Data-driven graph learning: optimization problem, cost function and coding information

Venkat Roy

November 30, 2020

Data and assumptions

1. Simulated hitpatterns: $X : \mathbb{R}^{127\times 989875}$. (Monte Carlo dataset)
2. Data matrix: $B = XX^T : \mathbb{S}^{127\times 127}$.
3. Graph topology : $L = D - W$.
4. Graph learning: estimation of $L \in \mathbb{S}_{+}^{127}$ using B assuming the signal is smooth over graph.

Set of graph Laplacians based on graph connectivity

1. Set of general graph Laplacians:
   $\mathcal{L}_g = \{\mathsf{L} | \mathsf{L} \succeq 0, [\mathsf{L}]_{ij} \leq 0, \text{ for } i \neq j\}$.

2. Set of combinatorial graph Laplacians:
   $\mathcal{L}_c = \{\mathsf{L} | \mathsf{L} \succeq 0, [\mathsf{L}]_{ij} \leq 0, \text{ for } i \neq j, \mathsf{L}\mathsf{1} = 0\}$. (no isolated node)

3. Laplacian to weighted adjacency : $\mathsf{W} = -(\mathsf{L} - \mathrm{diag}(\mathrm{diag}(\mathsf{L})))$
4. Unweighted adjacency (binary matrix) : $[\mathsf{A}]_{ij} = 1$ if $[\mathsf{W}]_{ij} > \epsilon$ else $[\mathsf{A}]_{ij} = 0$, where ϵ is the edge selection threshold.
5. Degree matrix: D (diagonal matrix with the main diagonal with the sum of the rows of A).

Main graph learning cost

1. Optimization problem:

$$\min_{\mathsf{L}\in\mathcal{L}_g} \underbrace{\mathrm{tr}(\mathsf{X}^T\mathsf{L}\mathsf{X})}_{\mathsf{I}} - \underbrace{\log|\mathsf{L}|}_{\mathsf{II}} + \underbrace{\alpha\|\mathsf{L}\|_{1,\mathrm{off}}}_{\mathsf{III}}$$

2. I : Smoothness prior
3. II : Constraint on the sum of the eigenvalues of L as $\log|\mathsf{L}| = \sum_{i=1}^{N} \log\lambda_i$ (computational stability) ($|\mathsf{L}|$ is the determinant of L.)
4. III : Minimizing the off-diagonal elements (capturing local interactions)

Simplification of the graph learning cost

1. Cost function:
   $f(\mathsf{L}) = \mathrm{tr}(\mathsf{LB}) - \log|\mathsf{L}| + \alpha\|\mathsf{L}\|_{1,\mathrm{off}}; \quad \mathsf{B} = \mathsf{XX}^T.$
2. $f(\mathsf{L}) = \mathrm{tr}(\mathsf{LB}) - \log|\mathsf{L}| + \mathrm{tr}(\mathsf{LZ}); \quad \mathsf{Z} = \alpha(\mathsf{I} - \mathsf{11}^T).$
3. $f(\mathsf{L}) = \mathrm{tr}(\mathsf{LG}) - \log|\mathsf{L}|; \quad \mathsf{G} = \mathsf{B} + \mathsf{Z}.$

Combinatorial graph learning problem

1. Main problem:

$$\min_{\mathsf{L}\in\mathcal{L}_g} \quad \mathrm{tr}(\mathsf{L}\mathsf{G}) - \log|\mathsf{L}| \tag{1}$$

2. Combinatorial optimization problem:

$$\min_{\mathsf{L}\in\mathcal{L}_c} \quad \mathrm{tr}(\mathsf{L}(\mathsf{G}+\mathsf{R})) - \log|(\mathsf{L}+\mathsf{R})| \tag{2}$$

3. Here, R is a regularization matrix because in (2), L is singular due to the connectivity constraint $\mathsf{L}\mathsf{1} = 0$. Problem (1) and (2) are equivalent. Proof : next slide

4. Both problems (1) and (2) are convex optimization problems. [*Egilmez, Pavez, Ortega*, 2017]

Proof of the equivalence of the cost functions

1. Main problem (general set of Laplacians: $\mathcal{L}_g$) : $\min_{\mathsf{L}\in\mathcal{L}_g} \operatorname{tr}(\mathsf{LG}) - \log|\mathsf{L}|$
2. Combinatorial graph learning problem:
$\min_{\mathsf{L}\in\mathcal{L}_g} \operatorname{tr}(\mathsf{L}(\mathsf{G}+\mathsf{R})) - \log|(\mathsf{L}+\mathsf{R})|$ s.t. $\mathsf{L1} = 0$.
3. Let us assume $\mathsf{R} = \frac{1}{N}11^T$.
Then $\operatorname{tr}(\mathsf{L}(\mathsf{G}+\mathsf{R})) = \operatorname{tr}(\mathsf{LG} + \frac{1}{N}\operatorname{tr}(\mathsf{L}11^T))$.
Using $\mathsf{L1} = 0$, we have $\operatorname{tr}(\mathsf{L}(\mathsf{G}+\mathsf{R})) = \operatorname{tr}(\mathsf{LG})$.
4. For combinatorial Laplacians the eigenvector corresponding to the first eigenvalue, i.e., 0 is given by $\mathsf{v}_1 = \frac{1}{\sqrt{N}}1$ (by definition). So, we have $\lambda_1(\mathsf{L}) = 0$. In that case, in eigen decomposition form we can write
$$\frac{1}{N}11^T + \mathsf{L} = (\underbrace{\lambda_1(\mathsf{L})}_{0} + 1)\mathsf{v}_1\mathsf{v}_1^T + \sum_{i=2}^{N} \lambda_i(\mathsf{L})\mathsf{v}_i\mathsf{v}_i^T.$$
5. As the determinant is the product of the eigen values
$$\log|\frac{1}{N}11^T + \mathsf{L}| = \log(1.\prod_{i=2}^{N} \lambda_i(\mathsf{L})) = \log|\mathsf{L}|.$$
6. So, we can claim that (1) and (2) are equivalent.

Description of the code : used toolboxes

- Graph Laplacian Learning (GLL) Package v2.2:
  https://github.com/STAC-USC/Graph_Learning
  Use: solving the block coordinate descent (BCD) algorithm
- GSP box : https://epfl-lts2.github.io/gspbox-html/ (optional)
  Available in Python as PyGSP :
  https://github.com/epfl-lts2/pygsp
- Disciplined convex programming (CVX) :
  http://cvxr.com/cvx/
  General purpose convex optimization problem solver but slower in speed than block coordinate descent (with increasing number of nodes) (not included in the repository).

Description of the code : files in the DIDACTS repository

- Graph Laplacian Learning (GLL) Package v2.2:
  Toolbox: Graph_Learning_master.zip
- Main graph learning code: Graph_learning_main.m
  Input: Data: XX^T, Initialization (adjacency matrix) of the BCD algorithm, sensor (PMT) coordinates.
  Output: Learned graph laplacian, adjacency and degree matrices
- Combinatorial graph learning function using BCD: estimate_cgl.m
- CVX implementation of graph learning: lines 36 to 45 of Graph_learning_main.m (cost function can be changed, e.g. extra priors can be added)

Input and output files

- **Input**: Data: XX^T (data_cov.csv), PMT coordinates: top_PMT_coordinates.mat.
- **Output**: Estimated adjacency from Laplacian (A_estimated.csv)