M1 Internship defense 2024
Screening for Graph Learning

By

Amrithya Balaji
Supervisor: Benjamin Girault

Machine Learning and Data Mining
Université Jean Monnet

Screening for Graph Learning

Abstract

Graph learning is a subset of machine learning that focuses on the analysis and interpretation of data presented in graph form. Data samples are assigned to vertices on a graph, and edge weights indicate similarities between them. To learn a sparse graph, we use a coordinate minimization algorithm that updates an edge weight with each iteration and implement a variable screening method to speed up computations.

1. Context of the internship

The University of Saint-Etienne, the National Research Centre (CNRS), and the Institut d'Optique Graduate School collectively operate the joint research unit, Laboratoire Hubert Curien (UMR 5516) located in Saint-Etienne since 1995. Malice (MAchine Learning with Integration of SurfaCe Engineering knowledge: Theory and Algorithm) is a collaborative project team under Inria (French national institute for research in digital science and technology) and CNRS and the goal is to address scientific and technological challenges related to surface engineering.

The internship is on the topic "Screening for Graph Learning" in Inria MALICE project-team, Laboratoire Hubert Curien under the supervision of Prof Benjamin Girault. The main goal of the the internship is to implement screening to speed up computations in a set of graph learning algorithms.

2. Introduction

Many real-world systems and datasets can be naturally represented as graphs, which connect entities (nodes) via relationships (edges). When dealing with multivariate data, it's natural to want to model the interdependence of the system's variables. Graph learning enables us model complex relationships and dependencies between entities that may not be visible in other simpler data structures such as array or matrix. A classical approach to obtain a graph is by learning from the data using inverse covariance estimation (Girault et al., 2023). Our motive is to introduce a maximum likelihood method for calculating edge and vertex weights.

Graphs are versatile data representation forms that can describe the geometric structure of data domains. We deal with graph learning algorithm specifically coordinate minimization and graph signal processing to learn a spares graph with all the important relationships or connections between data points. We use the inverse covariance estimation approach to identify relationships between variables. This method often involves techniques like the graphical lasso, which tells sparsity on the inverse covariance matrix, making it easier to interpret the graph structure.

We have maximum likelihood optimization using coordinate minimization to find the parameter values that maximize the likelihood function in an efficient way. This can provide simpler updates than gradient decent. Lasso or ℓ_1 is a regularization technique that penalizes the coefficients, often leading some coefficients to be exactly zero (Herzet et al., 2022). Similarly, screening procedures can identify variables that can stay at zero, and those variables need not to be considered to avoid further computation.With graph learning, Screening methods are also implement which are effective tools for resolving optimization problems where it speed ups the computations. The screening method involves the comparison between the values of a relaxed primal cost and the dual cost to derive the guarantees for the optimum.

This report provides a brief background on graphs and graph learning algorithm, specifically Coordinate Minimization and our region-free screening method.

3. Background

3.1. Graphs

Graphs are useful for modeling high-dimensional data and have been used to solve data processing problems in various fields. Graph nodes represent variables or features, while edges represent the relationships between them. A graph $\mathcal{G}$ can be represented as a set of vertices $\mathcal{V}$, edge set $\mathcal{E}$, and a weight matrix $\mathbf{W}$, as shown below. The edge weight function $w : \mathcal{E} \to \mathbb{R}$ is defined such that for each edge $e \in \mathcal{E}$, w_e denotes the weight assigned to e. Edges can have different weights, and the weight matrix will store these values.

$$\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$$

$$w : \mathcal{E} \to \mathbb{R}, \quad e \mapsto w_e$$

Let $\mathcal{V} = \{1, \cdots, m\}$, and let the edge set $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ have individual edges denoted by $e = (i, j) \in \mathcal{E}$, with

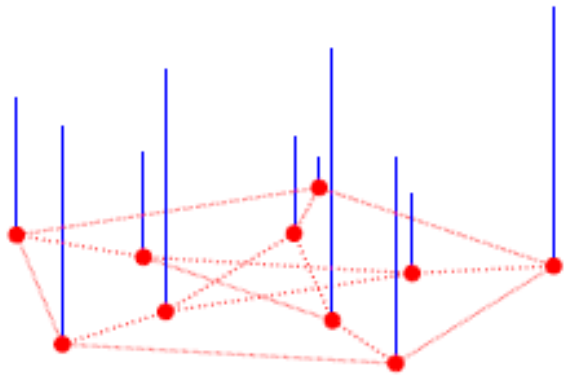


Figure 1. Representation of Graph Signal (Shuman et al., 2013), where the height of each blue bar represents the signal value of the vertex.

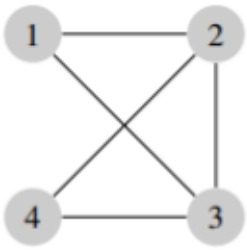
corresponding edge weight $w_e = w_{ij}$. The weight matrix $\mathbf{W} = (w_{ij})$ is symmetric and non-negative, and $w_{ij} > 0$ if and only if $(i, j) \in \mathcal{E}$ and $(j, i) \in \mathcal{E}$.An undirected graph is one in which the edges have no specific direction assigned to them. Since we consider undirected graphs, we do not repeat edges when listing them; that is, if $(i, j) \in \mathcal{E}$, we do not include (j, i) in the edge set.

For example, the weight of an edge in a graph can indicate the degree of similarity between two vertices. Connectivities and edge weights are determined by the physics of the problem or derived from data. Edge weight may be inversely proportional to the physical distance between nodes in the network. Graphs represent a finite collection of samples, with one sample at each vertex. We collectively refer to these samples as a graph signal represented in figure 1.

Suppose that the value of the signal $g : \mathcal{V} \to \mathbb{R}$ on a vertex i is $g(i) \in \mathbb{R}$. Given a vertex in the graph, this will give a real number as output.The graph laplacian is an important component for graph signal processing because the eigenvectors and eigenvalues of this matrix serve as the foundation for frequency domain analysis of graph signals. The graph Laplacian $\mathbf{L}$ is defined as follows.

$$\mathbf{L} := \mathbf{D} - \mathbf{W}$$

where degree matrix $\mathbf{D}$ is a diagonal matrix and captures the total weight (or summed edge weights) incident to each vertex along the diagonal elements of the matrix. In a graph with n vertices, the degree matrix $\mathbf{D}$ is an $n \times n$ matrix with the element $\mathbf{D}_{ii}$ (the element on the diagonal at row i and column i) representing the degree of vertex i. Every



$$\mathbf{D} = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix} \quad \mathbf{W} = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

$$\mathbf{L} = \begin{bmatrix} 2 & -1 & -1 & 0 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 3 & -1 \\ 0 & -1 & -1 & 2 \end{bmatrix}$$

Figure 2. Calculating the Laplacian of a graph

off-diagonal element is zero :

$$\mathbf{D}_{ii} = \deg(v_i)$$
$$\mathbf{D}_{ij} = 0 \quad \text{for} \quad i \neq j$$

where $\deg(v_i)$ is the degree of vertex i.

The Laplacian matrix (Fig 2) helps represent the structure of the graph, capturing the relationships and connections between nodes. The eigenvalues and eigenvectors of the Laplacian matrix are used to understand the properties of the graph, such as connectivity and graph partitions.

Let the unweighted incidence matrix of a graph with n edges and m vertices be $B = [\mathbf{b}_1 \cdots \mathbf{b}_n]$ and $W = diag(\mathbf{w}_1 \cdots \mathbf{w}_n)$ such that for any edge $e = (i, j) \in \mathcal{E}$, where $i < j$, we have $B_{i,e} = 1$ and $B_{j,e} = -1$, and B is zero else where, the Laplacian can also be written as

$$L = BWB^T. \tag{1}$$

A *sparse graph* is one in which the number of edges is significantly smaller compared to the maximum possible number of edges, typically in proportion to the number of vertices.Sparse graphs often have relatively few edges connecting the vertices.

On the other hand, a *well-connected graph* has a higher edge density, meaning it contains a larger proportion of possible

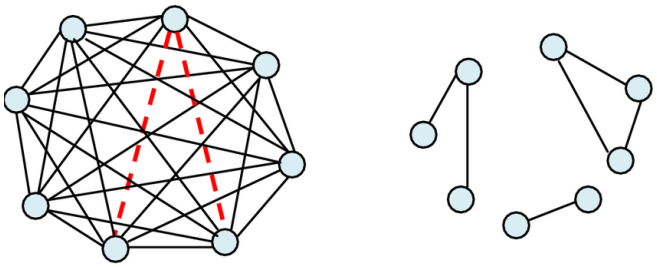


Figure 3. A well connected graph (Left) and a sparse graph(right) (Barham et al., 2019)

edges. In such graphs, most vertices are connected to many other vertices, resulting in a dense network of edges.There is a path between any pair of vertices.

The main goal of this paper is to model the statistical relations between vertices while remaining sparse.The main difference between sparse and well-connected graphs (Fig 3) lies in their edge density.

There are several ways to measure a graph's connectivity (Pavez & Ortega, 2020). Well-connected graphs can be defined as having a low total resistance or high algebraic connectivity. These quantities describe the rate of information propagation and communication within a network. A third approach uses the matrix-tree theorem, which states that the number of spanning trees in a graph equals the minor (determinant of a sub-matrix) of the Laplacian. To improve connectedness, consider adding edges or increasing edge weights. To determine a graph connectedness we define:

- A path from node i to node j is a sequence of edges $P_{ij} = \{(i_1, i_2), (i_2, i_3) \cdots, (i_{t-1}, i_t)\} \subset \mathcal{E}$ so that $i = i_1$, and $j = i_t$, the graph is said to be well connected.
- A graph is connected if $\forall (i, j) \in \mathcal{V} \times \mathcal{V}$.
- If and only if $\lambda_2 > 0$ where λ_2 is the second smallest eigenvalue of the Laplacian matrix.

3.2. Coordinate Minimization

Coordinate minimization is an optimization algorithm used in machine learning. Suppose we have a function $f(\mathbf{x})$ where $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ is a vector of n parameters. Each coordinate direction corresponds to one of the parameters x_i in the vector $\mathbf{x}$. When the algorithm selects a coordinate direction x_i, it optimizes the objective function $f(\mathbf{x})$ with respect to x_i, while keeping all other coordinates x_j (where $j \neq i$) fixed. It is an iterative method for optimizing a function by successively minimizing along single coordinate directions. This is simpler than optimizing all parameters simultaneously because updating a single parameter x_i while keeping others fixed has less computations.

We assume our data is i.i.d. multivariate Gaussian. Our goal is to model data using a graph defined by its Laplacian and signal inner product matrix, $Q = \text{diag}(q)$ where q is the vertex importance vector.

Theorem 1. (Girault et al., 2023) With the power spectrum $\gamma(\lambda) = (1 + \lambda)^{-1}$ where γ is the the variance of spectral component of the signal, the covariance matrix of a graph signal can be written using the (L, Q)-GFT as:

$$\Sigma = [Q + L]^{-1}. \tag{2}$$

Using maximum likelihood and combining the Gaussian assumption (3) with Theorem. 1 (2), we get from thesection appendix A.1 (11) the following convex cost function to minimize:

$$F(L, Q) = -\log\det(Q + L) + \text{tr}((Q + L)S),$$

where S is the empirical covariance matrix from (10).

The trace term in the cost function (11) can be written as a sum $\text{tr}(LS) + \text{tr}(QS)$. Further more, the terms can be written as:

$$\text{tr}(LS) \text{ can be written as } \text{tr}(B^T w_e B S) = \sum_{i=1}^{n} (w_e B^T S B)$$

where $(B^T w_e B S) =$ edge cost and $w_e =$ edge weight

$$\text{tr}(QS) \text{ can be written as } \text{tr}(q_i \delta_i^t \delta_i S) = \sum_{i=1}^{m} (q_i \delta_i^T S \delta_i)$$

where $(\delta_i^T S \delta_i) =$ vertex cost and $q_i =$ vertex importance

and with δ_i as a kronecker vector

Edge cost

We define h_e as the cost of including edge weight $e = (i, j)$ in a graph. Edges with lower costs are typically given higher weights.

$$h_e = b^T S b_e$$

Effective Resistance

The intution behind the name effective resistance, is with an electrical network on n nodes in which each edge e corresponds to a link of conductance w_e (i.e., a resistor of resistance $\frac{1}{w_e}$). Then, the effective resistance r_e across an edge e is the potential difference induced across it when a unit current is injected at one end and extracted at the other.

$$r_e^t = b_e^T (Q^{(t)} + L^{(t)})^{-1} b_e$$

Vertex cost

We define p_i as the cost of including a vertex. It is similar to the edge cost h_e

$$p_i = \delta_i^T S \delta_i$$

where δ_i is a Kronecker vector,

$$\delta_i(j) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

Vertex Effective Resistance

We get vertex effective resistance, denoted by u_i, from

$$u_i = (Q^{(t)} + L^{(t)})^{-1}_{ii} = \Sigma_{ii}$$

To solve the following graph learning problem,we have [(11),(1)]:

$$\min_{\substack{w\geq 0 \\ q>0}} F\left(B\,W\,B^T,\, \text{diag}(q)\right). \tag{3}$$

where $W = \text{diag}(w_1, \ldots, w_M)$ is the diagonal matrix collecting edge weights.

Coordinate minimization iterates through all edges and vertices, updating their weights. Updates can be calculated with the introduced hyperparameter q_{min} where $q_{min} > 0$:

$$\delta_e^{(t)} = \max\left(-w_e^{(t)}, \frac{1}{h_e} - \frac{1}{r_e}\right),$$
$$\delta_i^{(t)} = \max\left(q_{\text{min}} - q_i^{(t)}, \frac{1}{p_i} - \frac{1}{u_i}\right).$$

Any update to w_e (resp. q_i) will only modify the first (resp. second) trace in this sum.

4. Region-Free Screening

We solve our graph learning's optimization problem using both the primal and dual problem.This section introduces the dual problem, which is derived using the fenchel conjugate.The screening method involves comparing relaxed primary and dual costs to determine the optimal solution.

4.1. Primal and Dual Problem

Primal Problem

We have our cost function (11) in the form of

$$x \mapsto f(Ax) + g(x)$$

With the graph learning algorithm, our objective function, integrating non-negativity constraints for graph edge weights ans $q_i \geq q_{\text{min}}$ is:

$$F : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$$

$$(Q, L) \mapsto -\log\det(Q + L) + \text{tr}[(Q + L)\Sigma]$$

With

$$\mathbf{Q}(\mathbf{q}) = \text{diag}(\mathbf{q}) = \sum_{i=1}^{n} \delta_i \delta_i^T q_i$$

$\mathbf{L}(\mathbf{w}) = \mathbf{B}\,\text{diag}(\mathbf{w})$; $\mathbf{B}^T = \sum_{e=1}^{m} \mathbf{b}_e \mathbf{b}_e^T w_e$; $\mathbf{B} = [\mathbf{b}_1 \ldots \mathbf{b}_m]$, reparametrization term $r = q - q_{\min}1$ and our primal problem $(\mathbf{r}^*, \mathbf{w}^*)$, we have:

$$\arg\min_{\mathbf{r},\mathbf{w}\geq 0} -\log\det(\sum_{i=1}^{n} r_i \delta_i \delta_i^T + \sum_{e=1}^{m} w_e \mathbf{b}_e \mathbf{b}_e^T + q_{min}\mathbf{I}) + \mathbf{p}^T\mathbf{r} + \mathbf{h}^T\mathbf{w}$$

Dual Problem

Since

$$f(P) = -\log\det(P). \tag{4}$$

(4) is assumed to be convex, closed, and proper. The dual problem of the primal can be written. We compute the Fenchel conjugate f^* of (4).

$$f^*(U) = \sup_P \text{tr}(P^\top U) + \log\det(P)$$

Since $\log\det(P)$ requires P to be positive definite, $P \succ 0$:

$$f^*(U) = \sup_P(\text{tr}(P^T U) + \log\det(P)) \tag{5}$$

sup goes to infinity if U has any non-negative eigenvalue. Hence, U must be negative semi-definite.

Maximization Condition:

With Maximization Condition $\frac{\partial}{\partial P}(\text{tr}(P^T U) + \log\det(P)) = 0$:

$$U + P^{-1} = 0 \implies P = -U^{-1} \tag{6}$$

Fenchel Conjugate $f^*(U)$:

Substituting (6) in (5) (Herzet et al., 2022):

$$f^*(U) = (\langle U, -U^{-1}\rangle + \log\det(-U^{-1})) + \mathbb{I}(U \in \mathbb{S}_n^-)$$

Using the identity $\text{tr}(AB) = \text{tr}(BA)$:

$$\langle U, -U^{-1}\rangle = -\text{tr}(I) = -n$$

$$\log\det(-U^{-1}) = -\log\det(U) - n\log(-1)$$

Since $U \in \mathbb{S}_n^-$:

$$f^*(U) = -n - \log\det(U) + \mathbb{I}(U \in \mathbb{S}_n^-)$$

Dual Problem Formulation

From the computed Fenchel conjugate, we can write the dual problem using the conjugate function and the constraints on U:

Dual Cost:

$$d(U) = -q_{\min}\mathrm{tr}(U) + n + \log\det(U) + \mathbb{I}(U \in \mathbb{S}_n^-)$$

Dual Constraints:

Derived from the primal constraints, we have:

$$\delta_i^T U \delta_i \leq p_i, \quad \forall i \in \{1, \ldots, n\}$$

$$b_e^T U b_e = h_e, \quad \forall e \in \{1, \ldots, m\}$$

So, the dual problem can be written as

$$d(U) := -q_{\min}\,\mathrm{tr}(U) + n + \log\det(U) \tag{7}$$

4.2. Screening Method

Primal cost

Update the relaxed primal cost with respect to the current cost and the primal cost update where l is the variable. Primal cost update (PCU) can be calculated where l is the variable to be updated :

$$\text{PCU} = \begin{cases} -\log(1+\delta_e r_e) - \delta_e h_e & \text{when l = edge,} \\ -\log(1+\delta_i u_i) - \delta_i p_i & \text{when l = vertex.} \end{cases}$$

$$\text{relaxed primal cost} = \text{current cost} + \text{PCU} \tag{8}$$

Dual Admissible Solution

From a primal admissible solution (r, w), we construct a dual admissible solution.

$$\bar{U} = \min\left\{(\frac{p_i}{u_i})_i, (\frac{h_e}{r_e})_e\right\} \times \Sigma \tag{9}$$

The algorithm for the screening method is given in Algorithm 1. For any feasible primal solution (r_l, w_l) and any feasible dual solution $\bar{U}$, the dual cost $d(\bar{U})$ is always less than or equal to the primal cost $p_l(r_l, w_l)$.

The comparison $d(\bar{U}) > p_l(r_l, w_l)$ is to determine whether a variable should be set to zero or to update the variable. If the dual cost exceeds the primal cost, setting it to zero might improve our objective.

5. Results

An exponential variogram shows the spatial variability (or semi-variance) of a random field against the distance between sample points. The exponential variogram $\gamma(h)$ is defined as:

$$\gamma(h) = C_0 + C\left(1 - e^{-\frac{h}{a}}\right)$$

Algorithm 1 Region-Free Screening

Data: Current cost
Result: : Whether variable l should be set to 0 or to be updated.

1: Compute the relaxed primal cost $p_l(r_l, w_l)$ according to (8);
2: Compute the admissible dual $\bar{U}$ according to (9);
3: Compute the dual cost $d(\bar{U})$ according to (7);
4: **return** if dual cost $d(\bar{U}) >$ primal cost $p_l(r_l, w_l)$

where h is the distance between two points, C_0 is the nugget, which represents the y-intercept, C is the sill, which is the total variance minus the nugget. a is the range, which represents the distance over which spatial correlation diminishes.The parameter range of a variogram defines the distance beyond which the spatial correlation becomes negligible.The range can determine the sparsity of the graph.For example, vertices i and j are more likely to be connected if their distance d_{ij} is within a range where $\gamma(d_{ij})$ (semivariance from the Exponential Variogram) is higher.

This table 1 displays the outcomes of coordinate minimization algorithms with varying constraints. The algorithms are evaluated using two primary metrics: average runtime (in seconds) and sparsity. The experiments are carried out over three different ranges (1, 0.1, and 0.01) and under two conditions for each range: with and without Q. Every time, five sets of vertices are generated. The table 1 below displays the average run time.The best graph produced by our algorithm is a sparse graph in figure 5, whereas the initial graph in figure 4 is a well-connected graph. The figure 4 is the resultant for Coordinate Minimization with Q and with Screening. The result table 1 shows that the inclusion of Q generally improves the sparsity of the solutions, especially as the range narrows.

6. Conclusion

In this paper, we explored the implementation of a graph learning algorithm that uses coordinate minimization and screening methods to increase computational efficiency. Our results showed that incorporating screening significantly reduced average runtime.. This improvement was visible when the average run times of methods with and without screening were compared across different ranges.

The efficacy of the screening method in speeding up computations while maintaining the accuracy of the graph learning process.Overall, the combination of coordinate minimization and variable screening is an effective tool for graph learning, providing both computational efficiency and robust performance. Future work may involve more complex and real time datasets such as Molene Dataset (Girault, 2017).

Constraints		Without Q		With Q	
Range	**Screening**	**Avg run time in sec**	**Sparsity**	**Avg run time in sec**	**Sparsity**
1	✓	10.3	90.6%	13.1	90.9%
1	✗	11.1		13.7	
0.1	✓	117.6	9.8%	45.0	85.3%
0.1	✗	104.4		39.0	
0.01	✓	120.0	0%	3.4	79.5%
0.01	✗	108.0		2.9	

Table 1. Comparison of Average Run Time in Seconds with and without Q

7. Acknowledgements

I would like to express our sincere gratitude to my supervisor and professor Benjamin Girault for their invaluable guidance and supervision throughout this internship. Their expertise and insights have played a pivotal role in shaping the direction and success of our work.

I would also like to extend my heartfelt thanks to Naïma Chalais Traoré (Laboratoire Hubert Curien - Cellule d'appui à la Recherche) and other members of MLDM (Machine Learning and Data Mining) program for their unwavering support throughout my internship. Furthermore, I would like to acknowledge our MLDM (Machine Learning and Data Mining) classmates who generously shared their knowledge and provided assistance when needed.

I am grateful to all individuals who have contributed to this endeavor, and their support has been invaluable.

References

Barham, R., Sharieh, A., and Sleit, A. Multi-moth flame optimization for solving the link prediction problem in complex networks. *Evolutionary Intelligence*, 12, 12 2019. doi: 10.1007/s12065-019-00257-y.

Girault, B. Molene-dataset. https://github.com/bgirault-usc/Molene-Dataset, 2017. Repository.

Girault, B., Pavez, E., and Ortega, A. Joint graph and vertex importance learning, 2023.

Herzet, C., Elvira, C., and Dang, H.-P. Region-free Safe Screening Tests for l1-penalized Convex Problems. In *Eusipco 2022 - 30th European Signal Processing Conference*, pp. 1–5, Belgrade, Serbia, August 2022. URL https://centralesupelec.hal.science/hal-03806099.

Pavez, E. and Ortega, A. An efficient algorithm for graph laplacian optimization based on effective resistances, 2020.

Shuman, D. I., Narang, S. K., Frossard, P., Ortega, A., and Vandergheynst, P. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine*, 30(3):83–98, May 2013. ISSN 1053-5888. doi: 10.1109/msp.2012.2235192. URL http://dx.doi.org/10.1109/MSP.2012.2235192.

A. Appendix

A.1. Maximum Likelihood Estimation

Setup

Gaussian Distribution: The graph signal x is assumed to be Gaussian distributed. The Gaussian distribution is defined by its mean (assumed to be zero for simplicity) and its covariance matrix Σ.

Covariance Matrix: From Theorem 1, the covariance matrix Σ is given by:

$$\Sigma = (Q + L)^{-1}$$

This can be generalized in terms of precision matrix Θ,

$$\Sigma = \Theta^{-1}$$

Given that Q is Hermitian positive definite and L is Hermitian positive semi-definite, then $Q + L$ is definite, hence invertible, and Σ is well-defined and a valid covariance matrix.

Multivariate Gaussian Distribution

For a random vector $x \in \mathbb{R}^N$ following a multivariate Gaussian distribution with mean vector μ and covariance matrix Σ, the probability density function (PDF) is given by:

$$p(x) = \frac{1}{(2\pi)^{N/2} \det(\Sigma)^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right)$$

When the mean vector $\mu = 0$, this simplifies to:

$$p(x) = \frac{1}{(2\pi)^{N/2} \det(\Sigma)^{1/2}} \exp\left(-\frac{1}{2}x^T \Sigma^{-1} x\right)$$

Likelihood Function

Given n independent observations $\{x_i\}_{i=1}^n$ from this distribution, the likelihood function $\mathcal{L}$ is :

$$\mathcal{L}(L, Q \mid \{x_i\}) = \prod_{i=1}^{n} p(x_i)$$

Substituting the PDF of the multivariate Gaussian distribution, we get:

$$\mathcal{L}(L, Q \mid \{x_i\}) = \prod_{i=1}^{n} \frac{1}{(2\pi)^{N/2} \det(\Sigma)^{1/2}} \exp\left(-\frac{1}{2}x_i^T \Sigma^{-1} x_i\right)$$

$$\mathcal{L}(L, Q \mid \{x_i\}) = \left(\frac{1}{(2\pi)^{N/2} \det(\Sigma)^{1/2}}\right)^n \exp\left(-\frac{1}{2}\sum_{i=1}^{n} x_i^T \Sigma^{-1} x_i\right)$$

Substituting $\Sigma = (Q + L)^{-1}$ into the likelihood function, we get:

$$\mathcal{L}(L, Q \mid \{x_i\}) = \left(\frac{\det((Q+L)^{-1})^{1/2}}{(2\pi)^{N/2}}\right)^n \exp\left(-\frac{1}{2}\sum_{i=1}^{n} x_i^T (Q+L) x_i\right)$$

Using the property of determinants, $\det(A^{-1}) = \frac{1}{\det(A)}$, we have:

$$\mathcal{L}(L, Q \mid \{x_i\}) = \left(\frac{\det(Q+L)^{1/2}}{(2\pi)^{N/2}}\right)^n \exp\left(-\frac{1}{2}\sum_{i=1}^{n} x_i^T (Q+L) x_i\right)$$

$$\mathcal{L}(L, Q \mid \{x_i\}) = \frac{\det(Q+L)^{n/2}}{(2\pi)^{nN/2}} \exp\left(-\frac{1}{2}\sum_{i=1}^{n} x_i^T (Q+L) x_i\right)$$

Log-Likelihood

The logarithm of the likelihood function:

$$\log \mathcal{L}(L, Q \mid \{x_i\}) = \log\left(\frac{\det(Q+L)^{n/2}}{(2\pi)^{nN/2}} \exp\left(-\frac{1}{2}\sum_{i=1}^{n} x_i^T (Q+L) x_i\right)\right)$$

Simplifying the right term using the properties $\log(a^b) = b\log(a)$ and $\log(\exp(x)) = x$:

$$\log\left(\frac{\det(Q+L)^{n/2}}{(2\pi)^{nN/2}}\right) = -\frac{nN}{2}\log(2\pi) + \frac{n}{2}\log\det(Q+L)$$

$$\log\left(\exp\left(-\frac{1}{2}\sum_{i=1}^{n} x_i^T (Q+L) x_i\right)\right) = -\frac{1}{2}\sum_{i=1}^{n} x_i^T (Q+L) x_i$$

We get:

$$\log \mathcal{L}(L, Q \mid \{x_i\}) = -\frac{nN}{2}\log(2\pi) + \frac{n}{2}\log\det(Q+L) - \frac{1}{2}\sum_{i=1}^{n} x_i^T (Q+L) x_i$$

Negative Log-Likelihood

The negative log-likelihood is often minimized instead of maximizing the log-likelihood:

$$-\log \mathcal{L}(L, Q \mid \{x_i\}) = \frac{nN}{2}\log(2\pi) - \frac{n}{2}\log\det(Q+L) + \frac{1}{2}\sum_{i=1}^{n} x_i^T (Q+L) x_i$$

So considering the term,

$$\sum_{i=1}^{n} x_i^T (Q+L) x_i = \sum_{i=1}^{n} \text{tr}(x_i^T (Q+L) x_i)$$

Using the cyclic property of the trace ($\text{Tr}(ABC) = \text{Tr}(CAB) = \text{Tr}(BCA)$), we can write:

$$\text{tr}(x_i^T (Q+L) x_i) = \text{tr}((Q+L)\sum_{i=1}^{n} x_i^T x_i)$$

Now, divide and multiply by n to get in terms of covariance matrix S:

$$\text{tr}(x_i^T (Q+L) x_i) = n \cdot \text{tr}\left((Q+L)\frac{1}{n} x_i^T x_i\right)$$

Since $\frac{1}{n} x_i^T x_i$ is the covariance matrix S:

$$\text{tr}(x_i^T (Q+L) x_i) = n \cdot \text{tr}\left((Q+L)S\right)$$

The negative log-likelihood can be written as:

$$-\log \mathcal{L}(L, Q \mid \{x_i\}) = \frac{nN}{2}\log(2\pi) - \frac{n}{2}\log\det(Q+L) + \frac{n}{2}\text{tr}((Q+L)S)$$

where

$$S = \frac{1}{n}\sum_{i=1}^{n} x_i x_i^T. \tag{10}$$

Cost Function

The constant term $\frac{nN}{2}\log(2\pi)$ can be ignored because it does change our optimum and $fracn2$ factor does not alter the argmin. The cost function $F(L, Q)$ to minimize becomes :

$$F(L, Q) = -\log\det(Q + L) + \text{tr}((Q + L)S) \tag{11}$$

A.2. Results

Figure 4. Initial Graph when range = 0.1. The darker the line, the higher the edge weight, and the lighter the line, the lower the edge weight.

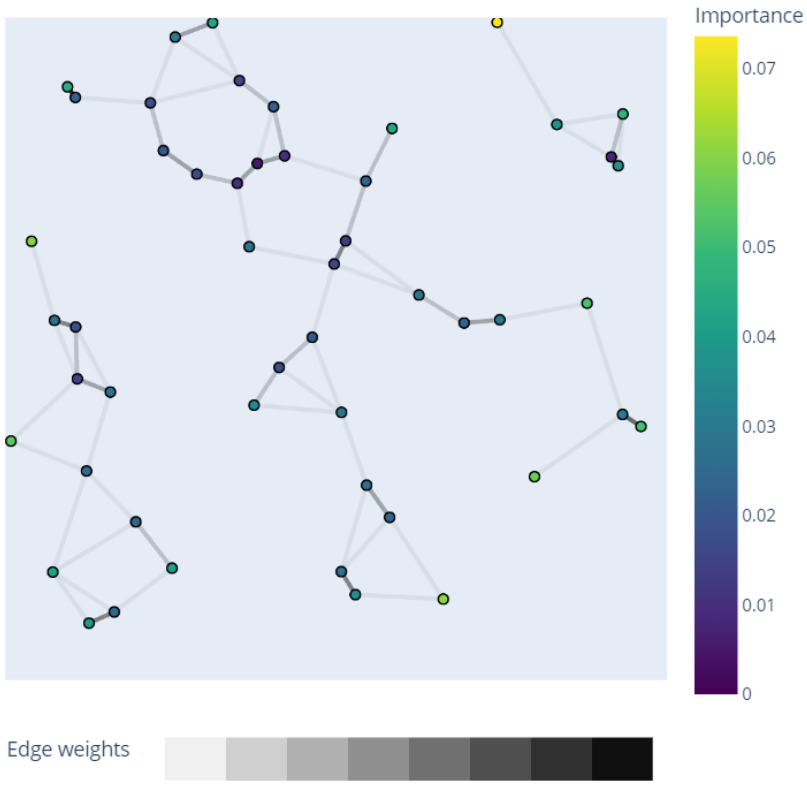


Figure 5. The resultant graph when range = 0.1 and with vertex importance.