Abrar Zahin
azahin@asu.edu

Stochastic Modeling in Biology
APM 541

Structure Learning in Gaussian Graphical Models

Abrar Zahin

December 7, 2022

1 Introduction

Probabilistic graphical models have emerged as a powerful and flexible formalism for expressing and leveraging the relationships among entities in large interacting systems. They have come to find applications in a wide range of domains such as statistical physics and computational biology to natural language processing and computer vision. One important problem associated with graphical models is that of learning the structure of dependencies between the variables described by such a model from data. This is useful as it not only allows a succinct representation of a potentially complex multivariate distribution, but it might in fact reveal fundamental relationships among the underlying variables. This problem of learning the structure associated to a graphical model has a long and prodigious history. In this project, I will first give a brief review of two papers ([CS20] and [MVL20]) on structure learning of Gaussian graphical models (introduced later). Then, I will point out a different toy algorithm motivated from [CS20] which can be used to quickly capture the changes in the underlying models. We first start with the typical structure learning problem setup for (Gaussian) graphical models. In this report, vectors will be denoted as both $\underline{X}$ and $\mathbf{X}$.

2 Problem Setup

Let $G = (V, E)$ (where $V = [p]$) is an undirected graph on the vertex set $[p] = \{1, 2 \dots p\}$ with edge set $E \subseteq \binom{p}{2}$. Each vertex of the graph is associated with the components of zero mean Gaussian Random vector $\underline{X} \in \mathbb{R}^p$, covariance matrix $\Sigma \in \mathbb{R}^{p \times p}$ and Inverse covariance matrix $\Theta \triangleq \Sigma^{-1}$. $\underline{X}$ is said to be Markov w.r.t to the graph G if for any pair of vertices i, j, $\{i, j\} \notin E$ implies X_i and X_j are conditionally independent given $X_{[p]\setminus\{i,j\}}$. $\forall \{i, j\} \notin E$, $\Theta_{ij} = 0$ and thus our goal is to learn the non-zero entries of Θ. In Gaussian graphical model reconstruction task, we are interested in algorithms which can output an accurate estimate $\widehat{G}$ of G, i.e, $\mathbb{P}\left(G \neq \widehat{G}\right) > 1 - \delta$ for a given confidence $\delta > 0$.

3 Information Theoretic Optimal Learning of Gaussian Graphical Models

This paper [MVL20] first points out the information-theoretic lower bound (IT) lower bound for minimum number of samples n^* required for reconstructing a sparse graph, which is given as follows

$$n^* > \max\left\{\frac{\log\binom{p-d}{2}-1}{4\kappa^2}, \frac{2\left(\log\binom{p}{d}-1\right)}{\log\left(1+\frac{d\kappa}{1-\kappa}\right), \frac{d\kappa}{1+(d-1)\kappa}}\right\} \tag{1}$$

where $\kappa \triangleq \min_{(ij\in E)} \frac{|\Theta_{ij}|}{\sqrt{\Theta_{ii}\Theta_{jj}}}$ and denotes the minimum normalized edge strength.

Note that bound in eq. (1) depends only on the parameters of the underlying graph, not on any additional assumptions. Then they proposed two algorithms named *Sparse Least-squares Inverse Covariance Estimator* (SLICE) and *Degree- constrained Inverse Covariance Estimator* (DICE), and their sample complexity scales like $d + \frac{32}{\kappa^2}\log\left(\frac{4p^{d+1}}{\delta}\right)$ and $2d + \frac{192}{\kappa^2}d\log p + \frac{64}{\kappa^2}\log\left(\frac{4d}{\delta}\right)$ respectively. Both DICE and SLICE consist of three steps and a brief overview of the steps are given as follows.

3.1 DICE

3.1.1 Estimating conditional variances:

In this step, DICE obtains an estimate of the conditional variances $\widehat{\Theta_{ii}}$ of each variable $i \in [p]$ conditioned on all the neighbors of i by solving the following optimization problem

$$\frac{1}{\widehat{\Theta_{ii}}} = \min_{\widehat{\beta}\in\mathbb{R}^{p-1}} L_i\left(\widehat{\beta}, \widehat{\Sigma}\right) = \frac{1}{n}\sum_{k=1}^{n}\left(x_i^k + \sum_{j\neq i}\beta_{ij}x_j\right)^2, \text{such that } ||\widehat{\beta}||_0 \leq d \tag{2}$$

3.1.2 Iterative Support Testing:

Fix $i \in V$ and this subroutine will test all obtained candidate neighborhoods for i. Consider a candidate neighborhood $B_1 \subset V\backslash\{i\}$ with $|B_1| = d$. The objective of this subroutine is to obtain the true neighborhood $B_i \subseteq B_1$. This testing proceeds as follows.

1. Fix some adversarial neighborhoods $B_2 \subset V\backslash\{\{i\} \cup B_1\}$ with cardinality d.
2. Let $B_iB_j = B_i \cup B_j$ and compute regression coefficients $\widehat{\beta}_{iB_1B_2} = -\widehat{\Sigma}^{-1}_{B_1B_2,B_1B_2}\widehat{\Sigma}^{-1}_{B_1B_2,i}$ and B_1 is a neighborhood if $\forall B_2$,

$$\max_{j\in B_2} \widehat{\kappa}_{ij} \triangleq |\widehat{\beta_{ij}}|\sqrt{\frac{\widehat{\Theta_{ii}}}{\widehat{\Theta_{ij}}}} < \frac{\kappa}{2} \tag{3}$$

There are two key things to notice in eq. (3)

- Assume that $\widehat{\beta}_{iB_1B_2}$ is correct, then $\forall B_2$ and $\forall j \in B_2$, estimate $\widehat{\kappa}_{ij} \approx \kappa_{ij}$ (where κ_{ij}=0)

- If $\exists j$ such that $j \in B_i \backslash B_1$, then condition in eq. (3) will fail make B_1 fail the criterion.

3.1.3 Eliminate non-edges:

Given that we have a candidate neighborhood B_1 such that $|B_1| = d$ and $B_i \subseteq B_1$, this subroutine will eliminate all extra edges from B_1 by computing the estimated edge strength $\widehat{\kappa}_{ij}\ \forall j \in B_1$ and discarding any $j \in B_1$ as an edge if $\widehat{\kappa}_{ij} < \frac{\kappa}{2}$

3.2 SLICE

Their second proposed algorithm named SLICE offers better computational complexity by trading off in sample complexity which exploits mixed integer programming formulation in Phase 3 from DICE. Key steps of SLICE are as follows

3.2.1 Least Squares with l_0- constraint

$$\widehat{\beta}_i = \min_{\widehat{\beta} \in \mathbb{R}^{p-1}} L_i\left(\widehat{\beta}, \widehat{\Sigma}\right) = \frac{1}{n}\sum_{k=1}^{n}\left(x_i^k + \sum_{j \neq i}\beta_{ij}x_j\right)^2, \quad \text{s.t } ||\widehat{\beta}||_0 \leq d \tag{4}$$

By comparing eq. (2) and eq. (4) we can see that the purpose of SLICE is to estimate regression coefficients rather than estimating the conditional variances as it was in DICE.

3.2.2 Estimate the support

After estimating $\widehat{\beta}_i\ \forall i \in V$, the estimate of edge-set $\widehat{E}$ can be obtained by following thresholding procedure

$$\widehat{E} = \left\{(i,j) \in V \times V : \sqrt{|\widehat{\beta}_{ij} \times \widehat{\beta}_{ji}|} > \frac{\kappa}{2}\right\} \tag{5}$$

3.2.3 Implementation as a mixed integer quadratic program

In order to prevent an exhaustive search over all possible size d neighborhood of each vertex $i \in V$, when d is big enough, this phase of SLICE algorithm formulates the problem as a significantly faster mixed integer quadratic program. In the following formulation L and U denote upper and lower bounds on the regression. variables.

$$\begin{aligned}
\min_{\widehat{\beta} \in \mathbb{R}^{p-1}} \quad & \beta_i^T \widehat{\Sigma}_{\bar{i}\bar{i}} + 2\widehat{\Sigma}_{\bar{i}i} + \widehat{\Sigma}_{ii} && \text{(6a)}\\
\text{such that} \quad & s_{ij}L \leq \beta_{ij} < s_{ij}U, \ \ \forall j \neq i && \text{(6b)}\\
& \textstyle\sum_{j \neq i} s_{ij} = d && \text{(6c)}\\
& s_{ij} \in \{0,1\} \ \ \forall j \neq i && \text{(6d)}
\end{aligned}$$

3.3 Condition Number dependence

One important aspect of this paper is that their proposed algorithms are not sensitive to condition number parameter absent in IT lower bound (eq. (1)). They illustrated this fact by sketching a sequence of matrices which has a growing condition number whereas sample complexity of DICE and SLICE are not scaled accordingly.

4 Learning Gaussian Graphical Models via Multiplicative Weights

Consider the preceding setup for GGM reconstruction, we will now provide a brief overview of the online algorithm proposed in [CS20]. Let X_i and $\underline{X_{\bar{i}}}$ denote i^{th} coordinate and all other coordinates except i of $\underline{X}$ respectively.We can express $\mathbb{E}\left[X_i|\underline{X_{\bar{i}}}\right]$ by a linear combination of measurements from other $p-1$ nodes as follows,

$$\mathbb{E}\left[X_i|\underline{X_{\bar{i}}}\right] = \sum_{j\neq i}\left(-\frac{\Theta_{ij}}{\Theta_{ii}}\right)X_j = \underline{w_i}\cdot\underline{X_{\bar{i}}}$$

Where $\underline{w_i} \in \mathbb{R}^{p-1}$ represents corresponding weights of samples from other $p-1$ nodes estimate X_i. In what follows is a brief listing of the core parts the online algorithm. For a detailed description of different parameters see the paper.

Algorithm 1 Learning weight vector for a node i

1: Input: N samples of X, $\underline{v}^{(0)}$ as $\left(\frac{1}{p},\frac{1}{p},\ldots,\frac{1}{p}\right)\in\mathbb{R}^n$, learning rate, β distribution vector
2: $\underline{\rho}$, $\lambda_i = \sum_{i\neq j}\left|\frac{\theta_{ij}}{\theta_{ii}}\right|$ and $\max_i \lambda_i \leq \lambda$.
3: Output: A "good" approximation $\underline{v}^*$ of $\underline{w_i}$.
4: **for** $n = 1$ to N **do**
5: $\quad\underline{\rho}^{(n)} = \frac{\underline{v}^{(n-1)}}{||\underline{v}^{(n)}||_1}$ // $\underline{v}^{(n-1)}, \underline{\rho}^{(n)} \in \mathbb{R}^n$
6: $\quad\underline{l}^{(n)} = (1/2)(\underline{1} + \underbrace{(\lambda\underline{\rho}^{(n)}\cdot\underline{x}^{(n)} - y^{(n)})}_{\text{prediction error}}\underline{x}^{(n)})$
7: $\quad\forall i\in[p]$, $\underline{v}_i^{(n)} = \underline{v}_i^{(n-1)}\underline{\beta}^{l_i^{(n)}}$
8: **end for**
9: Get N candidate weight vectors.
10: Further use M samples to see which candidate vector exhibits smallest empirical risk.
11: Return $\underline{v}_i^* = \lambda\underline{\rho}^n$, where n denotes the candidate weight vector with smallest empirical risk.

Although the algorithm presented in [CS20] is the adoption of SPARSITRON algorithm proposed in [KM17] for Gaussian case, [CS20] tackle some challenges due to the continuous and unbounded nature of the problem which prohibits the use of several parts of the analysis in [KM17]. Their sample complexity scales like $\mathcal{O}\left(\left(\frac{\lambda}{\kappa}\right)^4\log^3\frac{p}{\delta}\right)$.

References

[KM17] Adam Klivans and Raghu Meka. "Learning graphical models using multiplicative weights". In: *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE. 2017, pp. 343–354.

[CS20] Anamay Chaturvedi and Jonathan Scarlett. "Learning Gaussian Graphical Models via Multiplicative Weights". In: *arXiv preprint arXiv:2002.08663* (2020).

[MVL20] Sidhant Misra, Marc Vuffray, and Andrey Y Lokhov. "Information theoretic optimal learning of gaussian graphical models". In: *Conference on Learning Theory*. PMLR. 2020, pp. 2888–2909.