Benchmarks for spectralGraphTopology

Vinícius

December 1, 2018

1 Clustering: warm-up

To warm-up, we consider the classical problem of clustering nodes distributed in $\mathbb{R}^2$. To do that, we generate 100 nodes per cluster distributed according to structures colloquially known as *two-moons*, *two-circles*, *three-circles*, and *worms*. Figure 1 depicts the results of learning the clusters structures using the proposed algorithm denoted as *Spectral Graph Learning* (SGL). As we can note, SGL is able to perfectly distinguish the cluster membership of all the datapoints for all datasets. Additionally, Figure 2 depicts the convergence trend of the terms in the objective function for the *two-moons* structure.

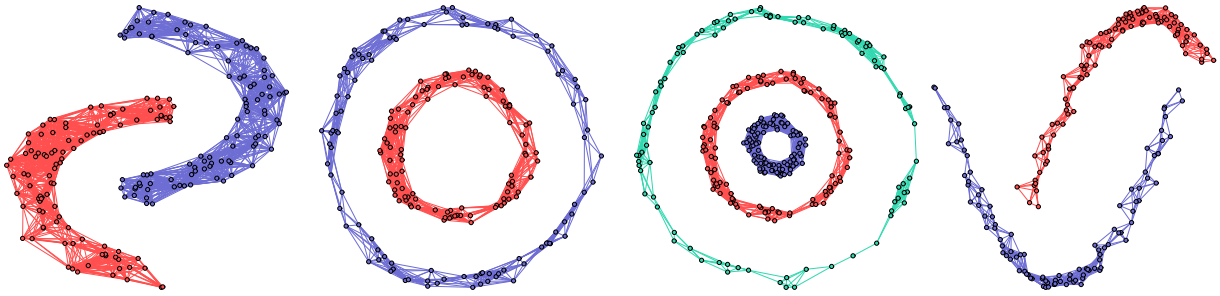


Figure 1: Clustering results learned by the SGL algorithm.

2 Grid graphs

A comprehensive performance evaluation of our spectral graph topology algorithm was performed considering grid graph models with 64 nodes denoted as $\mathcal{G}_{\text{grid}}^{(64)}$. We compare the performance of the SGL algorithm against state-of-the-art algorithms, namely CGL, CGL($\mathbf{A}$). Recall that CGL($\mathbf{A}$) stands for the CGL algorithm equipped with knowledge of the connectivity matrix $\mathbf{A}$, which gives the exact information of which nodes are connected to which ones.

The experimental setup is as follows. The edges of the graph model are sampled from $\text{Uniform}(0.1, 3)$. The Laplacian matrix estimation is carried out on the basis of T samples distributed according to $\mathcal{N}(\mathbf{0}, \mathbf{L}_{\text{grid}}^{\dagger})$. We repeat that experiment 20 times for every value of T and we average out the relative errors and F-scores[1].

Some hyperparameter tunning is required. For the SGL algorithm, we fix $\beta = 10$ for the values of T such that $T/N > 5$. Otherwise, we start with $\beta = 10^{-2}$, and we exponentially increase it up to $\beta = 4$. Additionally, we fix $\alpha = 0$.

For the CGL and CGL($\mathbf{A}$) algorithms, we follow the recipe by Eglimez, i.e., we choose α from $\{0\} \cup \left\{0.75^r \left(s_{\max}\sqrt{\log(N)/T}\right) | r = 1, 2, ..., 14\right\}$, such that the relative error between the estimated Laplacian and the ground truth is minimized, where $s_{\max}$ is the maximum off-diagonal element of the covariance matrix.

Figure 3 compares the performance of the algorithms for different sample size regimes for the grid graph model. As it can be noted, the SGL algorithm outperforms the CGL in both relative error and F-score senses.

[1]For the computation of the F-score, we ignore edge weight values which are less than 10^{-1}.

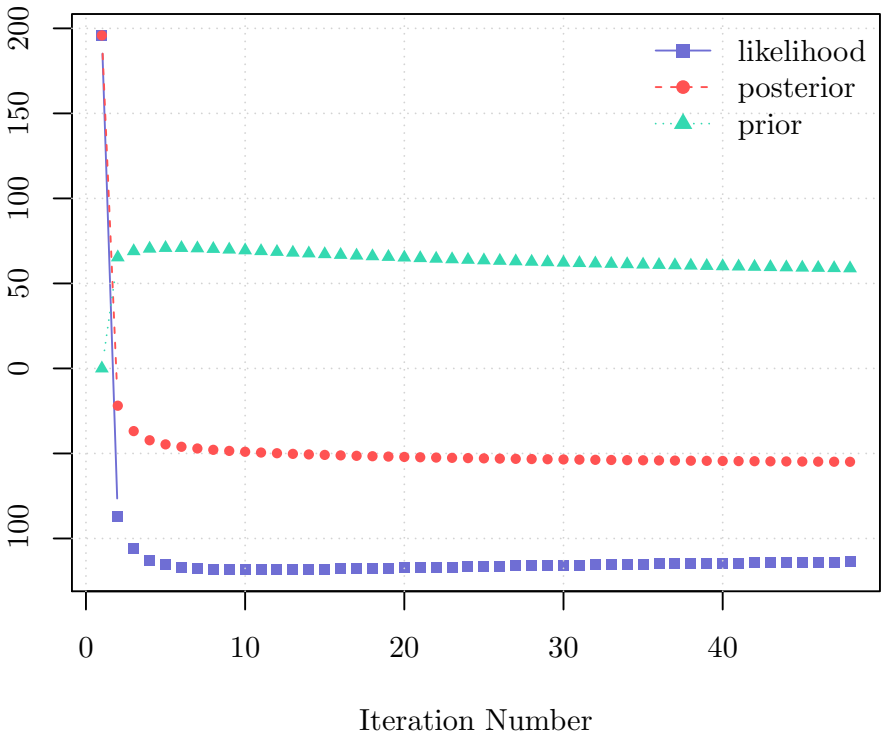


Figure 2: Convergence trend of the SGL algorithm for the *twomoon* dataset.

More precisely, for the case when the sample size is equal to the number of nodes, the difference in relative error and in F score are around 12% and 23%, respectively. As expected, with the additional prior knowledge of the connectivity matrix $\mathbf{A}$, the CGL($\mathbf{A}$) algorithm basically attains a perfect F-score for $T/N \geq 10$. However, the connectivity matrix is not always available in practical problems, especially in clustering tasks where the goal is precisely to understand the connectivity membership among the nodes. Nonetheless, the proposed SGL algorithm presents a comparable performance against CGL($\mathbf{A}$). For instance, at $T/N = 5$, the difference in relative error is only around 2.5%, and it keeps decreasing even further until virtually equal performance after $T/N = 100$, where the difference in relative error is around 1.2%. Additionally, we noted that the SGL algorithm requires far less tunning than CGL. At last, we add the relative error and the F-score for the Moore-Penrose inverse of the sample covariance matrix (ISCM) for completeness.

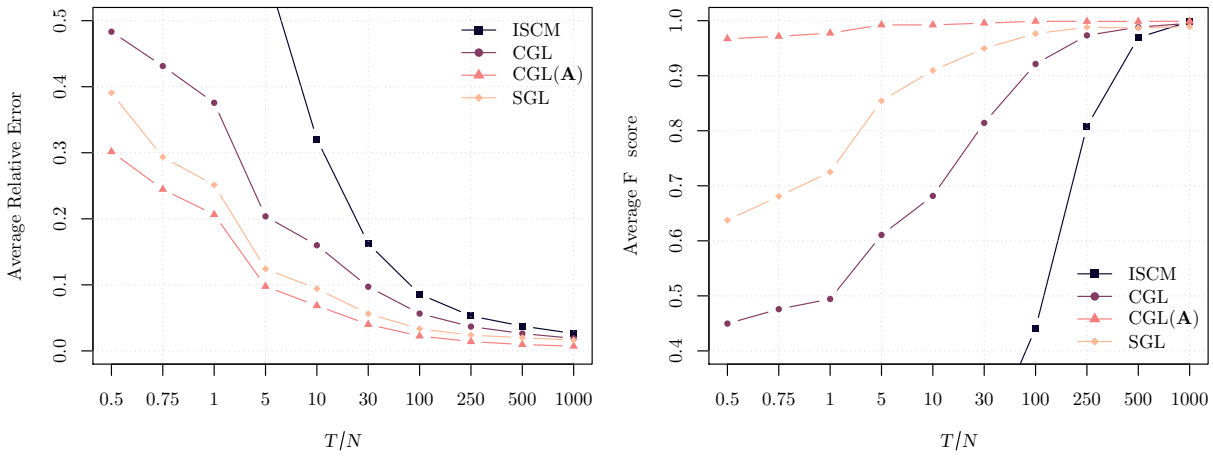


Figure 3: Average performance results for learning Laplacian matrix of a $\mathcal{G}_{\text{grid}}^{(64)}$.

3 Erdos-Renyi graphs

The experimentation on the Erdos-Renyi model is carried out in a similar fashion as for the grid graph. We consider an Erdo-Renyi graph $\mathcal{G}_{\mathsf{ER}}^{(64,p)}$ with $p = 0.1$, where p is the probability that a particular node is connected to any other node.

On what concerns hyperparameter tunning, we fix $\beta = 1.29$ and $\alpha = 1.3 \cdot 10^{-2}$ for the values of T such that $T/N > 1$, otherwise we fix $\alpha = 0$ and start with $\beta = 10^{-2}$ and we exponentially increase it up to $\beta = 1$.

The hyperparameter tuning for the CGL and CGL($\mathbf{A}$) algorithms is done as described for the grid model.

Figure **??** reveals that the algorithms SGL and CGL obtain a similar performance across the sample size regimes. Unsurprisingly, the algorithm CGL($\mathbf{A}$) attains nearly perferct F-score for $T/N > 10$.

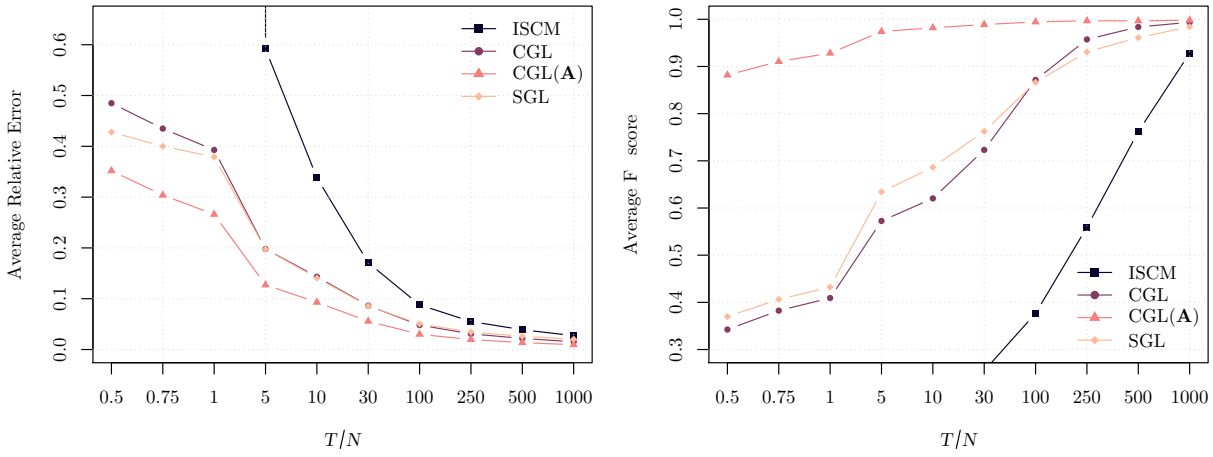


Figure 4: Average performance results for learning Laplacian matrix of a $\mathcal{G}_{\mathsf{ER}}^{(64,0.1)}$.

4 Animals dataset

The animals dataset consists of binary values which are the answers to questions such as "is warm-blooded?", "has lungs?", etc. There are in total 102 such questions, which make up the features, and 33 animal categories, which are the nodes. Figure **??** shows the results of estimating the graph Laplacian of the animals dataset using the GGL algorithm with $\alpha = 0.1$ and the proposed SGL algorithm with $\beta = 1/2$.

The evaluation of the estimated graphs is done by natural intuition, i.e., we expect that similar animals such as (*ant*, *cockroach*), (*bee*, *butterfly*), and (*trout*, *salmon*), would be clustered together, while presently virtually zero connection to other (group of) animals. Under this sense, it can be seen that the SGL algorithm yields a more intutive graph than the one learned by GGL.

5 Does a better initialization affect the final estimation?

The standard initialization is to take $\mathbf{w}_0$ as the off-diagonal elements of the pseudoinverse of the sample covariance matrix, denoted as $\hat{\mathbf{\Theta}}_{\mathsf{scm}}$. This is tantamount to solving the following problem

$$\begin{array}{ll} \underset{\mathbf{w}}{\text{minimize}} & ||\hat{\mathbf{\Theta}}_{\mathsf{scm}} - \mathcal{L}\mathbf{w}||_{\text{off},F}^2 \\ \text{subject to} & \mathbf{w} \geq \mathbf{0}. \end{array}$$

In order to improve the initial estimate, we consider the following optimization problem

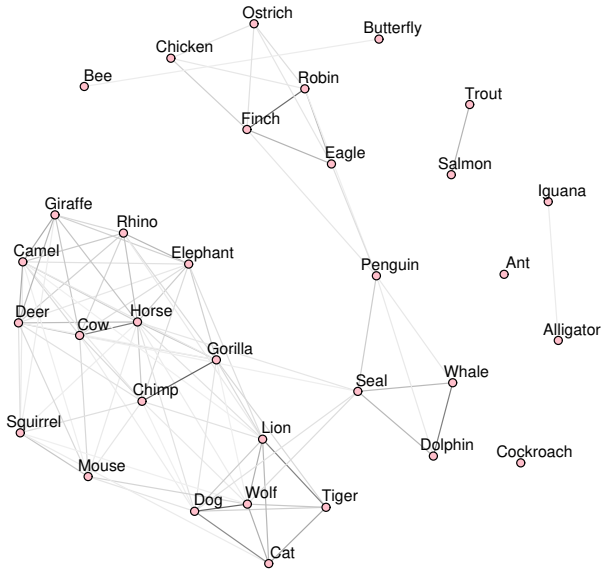
$$\begin{array}{ll} \underset{\mathbf{w}}{\text{minimize}} & ||\hat{\mathbf{\Theta}}_{\mathsf{scm}} - \mathcal{L}\mathbf{w}||_F^2 \\ \text{subject to} & \mathbf{w} \geq \mathbf{0}. \end{array} \tag{1}$$

Note that,

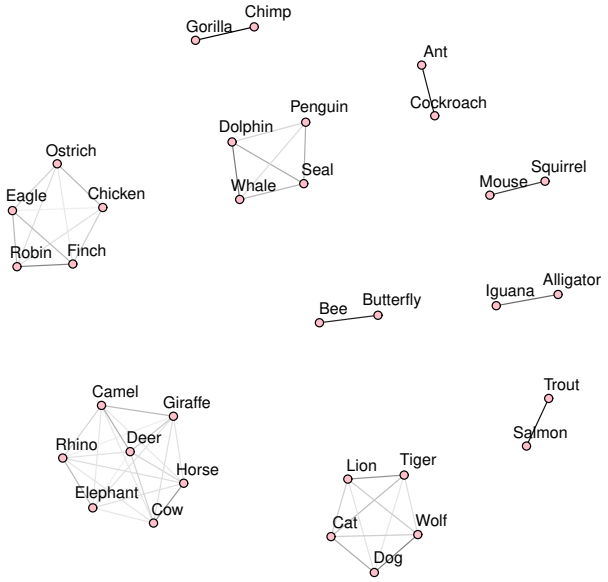
$$||\hat{\mathbf{\Theta}}_{\mathsf{scm}} - \mathcal{L}\mathbf{w}||_F^2 = ||\text{vec}\left(\hat{\mathbf{\Theta}}_{\mathsf{scm}}\right) - \text{vec}(\mathcal{L}\mathbf{w})||_2^2 = ||\text{vec}\left(\hat{\mathbf{\Theta}}_{\mathsf{scm}}\right) - \mathbf{R}\mathbf{w}||_2^2, \tag{2}$$

where $\mathbf{R} \triangleq \text{vec} \circ \mathcal{L}$.

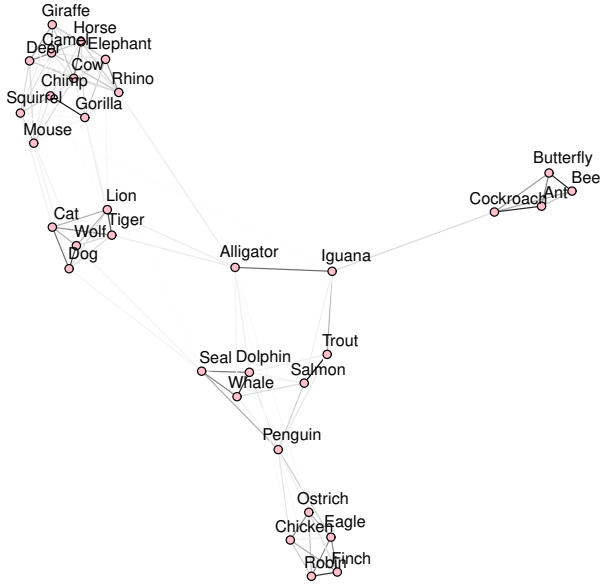
We denote the above algorithm as LLQP. Figure **??** shows the performance of the SGL algorithm when initialized by LLQP and ISCM. This experiment indicates that the SGL algorithm remains neutral to the different initilization procedures discussed here, even though it is clear that the LLQP shows a far better performance than ISCM.



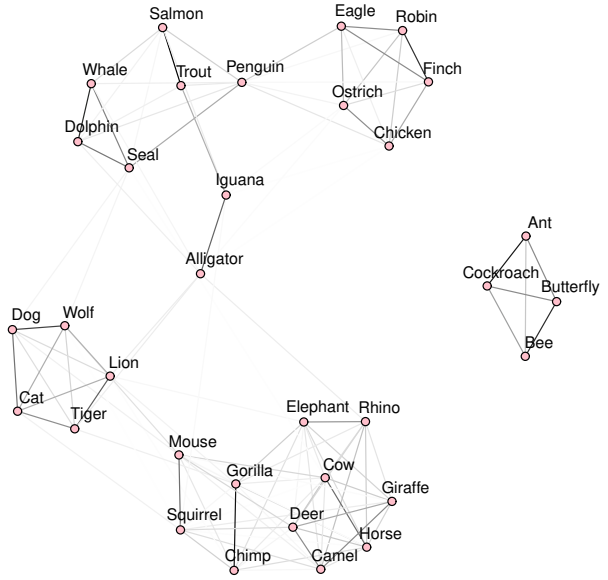
(a) GGL($\alpha = 10^{-1}$).



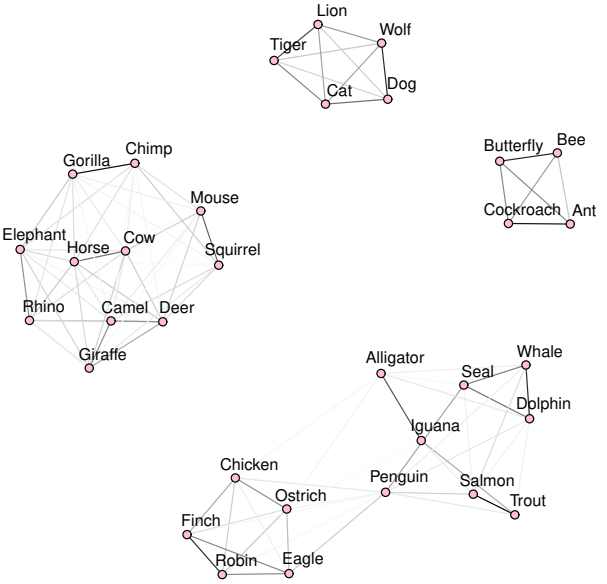
(b) SGL with $\beta = 1/2$ and $K = 10$.



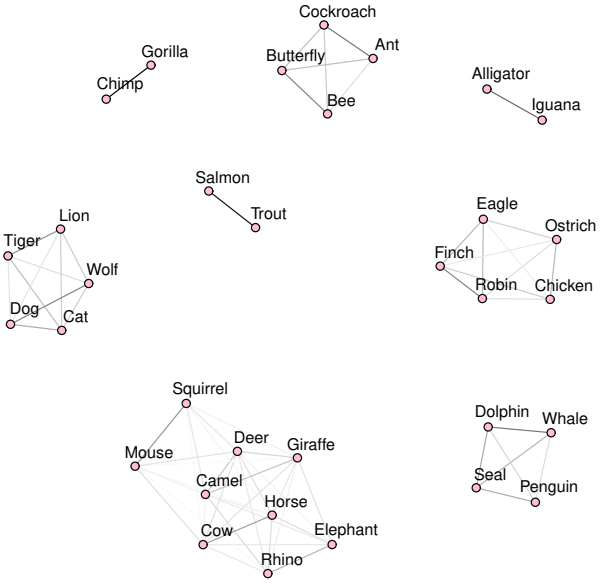
(c) SGL with $\beta = 1/2$ and $K = 1$.



(d) SGL with $\beta = 1/2$ and $K = 2$.



(e) SGL with $\beta = 1/2$ and $K = 4$.



(f) SGL with $\beta = 1/2$ and $K = 8$.

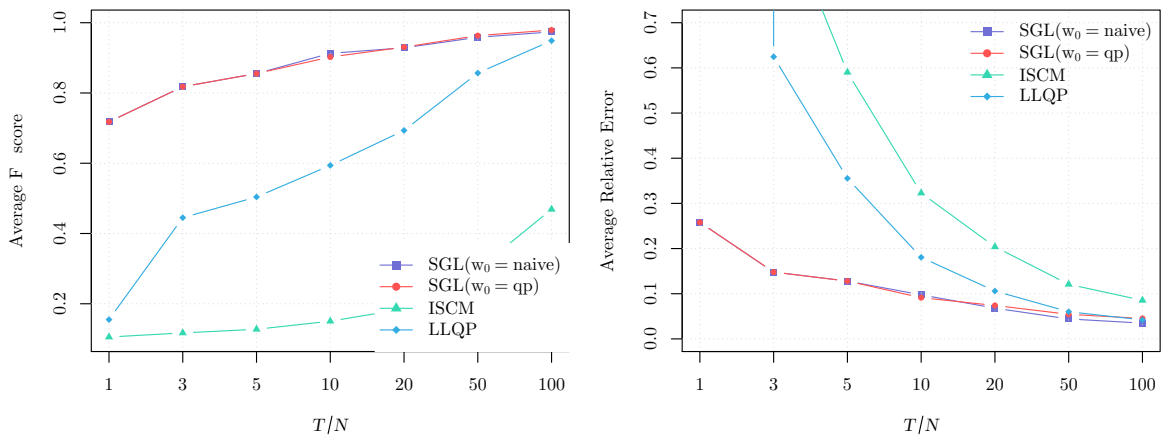


Figure 6: Relative error and F-score attained by SGL with different initialization strategies. SGL($\mathbf{w}_0 = \text{naive}$) and SGL($\mathbf{w}_0 = \text{qp}$) stand for the initialization with ISCM and LLQP, respectively.