{\mathsf{minimize}} \quad -\log\det(\mathbf{\Lambda}) + \frac{\beta}{2}\|\mathbf{U}^T(\mathcal{L}\mathbf{w})\mathbf{U} - \mathbf{\Lambda}\|_F^2. \tag{13}$$

Note that the Λ is forced to be diagonal matrix and thus (13) can be further written as

$$\underset{\mathbf{\Lambda}\in\mathcal{S}_{\mathbf{\Lambda}}}{\mathsf{minimize}} \quad -\sum_{i=1}^{q}\log\lambda_i + \frac{\beta}{2}\|\boldsymbol{\lambda} - \mathbf{d}\|_2^2 \tag{14}$$

$$\text{where, } \mathcal{S}_{\mathbf{\Lambda}} := \{\alpha_1 \le \lambda_{K+1} \le \lambda_{K+2} \le \cdots \le \lambda_n \le \alpha_2\} \tag{15}$$

where $\mathbf{d} = \mathrm{Diag}(\mathbf{U}^T(\mathcal{L}\mathbf{w})\mathbf{U})$. Note that the first K lambda's are zero $\{\lambda_i = 0\}_{i=1}^K$.

We summarize the algorithm for obtaining the graph with K-components. $\mathbf{S}$(Sample covariance matrix), K(Number of Desired Components), and α_1, α_2(Constraints for problem (12))

Algorithm 1 Graph Learning with K-components

```
Input: $\mathbf{S}$,$K$ $\alpha_1, \alpha_2$
1: repeat
2:     repeat
3:         Update $\mathbf{w}^{(k)}$ by solving (9);
4:         Update $\boldsymbol{U}^{(k)}$ by solving (11);
5:         Update $\boldsymbol{\Lambda}^{(k)}$ by solving (14);
6:     until Inner loop convergence
7:     $\beta \leftarrow \rho\beta$ with $\beta > 1$;
8: until Outer loop convergence
9: return $\mathcal{L}\mathbf{w}$
```

Important Points.

- The algorithm will lead to a natural framework for clustering, the value of K decides the number of components/clusters.
- The value of α_1, α_2 are problem dependent. The proper tuning of these values is crucial for better performance.
- For simulations, we will borrow examples and applications from the contemporary works in [1–3].

Synthetic Experiments

Due to the problem structure, designing an even experiment with synthetic data and computing metrics to evaluate graph learning performance is non-trivial. Therefore, we will instead consider scenarios where we have an input empirical covariance matrix obtained from the precision matrix (Laplacian) matrix with K-components. Precisely, we will follow the given steps:

1. Construct a square matrix $\mathbf{L} \in \mathbb{R}^{n\times n}$, from the set (3). The set is also equivalent as $\{\mathbf{L} : \mathbf{L} = \mathbf{L}^T, \ , \mathbf{L}_{ij} \leq 0, \ \mathbf{L}_{ii} = -\sum_{j=1, j\neq i}^{n} \mathbf{L}_{ij}\}$.
2. Based on requirement you may need to realize $\mathbf{L}$ as sparse. Sparse $\mathbf{L}$ means the nodes are not connected.
3. For $K-$component $\mathbf{L}$ will be of block diagonal form.

4. Generate $\mathbf{Y} \sim \mathcal{N}(\mathbf{0}, \mathbf{L}^{\dagger})$

5. $\mathbf{L}_{ref} = \mathbf{L}$

Now with the generated $\mathbf{Y}$ calculate $\mathbf{S}$ and use our algorithm to obtain the estimate of $\mathbf{L}$ as $\hat{\mathbf{L}}$.

Evaluate, relative error as the metric for performance evaluation, which is defined as:

$$\text{RE} := \frac{\left\| \mathbf{L}_{ref} - \hat{\mathbf{L}} \right\|_F}{\mathbf{L}_{ref}} \tag{16}$$

Where, $\|.\|_F$ is the Frobenius norm.

After finishing this we will redo the above simulations with noisy model. We will elaborate more on this later.

References

[1] D. A. Tarzanagh and G. Michailidis, "Estimation of graphical models through structured norm minimization," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 7692–7739, 2017.

[2] H. E. Egilmez, E. Pavez, and A. Ortega, "Graph learning from data under laplacian and structural constraints," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 6, pp. 825–841, Sept 2017.

[3] E. Pavez, H. E. Egilmez, and A. Ortega, "Learning graphs with monotone topology properties and multiple connected components," *IEEE Transactions on Signal Processing*, vol. 66, no. 9, pp. 2399–2413, May 2018.