# Graph learning perspectives in statistics and signal processing

Rodrigo C. G. Pena
LTS2, I&C, EPFL

***Abstract*—When one thinks of the word "mouse", it may come as an instance of a larger class of "rodents", or be associated to a particular computer peripheral, or yet, remind oneself of the mascot of an entertainment giant. Be that as it may, the point is that knowledge is often structured in a network-like fashion. Increasingly, machine learning algorithms have taken advantage of this sort of structure on the data to solve tasks more efficiently. This paper is concerned with the setting in which such a graphical representation is not available, but would be desirable. We discuss two approaches in the literature for learning similarity/dependence graphs from data, and also analyze a graph-based clustering algorithm to depict some of what can be done once we have a structured representation of the variables of the problem.**

***Index Terms*—graphs, inverse covariance, smoothness, spectral clustering**

## I. Introduction

GRAPHS are very efficient structures for representing data when one is interested in the relationships between the data points. When this underlying structure is available, predicting the signal value of a given node may reduce to observing its neighborhood. A good example of this possibility is the recommendation of ads in social networks: one can examine the kinds of ads on which the friends of a given user click to predict on which kinds of ads this same user would click.

In the past decade, graph-based methods have guaranteed their spot in the machine learning toolbox [1], and, more recently, have been showing great potential and flexibility in the area of signal processing [2]. In the statistics community, one is often interested in graphs to represent the conditional dependence between the random variables in a model [3]. Regardless of the perspective, a problem is shared among researchers in these fields: if their algorithms require graphical representations as input, what to do then when all they're are given are the data points themselves? For instance, we might have recorded the types of items bought by each user in an e-commerce store and want to generate a user-similarity graph in term of shopping patterns in order to improve the store's recommendation system.

There have been a few attempts to answer this question in the literature, based on intuition or strong model assumptions. The simplest and most used strategy in machine learning is based on the solution of the heat equation on a manifold [4]. It consists in assigning, for each pair of variables $i$ and $j$ an edge weight

$$W_{ij} = \exp\left(\frac{-\|x_i - x_j\|_2^2}{2\sigma^2}\right), \tag{1}$$

where $x_i$ is the observation vector of variable $i$, and $\sigma^2$ is a fixed parameter. In order to make the graph sparse, one can keep only the $k$ largest edges coming out of each node, creating thusly a $k$-Nearest Neighbors ($k$-NN) graph.

In what follows we present a couple of other solutions to the graph learning problem, showing possible links between the approaches, and also provide an example of what graph-based models can achieve. In Section II we present the model selection with sparseness prior by Banerjee *et al.* [5] for Gaussian and binary data. Then, in Section III, we discuss the work of Kalofolias [6], which learns a graph by enforcing the observed data to vary smoothly in this representation. Finally, Section IV deals not with graph construction, but with a graph-based algorithm, namely compressive spectral clustering, which will illustrate what can be done once we have an efficient representation of the similarities or dependencies of our variables.

### A. Notation and some definitions

Let $X = (x_1|x_2|...|x_N)^T \in \mathbb{R}^{N\times d}$, $x_j \in \mathbb{R}^d$, $\forall j \in \{1,...,N\}$, be our data matrix, consisting of $d$ observations of our $N$-dimensional signal or variable vector. An undirected graph representing the similarities or dependencies among the $N$ variables is a triple $G = (\mathcal{V}, \mathcal{E}, w)$ of a set $\mathcal{V}$ of $N$ nodes, a set $\mathcal{E}$ of up to $N(N-1)/2$ edges, and a set of edge weights $w$. The edge weight information can be encoded in a symmetric adjacency matrix $W$ such that $W_{ij}$ is the weight of the edge between nodes $i$ and $j$, $W_{ij} = 0$ if and only if nodes $i$ and $j$ are disconnected, $\forall i \neq j$, and $W_{ii} = 0, \forall i \in \{1,...,N\}$. Denote by $D$ the diagonal degree matrix defined by $D_{ii} = \sum_{j=1}^{N} W_{ij}$. A very important graph operator, the Laplacian, can then be written as $L = D - W$, in its combinatorial form, or $L = I - D^{-1/2}WD^{-1/2}$, in its normalized form. The eigenvectors of the graph Laplacian form a basis for defining a graph Fourier transform [2].

## II. Model Selection Through Sparse Maximum Likelihood Estimation for Multivariate Gaussian or Binary Data

This section discusses the statistical perspective on learning graphical representations for data, taking as example the work by Banerjee *et al.* [5]. Their focus is on data whose distribution

is Gaussian or binary, so that under a Markov random field framework, the parameters of these distributions encode the conditional dependence between variables.

### A. Problem

For a fixed data set $X$, model selection is often concerned with maximizing the *a posteriori* likelihood of the model $M$ given the data. Using Bayes' theorem, one can express this mathematically as follows:

$$\begin{aligned}\tilde{M} &= \arg\max_{M} \quad p(M|X) \\ &= \arg\max_{M} \quad p(X|M)p(M) \\ &= \arg\max_{M} \quad \log p(X|M) + \log p(M).\end{aligned} \tag{2}$$

The model in a Markov random field is the undirected graph on which the variables are the nodes. In order to solve Problem 2, one needs to establish the distribution of the data given the model, and also what's the prior distribution for the model. The former can be specified by modeling the data as having Gaussian, binary, Poisson, or any other known distribution. The latter can be given as penalty enforcing the graph to have some structure of interest.

### B. Solution

Banerjee *et al.* attack Problem 2 by first modeling the data as Gaussian. Model selection can then be seen as estimating the inverse covariance matrix $\Sigma^{-1}$. They also argue that the model should be simple, which is formalized in terms of parsimony: $\Sigma^{-1}$ should be sparse, *i.e.*, have few nonzero elements. Since the zeros in $\Sigma^{-1}$ correspond to conditional independence between pairs of variables, sparsity here has a convenient interpretation: the variables should have as few predictors as possible.

Under these circumstances, Problem 2 becomes

$$\tilde{\Sigma}^{-1} = \arg\max_{M \succ 0} \quad \log\det M - tr(SM) - \lambda\|M\|_{1,1}, \tag{3}$$

where $S$ is the sample covariance matrix, $\|\cdot\|_{1,1}$ is the elementwise $\ell_1$ norm, and $\lambda$ is a regularization parameter. When the number of observations $d$ is large, the inverse of the sample covariance might be an accurate estimator of $\Sigma^{-1}$, albeit not robust to outliers. When $d$ is small, however, $S$ might not even be invertible, so the prior term in Problem 3 can be seen as a regularizer guaranteeing the invertibility of $\tilde{\Sigma}^{-1}$. Indeed, by invoking the fact that the primal-dual gap is zero at the solution, and using simple triangle inequalities, Banerjee *et al.* prove that the eigenvalues of $\tilde{\Sigma}^{-1}$ are constrained by $\lambda$ to a strictly positive interval, which guarantees $\tilde{\Sigma}^{-1} \succ 0$.

Problem 3 has a non-smooth term (the prior) which makes it less straightforward to solve. The authors write $\lambda\|M\|_{1,1}$ as a dual norm, and invoke Sion's minimax theorem to write the dual to Problem 3 as

$$\begin{aligned}\tilde{\Sigma} = \arg\max_{W} \quad & \log\det W \\ \text{subject to} \quad & \|W - S\|_\infty \leq \lambda.\end{aligned} \tag{4}$$

Interestingly, while the primal solves for the inverse covariance, the dual finds the covariance matrix itself, displaying a nice symmetry between both problems. Problem 4 has a smooth objective with box constraints, and it's easy to see that the diagonal of the solution is given by $\tilde{\Sigma}_{ii} = S_{ii} + \lambda, \forall i \in \{1, ..., N\}$. Banerjee *et al.* propose solving for the rest of the matrix one row/column at a time, in a coordinate descent fashion.

Let $W_{\setminus j \setminus j} \in \mathbb{R}^{(N-1)\times(N-1)}$ denote matrix $W \in \mathbb{R}^{N\times N}$ with row/column $j$ removed. Also, let $s_j \in \mathbb{R}^{N-1}$ denote the $j$-th column of $S$ with the $j$-th element removed. Algorithm 1 then shows the authors' Block Coordinate Descent (BCD) scheme for solving Problem 4. Observe that Problem 5 is simply Problem 4 after fixing $W_{\setminus j \setminus j}$ and solving for the free row/column.

**Algorithm 1** Block Coordinate Descent (BCD) for solving 4

**Input:** $S$, $\lambda$, $\epsilon$
**Output:** $\tilde{\Sigma}$

1: $W \leftarrow S + \lambda I$
2: **while** $tr(W^{-1}S) - N + \lambda\|W^{-1}\|_{1,1} > \epsilon$ **do**
3:  **for** $j = 1$ to $N$ **do**
4:  Solve the problem

$$\tilde{w} \leftarrow \arg\min_{y} \{y^T (W_{\setminus j \setminus j})^{-1} y : \|y - s_j\|_\infty \leq \lambda\} \tag{5}$$

5:  Replace $j$-th column (resp. row) of $W$ with $\tilde{w}$ (resp. $\tilde{w}^T$)
6:  **end for**
7: **end while**
8: $\tilde{\Sigma} \leftarrow W$

BCD algorithms converge if there's a unique solution at each iteration. This turns out to be the case here because $W \succ 0$ at each iteration.

One question that remains is how to set $\lambda$. To answer this, first observe that if $\Sigma_{ij} = 0$, then variables $i$ and $j$ are completely independent. This means $i$ and $j$ are in different connected components on the graph, and we can in general permute the rows/columns of the covariance matrix so as to write it as a block-diagonal matrix $\Sigma = \text{blkdiag}(\mathcal{C}_{(1)}, ..., \mathcal{C}_{(l)})$, with one block for each connected component. It so happens that bounding the probability that the estimated connected component of each variable is not included in the true connected component of this variable is equal to bounding a standard statistic on the correlation coefficients under a null hypothesis. Banerjee *et al.* derive a formula for setting $\lambda$ so that this probability is as small as one desires.

If we are dealing with binary data instead, we can specify it's distribution with an Ising model:

$$p(x|\theta) = \exp\left[\sum_{i=1}^{N}\theta_i x_i + \sum_{i=1}^{N-1}\sum_{j=i+1}^{N}\theta_{ij}x_i x_j - A(\theta)\right], \tag{6}$$

where $A(\theta)$ is the log-partition function. In the corresponding graphical model, variables $i$ and $j$ are disconnected if and only

if $\theta_{ij} = 0$. We can collect all the parameters in a matrix $\Theta$,

$$\Theta = \begin{bmatrix} 0 & \theta_1 & \theta_2 & ... & \theta_N \\ \theta_1 & 0 & \theta_{12} & ... & \theta_{1N} \\ \theta_2 & \theta_{12} & 0 & ... & ... \\ ... & ... & ... & ... & ... \\ \theta_N & \theta_{1N} & ... & ... & 0 \end{bmatrix}, \quad (7)$$

and enforce a sparsity prior on it to solve Problem 2 for this setup. If we apply a log-determinant relaxation on $A(\theta)$, and solve for each row/column of $\Theta$ at a time we arrive at essentially the same as Algorithm 1, but with a slightly different initialization: $W \leftarrow S + (1/3)I$.

### C. Results

**Simulations**. Banerjee *et al.* run several numerical experiments, showing first that $\tilde{\Sigma}^{-1}$ is a much better estimator of $\Sigma^{-1}$ than $S^{-1}$ (when the inverse exists), and that even determining the threshold level for $S^{-1}$ to estimate the sparsity pattern of $\Sigma^{-1}$ is hard. They also show that there's an optimal interval of values for $\lambda$ for which $\tilde{\Sigma}^{-1}$ perfectly estimates the sparsity pattern of $\Sigma^{-1}$, an evidence that setting $\lambda$ is not so trivial. Also, if noise is added to the sample covariance matrix, the recovery error is still zero if $\lambda$ is set to equal the noise level. Finally, treating the estimated matrix as a machine that classifies edges as either zero or nonzero, the classifier is fairly stable to discarding observations, both in the Gaussian and binary cases.

**Real data**. The authors claim gene expression data is often assumed to be normally distributed, and test Algorithm 1 on two different data sets of this kind. For both of them the evaluation is subjective, by analyzing if the estimated dependence relationships make some biological sense. As an example, for the first data set, with $N = 6136$ variables and $d = 253$ observations, they observe that genes associated with iron homeostasis are in the same estimated connected component. They also analyze a set of US senate voting data, which they model as binary, and the evaluation is subjective as well. In the estimated graphical model, most Republican (resp. Democrat) senators have only Republican (resp. Democrat) neighbors. Senators with neighbors in both parties seem to be those whose media statements support this openness in voting strategy.

### D. Discussion

Algorithm 1 needs to invert $(N-1) \times (N-1)$ matrices $N$ times at each sweep. This results in a total complexity of $O(KN^4)$, for a fixed number of $K$ sweeps. Since they were not able to compute a bound on $K$, they also propose Nesterov's first order smoothing algorithm [7] to solve Eq. 3 and derive a rigorous complexity estimate with respect to accuracy: $O(N^{4.5}/\epsilon)$. As a comparison, if one were to use interior point methods to solve Problem 1, the complexity would have been $O(N^6 log(1/\epsilon))$. Despite the precise bound on Nesterov's algorithm, Banerjee *et al.* observe that a few sweeps is usually enough in practice for the convergence of BCG.

But BCG is not the fastest algorithm in the literature to solve Problem 3. Note first that the dual of Problem 5 in Algorithm 1 is

$$\arg\min_z z^T(W_{\setminus j \setminus j})z + z^T s_j + \lambda \|z\|_1, \quad (8)$$

which is a Lasso regression of the free row/columns onto the others. This is the main inspiration for the graphical Lasso (glasso) algorithm [8]. This simple change in perspective allows for a significant decrease in the complexity of the estimation of the inverse covariance to $O(N^3)$ because one can then employ a pathwise coordinate optimization scheme [9] to compute the solution to Problem 3. In fact, Banerjee *et al.* observe that glasso is faster themselves in their numerical experiments. There are more scalable algorithms nowadays that employ quadratic approximations to Problem 3, but the graphical Lasso is still a standard in inverse covariance estimation.

In the end, the strength of the work Banerjee *et al.* lies not in providing an efficient algorithm for sparse inverse covariance estimation, but in providing a solid theoretical ground for the problem to inspire other approaches.

## III. How to learn a graph from smooth signals

Let us assume now we have reason to believe our data lives in a smooth manifold, and its graph is but a sampling of this manifold, such that large edge weights indicate its respective nodes are close in the original manifold. Then, assume as well that the signal values vary smoothly in the neighborhood of any given node. This is the basic idea behind the work of Kalofolias [6].

### A. Problem

A simple and popular metric of local smoothness on graphs is $(1/2)\sum_{i,j} W_{ij}\|x_i - x_j\|_2^2$, which can be written as $tr(X^T L X)$, with help from the combinatiorial graph Laplacian. This quadratic form has been often used in graph-regularized signal recovery/denoising problems to enforce smoothness on the estimated signal. If we now fix the signals and consider the complementary task of learning a graph on which these observations are smooth, then we can write the general problem to solve as

$$\min_{L \in \mathcal{L}} \quad tr(X^T L X) + f(L), \quad (9)$$

where $\mathcal{L}$ is the set of valid Laplacian matrices. The function $f$ should prevent the trivial solution $L = 0$ and also impose additional structure via prior information. Note that the problem is written with respect to $L$, but the adjacency matrix $W$ can be uniquely determined from the Laplacian. The standard Gaussian edge assignment from Equation 1 can be seen as a solution to an instance of this problem with $f(W) = 2\sigma^2 \sum_{i,j} W_{ij}(\log W_{ij} - 1)$.

We can also think of smoothness in terms of graph filtering [2]. Let $L = U\Lambda U^T$ be the diagonalization of the graph Laplacian. We mentioned in the Introduction that for any $z \in \mathbb{R}^N$, $\hat{z} = U^T z$ defines a graph Fourier transform of signal $z$. Now, let $0 = \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_N$ denote the eigenvalues of $L$. A graph low-pass filter is a decreasing

function for large values of $\lambda$. Several smooth graph signal models in the literature can be seen as low-pass filtering of a possibly non-smooth original signal $z_0$. Three examples of such interpretation are summarized below:

$$\begin{aligned} &\text{Tikhonov regularization:} && z = (\alpha L + I)^{-1} z_0 \\ &\text{Dong } \textit{et al.}\text{'s [10]:} && z = \sqrt{L^\dagger} z_0,\ z_0 \sim \mathcal{N}(0, I) \\ &\text{Heat diffusion:} && z = \exp(-tL) z_0 \end{aligned} \tag{10}$$

### B. Solution

Kalofolias starts by showing that the trace term in Problem 9 can be written in a way that reveals further information about the problem. Let $\odot$ denote the Hadamard (elementwise) product, and $Z$ denote the pairwise distances matrix defined by $Z_{ij} = \|x_i - x_j\|_2^2$. Then, we obtain $tr(X^T L X) = \frac{1}{2}\|W \odot Z\|_{1,1}$, which is a weighted sparsity term on the edge weights. Nodes whose observations are similar (small $Z_{ij}$) are allowed to have a nonzero edge weight between them, while the edge weight between nodes with large $Z_{ij}$ is likely to be zero. One should also note that this means there is little use in making $f$ in Problem 9 have a term of the type $\|L\|_{1,1}$: enforcing our fixed observations to be smooth naturally implies enforcing $L$ to be sparse.

The regularization function is designed following two principles: i) each node should have at least one neighbor, *i.e.*, its degree should be at least one, and ii) it should be possible to control the sparsity pattern of the solution. This design gives rise to the following problem:

$$\min_{W \in \mathcal{W}_m} \quad \|W \odot Z\|_{1,1} - \alpha \mathbf{1}^T \log(W\mathbf{1}) + \beta \|W\|_F^2, \tag{11}$$

where $\alpha$ and $\beta$ are regularization parameters and $\mathcal{W}_m$ is the set of valid adjacency matrices. The log term in Problem 11 acts as a logarithmic barrier on each individual node degree, ensuring none of those is zero. The Frobenius norm term acts jointly with the $\ell_1$-norm term in controlling the sparsity pattern: since it is a quadratic term, it penalizes large edge weights much more than small ones, allowing for denser graphs as we increase $\beta$. If we fix the scale $\|W\| = s$ of the graph, for any norm $\|\cdot\|$, Kalofolias shows that we can couple $\alpha$ and $\beta$, effectively having to set only a single parameter.

To devise a scalable scheme for solving Problem 11, Kalofolias turns to primal-dual algorithms, as those described in [11]. Before delving into the primal-dual method, one should notice that because our adjacency matrix is symmetric, the only free variables in Problem 11 are the $N(N-1)/2$ lower-diagonal elements of $W$. We can assemble these elements into a vector $w \in \mathbb{R}_+^{N(N-1)/2}$ which we can treat as our optimization variable.

Having done this, the next step is to write Problem 11 in the standard form for primal-dual techniques,

$$\min_{w \in \mathbb{R}_+^{N(N-1)/2}} \quad f_1(w) + f_2(Kw) + f_3(w) \quad . \tag{12}$$

The functions $f_1$ and $f_2$ should have easy-to-compute proximity operators, while $f_3$ should have a Lipschitz-continuous gradient with constant $\zeta \in ]0, \infty[$. The operator $K$ should define a linear map from the primal variable space to the dual variable space.

Regarding Problem 11, we can take our dual variable to be the degree vector $d \in \mathbb{R}_+^N$, so that $K$ is the linear operator satisfying $W\mathbf{1} = Kw$. We can also enforce the nonnegativity of $w$ through an indicator function $\mathbb{I}_{\{w \succeq 0\}}$ which takes the value 0 if $w \succeq 0$, and $\infty$ otherwise. Finally, if we vectorize $Z$ in the same manner as we did with $W$, the $\ell_1$ term becomes the simple dot product $2w^T z$. Those ideas are summarized in the choice of functions below:

$$\begin{cases} f_1(w) = & \mathbb{I}_{\{w \succeq 0\}}(w) + 2w^T z \\ f_2(d) = & -\alpha \mathbf{1}^T \log(d) \\ f_3(w) = & \beta \|w\|_2^2, \qquad \text{with } \zeta = 2\beta \end{cases} . \tag{13}$$

Algorithm 2 shows a straightforward adaptation of Algorithm 6 in [11] for these particular choices of functions. This algorithm is deemed forward-backward-forward because it first performs gradient descents (Steps 2 and 3), then proximity operations (Steps 4 and 5), and finally gradient descents again (Steps 6 and 7)

**Algorithm 2** Forward-Backward-Forward primal-dual algorithm for solving 12

**Input:** $z$, $\alpha$, $\beta$, $w^{(0)} \in \mathbb{R}_+^{N(N-1)/2}$, $d^{(0)} \in \mathbb{R}_+^N$, stepsize $\gamma \in ]0, 1/(\zeta + \|K\|_2)[$, and tolerance $\epsilon$
**Output:** $w^{(n)} \in \mathbb{R}_+^{N(N-1)/2}$

1: **for** $n = 1$ to $n_{max}$ **do**
2: $\quad y_1^{(n)} \leftarrow w^{(n-1)} - \gamma(\nabla f_3(w^{(n-1)}) + K^T d^{(n-1)})$
$\quad\quad = w^{(n-1)} - \gamma(2\beta w^{(n-1)} + K^T d^{(n-1)}$
3: $\quad y_2^{(n)} \leftarrow d^{(n-1)} + \gamma K w^{(n-1)})$
4: $\quad p_1^{(n)} \leftarrow \text{prox}_{\gamma f_1} y_1^{(n)}$ ▷ elementwise
$\quad\quad = \max(0, y_1^{(n)} - 2\gamma z)$
5: $\quad p_2^{(n)} \leftarrow \text{prox}_{\gamma f_2^*} y_2^{(n)}$ ▷ elementwise
$\quad\quad = \left(y_2^{(n)} - \sqrt{(y_2^{(n)})^2 + 4\alpha\gamma}\right)/2$
6: $\quad q_1^{(n)} \leftarrow p_1^{(n)} - \gamma(\nabla f_3(p_1^{(n)}) + K^T p_2^{(n)})$
$\quad\quad = p_1^{(n)} - \gamma(2\beta p_1^{(n)} + K^T p_2^{(n)})$
7: $\quad q_2^{(n)} \leftarrow p_2^{(n)} + \gamma K p_1^{(n)})$
8: $\quad w^{(n)} \leftarrow w^{(n-1)} - y_1^{(n)} + q_1^{(n)}$
9: $\quad d^{(n)} \leftarrow d^{(n-1)} - y_2^{(n)} + q_2^{(n)}$
10: $\quad$ **if** $\|w^{(n)} - w^{(n-1)}\|_2 / \|w^{(n-1)}\|_2 \leq \epsilon$ **and**
11: $\quad\quad \|d^{(n)} - d^{(n-1)}\|_2 / \|d^{(n-1)}\|_2 \leq \epsilon$ **then**
12: $\quad\quad$ **return** $w^{(n)}$
13: $\quad$ **end if**
14: **end for**

### C. Results

The baseline for the experiments is the $k$-NN method with weights chosen as in Equation 1. The results are compared with the work of Dong *et al.* [10], which the author considers the state-of-the-art. Kalofolias shows that the problem solved by Dong *et al.* can be written in a similar fashion as Problem 11, granted, with a different penalty on the degrees of the nodes. He also provides a scalable implementation for this problem based on the same primal-dual technique as in Algorithm 2.

**Artificial data**. The author generates four types of ground-truth graphs: Uniform and Non-uniform Random Geometric, Erdős-Rényi, and Barabási-Albert. The first two are formed by sampling points on the plane and connecting them with weights as in Equation 1, with a posterior thresholding of small edge weights. Kalofolias refers to them as "manifold-based", in opposition to the last two. For each graph, he then creates noisy samples from the three kinds of signal models in Equation 10 to serve as observations. Problem 11 is solved for each signal-graph combination. Performance is measured by means of relative edge errors, relative degree errors, and f-score on the binary edge patterns. The errors are smaller (and the f-score is larger) for the solution of Problem 11 than for those of the state-of-the-art and the baseline in almost all configurations. The performance also seems to be slightly better when estimating the "manifold-based" graphs.

**Real data**. The ground-truth graph is not known, so performance is measured with respect to spectral clustering and label propagation tasks. If the data is known to have well-defined clusters or classes, then a good graph should reflect this property and, hence, lead to a good performance in the aforementioned tasks. **1)** USPS digits: 1001 images of 10 classes sampled non-uniformly. Kalofolias uses the standard spectral clustering algorithm [12] (see also Algorithm 3), and label propagation with $10\%$ known labels. The smallest clustering/classification errors are similar whether the graph is issued from Problem 11 or from the state-of-the-art [10]. Nevertheless, the clustering/classification is more robust in low edge density settings when using graphs learned with Problem 11. **2)** MNIST: the author tests label propagation for classifying between digits '1' and '2', varying the class proportions in the known label set. Using graphs learned with Problem 11 results in less missclassified and unclassifiable digits. The graphs from Problem 11 also contribute to an improved stability to changes in class proportions.

### D. Discussion

The complexity of Algorithm 2 is $O(N^2)$ per iteration. Moreover, parts of the algorithm can be run in parallel, which can help reduce the running time. Notice the difference in comparison to Banerjee *et al.*'s block coordinate descent scheme. Even though they solve different problems, the lesson to be taken from Kalofolias' solution is to take advantage of the symmetry at hand to turn a matrix problem into a simpler vector problem.

The superior performance of Problem 11 on the measured tasks might be due to its ability to generate better connected graphs than the state-of-the-art [10] for the same sparsity level. As evidence for the importance of enforcing connectivity, we can consider the low density setting, *i.e.*, when nodes have very few neighbors on average. In this case, Kalofolias' experiments show that learning the graph with the $k$-NN method, which enforces each node to have *exactly* $k$ neighbors, can sometimes result in smaller clustering/classification errors than when using Problem 11, which enforces each node to have *at least* one neighbor.

## IV. Compressive spectral clustering

This section deals with a graph-based algorithm, to show in a practical sense some of what can be achieved once we have a good graphical representation of our data. In what follows, we discuss the work of Tremblay *et al.* towards a compressive spectral clustering algorithm [13].

### A. Problem

Given a graph representing our $N$ variables, admit we have reason to believe this data might be organized into $k$ different clusters. In graph terms, this means there are $k$ sets of nodes very well connected with nodes from the same set, while nodes from different sets are poorly connected. This is often the case when observing communities in social networks, or if we have a labeling problem with $k$ different classes. Stated in simple terms, the problem is then to assign a cluster label to each node, while observing only the given graph.

Mathematically, we can rewrite the goal as follows. Let $c_j, \forall j \in \{1, ..., k\}, \in \mathbb{R}^N$ denote the ground-truth indicator vector of cluster $\mathcal{C}_j$, that is,

$$(c_j)_i := \begin{cases} 1 & \text{if} \quad i \in \mathcal{C}_j \\ 0 & \text{otherwise} \end{cases}. \tag{14}$$

We want to estimate each vector $c_j$, while observing only the graphs's adjacency matrix $W$, or equivalently, its Laplacian $L$, which in this section we consider in the normalized form.

As shown, for instance, by Koren [14], the first eigenvectors of the Laplacian form the projection vectors of an Euclidean embedding of the variables such that the nodes are maximally separated, while maintaining well-connected nodes close together. In this embedding, this means that if our data has clusters, then the nodes in each cluster are going to be close together, while being far from nodes in other clusters. From there, we can easily assign labels to each node according to a $k$-partition of the data into Voronoi cells, where $k$ is the number of clusters.

Those are the main ideas behind the standard spectral clustering algorithm, detailed in Algorithm 3. For a number $k$ of clusters, we compute the $k$ first eigenvectors of $L$, according to an increasing ordering of the respective eigenvalues. Were our clusters disconnected among themselves, those eigenvectors would correspond exactly to the indicator functions of each cluster. The last step consists in running $k$-means on the $N$ features formed by concatenating the $k$ eigenvectors.

### B. Solution

The goal of the work of Tremblay *et al.* is to speed up the computation of the features as well as the running time of $k$-means. This is done by leveraging results from graph-filtering of random signals and random sampling of bandlimited graph signals [15]. The outcome is an approximation of the SC algorithm, but one for which the error can be controlled. The full algorithm is detailed in Algorithm 4, and we discuss each of its steps in the following paragraphs.

**Algorithm 3** Spectral Clustering (SC) [12]

**Input:** Laplacian $L$, number of clusters $k$
**Output:** $\tilde{C} = (\tilde{c}_1|\tilde{c}_2|...|\tilde{c}_k) \in \{0,1\}^{N\times k}$

1: Compute the first $k$ eigenvectors of $L$, $U_{[k]} := (u_1|u_2|...|u_k) \in \mathbb{R}^{N\times k}$
2: Form $Y_{[k]}$ by normalizing the rows of $U_{[k]}$: $(Y_{[k]})_{ij} \leftarrow (U_{[k]})_{ij}/\sqrt{\sum_{j=1}^{k}(U_{[k]})_{ij}^2}$
3: Define the feature vector of each node $i \in \{1,...,N\}$ as the $i$-th row of $Y_{[k]}$: $f_i \leftarrow (Y_{[k]})^T\delta_i$
4: Run $k$-means with the Euclidean distance

$$D_{ij} := \|f_i - f_j\|_2 \tag{15}$$

**Algorithm 4** Compressive Spectral Clustering (CSC) [12]

**Input:** Laplacian $L$, number of clusters $k$, other parameters typically set as $n \leftarrow \lceil 2k\log k\rceil$, $d \leftarrow \lceil 4\log n\rceil$, $p \leftarrow 50$, and $\gamma \leftarrow 10^{-3}$
**Output:** $\tilde{C} = (\tilde{c}_1|\tilde{c}_2|...|\tilde{c}_k) \in \mathbb{R}^{N\times k}$

1: Estimate the $k$-th eigenvalue of $L$, $\lambda_k$
2: Compute the polynomial approximation $\tilde{h}_{\lambda_k}$ of order $p$ of the ideal low-pass filter $h_{\lambda_k}$
3: Generate $R = (r_1|r_2|...|r_d) \in \mathbb{R}^{N\times d}$ by drawing $d$ independent random vectors $r_i \sim \mathcal{N}(0, d^{-1}I), \forall i \in \{1,...,d\}$
4: Filter $R$ with $\tilde{H}_{\lambda_k}$ and define the feature vector of each node $i$ as the normalized $i$-th row of the resulting matrix:

$$\tilde{f}_i \leftarrow [(\tilde{H}_{\lambda_k}R)^T\delta_i]/\|(\tilde{H}_{\lambda_k}R)^T\delta_i\|_2$$

5: Generate a random sampling matrix $M \in \{0,1\}^{n\times N}$ and keep only $n$ feature vectors: $(\tilde{f}_{\omega_1}|\tilde{f}_{\omega_2}|...|\tilde{f}_{\omega_n})^T := M(\tilde{f}_1|\tilde{f}_2|...|\tilde{f}_N)^T$
6: Run $k$-means on the reduced dataset, with the Euclidean distance $\tilde{D}^r_{ij} := \|f_{\omega_i} - f_{\omega_j}\|_2, \forall(i,j) \in \{1,...,n\}^2$, to obtain reduced indicator vectors $\tilde{c}^r_j \in \mathbb{R}^n$, $j \in \{1,...,k\}$
7: Solve

$$\tilde{c}_j \leftarrow \arg\min_y \|My - \tilde{c}^r_j\|_2^2 + \gamma y^T(1-\tilde{H}_{\lambda_k})y \tag{16}$$

for each $j \in \{1,...,k\}$ to obtain the $k$ cluster indicator vectors on the full set of nodes

*1) Eigendecomposition bottleneck:* Let $h_{\lambda_k}$ denote the ideal low-pass graph filter with cutoff $\lambda_k$, that is, $h_{\lambda_k}(\lambda) = 1$ if $\lambda \leq \lambda_k$, and equal to 0 otherwise. Note that we can write this filter in its matrix form as $H_{\lambda_k} = Uh_{\lambda_k}(\Lambda)U^T = U_{[k]}U^T_{[k]}$.

Now, let $f_i$ and $\tilde{f}_i$, $\forall i \in \{1,...,N\}$ be defined as in Algorithms 3 and 4, respectively (setting for now $\tilde{H}_{\lambda_k} = H_{\lambda_k}$). Consider also the random matrix $R$ defined in Algorithm 4. Then, Tremblay *et al.* show that $\|\tilde{f}_i - \tilde{f}_j\|_2 = \|R^TU_{[k]}(f_i - f_j)\|_2$ is strongly concentrated around its mean, $\|f_i - f_j\|_2$, $\forall(i,j) \in \{1,...,N\}^2$, provided that $d = O(\log N)$. This means that if we low-pass filter enough random signals, the pairwise Euclidean distances are preserved with high probability, with respect to the distances in the standard SC algorithm.

From the full eigendecomposition in Algorithm 3, we have transitioned to the problem of estimating the $k$-th eigenvalue of $L$, *i.e.*, $\lambda_k$. Fortunalety, this also reduces to graph low-pass filtering of random signals. Puy *et al.* [15] show that if $d = O(\log N)$, then the total energy ($\ell_2$ sense) of the filtered signals $H_{\lambda_k}r_i$, $\forall i \in \{1,...,d\}$ is a good estimator of the number of eigenvalues below $\lambda_k$. Tremblay *et al.* suggest then we proceed by dichotomy, varying the cutoff eigenvalue of the low-pass filter until the total energy of the filtered signals is approximately equal to $k$.

*2) $k$-means bottleneck:* The authors attack this bottleneck by subsampling the features to cluster. The idea is that if one is able to perform the clustering for a subset of the nodes, but still accurately and efficiently interpolate the labels for the remaining nodes, then the complexity of the $k$-means step is reduced. The sampling matrix $M$ is generated by first drawing $n$ indices $\Omega := \{\omega_1,...,\omega_n\}$, then assigning $M_{ij} = 1$, if $j = \omega_i$, and equal to 0 otherwise.

Borrowing from results in [15], Tremblay *et al.* state that there always exists a sampling distribution for generating the indices $\Omega$ such that $M$ satisfies a Restricted Isometry Property (RIP), provided that $n = O(k\log k)$. It's this RIP that makes the reconstruction of the labels of the full set of nodes possible by solving Problem 16. Moreover, the recovery error is shown in [15] to depend on the ratio $(1-\tilde{h}_{\lambda_k}(\lambda_k))/(1-\tilde{h}_{\lambda_k}(\lambda_{k+1}))$. We know from spectral graph theory that if a graph has $k$ clusters, then there is a spectral gap between $\lambda_k$ and $\lambda_{k+1}$, so the reconstruction in Problem 16 will be good precisely if the data is clusterable into $k$ classes.

*3) Note about fast filtering:* In order for the filtering steps to be efficient, one should approximate $H_{\lambda_k}$ by a polynomial on $L$, $\tilde{H}_{\lambda_k} = \sum_{l=1}^{p}\alpha_l L^l$. If we do that, filtering reduces to a series of matrix-vector multiplications with an often sparse matrix $L$. Despite the approximation, the authors show that if the order $p$ of the polynomial is large enough, then filtering $d = O(\log n)$ is sufficient to preserve the Euclidean distances in the subsampled feature set with high probability. Nevertheless, instead of providing a theoretical a lower bound on $p$ for specific polynomial approximations, they simply state that $p = 50$ yields good results experimentally.

### C. Results

**Simulations**. The graphs are constructed via the Stochastic Block Model (SBM), which is regarded as benchmark for spectral clustering. This is because in the SBM one is able to set the intra- and inter-cluster connection probabilities, allowing the creation of more or less clusterable graphs. The performance is measured by the Adjusted Rand similarity index: relative number of agreements between ground-truth and estimated clusters, adjusted for expected results with random clustering. In the first experiments, the authors simply show that the clustering performance saturates at the default parameter values, justifying their choices as default values. The subsequent experiments test for accuracy and computation time. For large $k$, CSC is orders of magnitude faster than SC or Boutsidis' and Gittens' algorithm [16] (power method as surrogate for the eigendecomposition of the Laplacian). The clustering accuracy of CSC is only slightly lower than that of the methods just mentioned, the loss being orders of magnitude smaller than the gains in speed.

**Real data**. Tremblay *et al.* use the Amazon co-purchasing network [17], for which a ground-truth clustering is not known. The performance is measured then in terms of computation time for a given level of modularity (fraction of edges that fall in each cluster minus the expected fraction with random clustering). Here the gains in speed are also very pronounced: for $k = 1000$, SC still had not converged in the first $40$ hours, while CSC converged after $10$ hours. Boutsidis' and Gittens' method did not converge at all, maybe due to the features generated by the power method not being able to distinguish enough between classes.

### D. Discussion

While most works in the literature attempt to speed up the eigendecomposition *or* the $k$-means bottlenecks, CSC succeeds in attenuating both problems at the same time.

The complexity of CSC is $O(kn \log n + p|\mathcal{E}|(\log N + \log n + k))$: the first term comes from $k$-means clustering the reduced set of features, while the other terms come from the fast filtering of random signals and the gradient descent steps to solve Problem 16. In practice, we deal with sparse graphs *i.e.*, those for which $|\mathcal{E}| = O(N)$, therefore reducing the complexity of CSC to $O(k^2 \log^2 k + pN(\log N + k))$.

In comparison, the standard SC algorithm, using the popular ARPACK package for the eigendecomposition of $L$, has a complexity of $O(Nk^2 + k^3)$. It shows a slightly better dependence on $N$ than CSC, but a way worse dependence on $k$, which was observed in the experiments. Indeed, for large $N$, as soon as $k$ becomes larger than $O(\sqrt{\log N})$, which is often the case in practice, then CSC becomes much faster than SC.

Be that as it may, one should keep in mind that a central assumption of any spectral clustering algorithm is that the indicator vectors of the clusters live close to the span of the first $k$ eigenvectors of $L$. This implies one has created a graph that reflects this property, which is the central theme of these report.

## V. Research Proposal

Despite coming from different perspectives, Problems 3 and 9 have very similar signal models. Indeed, if our data $X$ is centralized, one can view Problem 9 as a restricted MAP problem on Gaussian variables with covariance matrix $(L+\delta I)^{-1}$, for some $\delta > 0$. This comes from the circularity of the trace: $tr(X^T L X) = tr(X X^T L) \propto tr(SL)$, where $S$ is the sample covariance matrix, as in Section II. The $\log\det$ term in Problem 3 enforces the recovered matrix to be invertible, so Problem 9 is a restricted MAP problem because in forcing $L$ to be a proper Laplacian, one implies $(L+\delta I)$ is invertible, but the converse doesn't hold. The main point is that one can interpret the smoothness assumption in Kalofolias' work as Gaussianity of the data.

To allow for more flexible signal models, while maintaining the nice connectivity priors in Kalofolias' work, a possible research direction is in using the following problem for graph learning:

$$\min_{W,Y} \quad \|X - g(L)Y\|_F^2 + \gamma \|Y\|_{1,1} - \alpha \mathbf{1}^T \log(W\mathbf{1}) + \beta \|W\|_F^2. \tag{17}$$

Problem 17 is based on ideas from sparse coding, where we search for a simple representation $Y$ of the data with help from a dictionary $g(L)$. Here, simplicity is translated as sparsity in its surrogate form, the $\ell_1$-norm on $Y$. The dictionary is written as a function of the Laplacian to make explicit the fact that we are learning a graph and from this graph we derive a dictionary for the representation of the data. The reasoning behind this formulation is that regardless of the signal model, one should be able to find a simple representation for it. As somewhat trivial example, in the spectral clustering setting the dictionary could consist on the stacking of the indicator vectors of the clusters.

Another possible direction is to borrow from ideas in Pattern Theory [18]. One of the main aims in this field formulated by Ulf Grenander and more recently championed by David Mumford is that one should be able to sample from the statistical models used to analyze the signals. The motivation is for one to be able to compare the signals in real world with the signals our models are solving for. Going back to the work of Banerjee *et al.*, they claim gene expression data is normally distributed, but they don't sample example gene expression data from their graphical model to confirm this belief.

In general lines, we would have a Gibbs Random Field model for our data which, by the Hammersley-Clifford theorem, is always the case if we assume the distribution satisfies the Markov properties on the undirected graphical representation we are trying to learn for our data. Then, we would solve a MAP problem for a set of given data and learn its graph. In a third step, we would use our probabilistic model to generate a sample signal and assess if it corresponds to the kind of signals we received as input. If not, we can update the model to learn a new graph that hopefully generates better samples.

The most interesting aspect of this scheme, is that it's the data itself that decides if the problem is successful in modeling it or not. Moreover, we can use this feedback to devise ways to update the model to improve its accuracy. However, there are obstacles to be overcome in this framework. For one, we should decide on the family of energy functions to use in the Gibbs measure. Other problems are choosing the comparison functions and the update procedures to guarantee convergence of the learning procedure. All of these might turn out to be application-specific, but the results of the use of this framework in speech recognition, reproducing the statistics of natural images, and the modeling of shapes [18], serve to motivate its translation to the context of graph learning.

## References


[1] O. Chapelle, B. Schölkopf, and A. Zien, *Semi-supervised learning*. Cambridge, MA: MIT Press, 2006.

[2] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Processing Magazine*, vol. 30, no. 3, pp. 83–98, Apr. 2013.

[3] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press, 2009.

[4] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Computation*, vol. 15, no. 6, pp. 1373–1396, Jun. 2003.

[5] O. Banerjee, L. El Ghaoui, and A. dAspremont, "Model Selection Through Sparse Maximum Likelihood Estimation for Multivariate Gaussian or Binary Data." *Journal of Machine Learning Research*, vol. 9, pp. 485–516, 2008.
[6] V. Kalofolias, "How to learn a graph from smooth signals," in *th International Conference on Artificial Intelligence and Statistics AISTATS*, Cadiz, Spain, 2016.
[7] Y. Nesterov, "Smooth minimization of non-smooth functions," *Mathematical Programming*, vol. 103, no. 1, pp. 127–152, May 2005.
[8] J. Friedman, T. Hastie, and R. Tibshirani, "Sparse inverse covariance estimation with the graphical lasso," *Biostatistics*, vol. 9, no. 3, pp. 432–441, Jul. 2008.
[9] J. Friedman, T. Hastie, H. Hfling, and R. Tibshirani, "Pathwise coordinate optimization," *The Annals of Applied Statistics*, vol. 1, no. 2, pp. 302–332, 2007.
[10] X. Dong, D. Thanou, P. Frossard, and P. Vandergheynst, "Learning Laplacian Matrix in Smooth Graph Signal Representations," *arxiv.org*, Jun. 2014.
[11] N. Komodakis and J.-C. Pesquet, "Playing with Duality," *IEEE Signal Processing Magazine*, pp. 31–54, Oct. 2015.
[12] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Advances in Neural Information Processing Systems*. MIT Press, 2001, pp. 849–856.
[13] N. Tremblay, G. Puy, R. Gribonval, and P. Vandergheynst, "Compressive Spectral Clustering," *arxiv.org*, Feb. 2016.
[14] Y. Koren, "Drawing Graphs by Eigenvectors," *Computers and Mathematics with Applications*, vol. 49, pp. 1867–1888, 2005.
[15] G. Puy, N. Tremblay, R. Gribonval, and P. Vandergheynst, "Random sampling of bandlimited signals on graphs," *arxiv.org*, pp. 1–24, Nov. 2015.
[16] C. Boutsidis, P. Kambadur, and A. Gittens, "Spectral clustering via the power method - provably," in *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, 2015, pp. 40–48. [Online]. Available: http://jmlr.org/proceedings/papers/v37/boutsidis15.html
[17] J. Yang and J. Leskovec, "Defining and evaluating network communities based on ground-truth," *Knowl. Inf. Syst.*, vol. 42, no. 1, pp. 181–213, 2015. [Online]. Available: http://dx.doi.org/10.1007/s10115-013-0693-z
[18] D. Mumford and A. Desolneux, *Pattern Theory: The Stochastic Analysis of Real-World Signals*, A. K. Peters, Ed. CRC Press, 2010.