Simultaneous Graph Signal Clustering and Graph Learning

Abdullah Karaaslanli [1] **Selin Aviyente** [1]

Abstract

Graph learning (GL) aims to infer the topology of an unknown graph from a set of observations on its nodes, i.e., graph signals. While most of the existing GL approaches focus on homogeneous datasets, in many real world applications, data is heterogeneous, where graph signals are clustered and each cluster is associated with a different graph. In this paper, we address the problem of learning multiple graphs from heterogeneous data by formulating an optimization problem for joint graph signal clustering and graph topology inference. In particular, our approach extends spectral clustering by partitioning the graph signals not only based on their pairwise similarities but also their smoothness with respect to the graphs associated with the clusters. The proposed method also learns the representative graph for each cluster using the smoothness of the graph signals with respect to the graph topology. The resulting optimization problem is solved with an efficient block-coordinate descent algorithm and results on simulated and real data indicate the effectiveness of the proposed method.

1. Introduction

In many modern data science applications, relationships between entities, such as features or data samples, are well described with a graph structure. While many real-world data are intrinsically graph-structured, e.g. social and traffic networks, there is still a large number of applications, where the graph topology is not readily available. For instance, gene regulations in biological applications or neuronal connections in the brain are not known. In these applications, the graphs need to be learned since they reveal the relational structure and may assist in a variety of learning tasks. GL deals with the inference of a topological structure among entities from a set of observations on these entities, i.e., graph signals.

Methodologies to learn a graph from data include naive methods such as k-nearest neighborhood (k-NN), probabilistic graphical models (Banerjee et al., 2008; d'Aspremont et al., 2008; Mazumder & Hastie, 2012; Hsieh et al., 2011) and more recently graph signal processing (GSP) (Mateos et al., 2019; Dong et al., 2019) and graph neural networks (GNNs) (Zhang et al., 2020; Wu et al., 2020; Bronstein et al., 2017). While the probabilistic graphical models assume the normality of the data, which is not true for most real-world data, GSP-based GL methods define observations on a collection of nodes as graph signals and fall into two categories. The first category assumes graph signals are outcomes of diffusion processes on graphs and reconstructs a graph from signals according to the diffusion model (Thanou et al., 2017; Pasdeloup et al., 2017; Shafipour et al., 2021; Segarra et al., 2017). The second category of methods promotes the smoothness of graph signals quantified by the Laplacian quadratic form or more generally via total variation (Dong et al., 2016; Kalofolias, 2016b; Berger et al., 2020). GNN-based methods, on the other hand, typically require a large volume of training data and the learned connectivity is often less explainable compared to probabilistic graph models and GSP methods.

Most of the work on GSP-based GL has focused on the case where all data points follow the same relational model described by a single graph. However, in practice, the data may be coming from multiple graphs, i.e., multiview graphs. Examples of this setup include gene regulation networks where regulations vary across different cell types, and in social networks, where a set of users have varying interactions across different social media platforms. In this paper, we address the problem of multiple graph learning from a heterogeneous set of graph signals, where each cluster is associated with a different graph structure. To this end, we propose GRASCale algorithm for simultaneous GRAph Signal Clustering And graph LEarning. Previous works that perform the same task employ only relations of the graph signals to the graphs associated with the clusters for clustering assignment. However, clustering algorithm can also benefit from side information in the form of pairwise relations between graph signals. For instance, in a recommendation

[1]Department of Electrical and Computer Engineering, Michigan State University, East Lansing, MI, US. Correspondence to: Abdullah Karaaslanli <karaasl1@msu.edu>.

Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

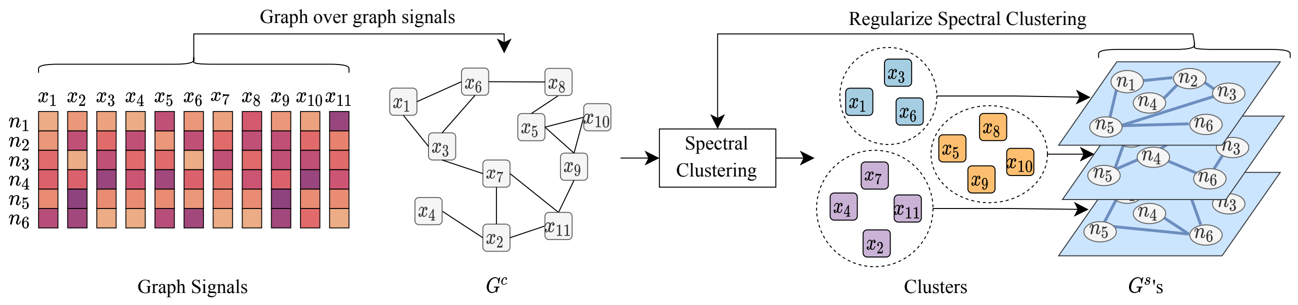
Graph over graph signals
Regularize Spectral Clustering
x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9 x_{10} x_{11}
n_1 n_2 n_3 n_4 n_5 n_6
Spectral Clustering
Graph Signals
G^c
Clusters
G^s's

Figure 1. Overview of the proposed approach: Pairwise similarity between graph signals (G^c) and smoothness of the graph signals with respect to graphs associated with each cluster (G^s's) are used jointly in spectral clustering while simultaneously learning G^s's.

system, when clustering graph signals, e.g. ratings for items, generated by a set of users, connections among the users can be used to inform the clustering algorithm. Therefore, we formulate GRASCale[1] with the following key contributions:

- We propose a new framework which is an extension of conventional spectral clustering where both the signals' pairwise similarity and smoothness with respect to the underlying graph structure are taken into account.
- The proposed methodology can learn the graph structures for mixed (heterogeneous) graph data.
- An efficient prox-linear block-coordinate descent (BCD) with improved consensus clustering based initialization is introduced for optimization.

The overall framework is depicted in Figure 1.

1.1. Related Work

Most of the existing work on GL considers simple data, where all data points follow the same model defined with only one graph. In recent work, GSP community has addressed the problem of learning multiple graphs from heterogenous data in two different settings: i) multiple views of the same data and ii) heterogenous data with possibly unknown cluster information.

The first class of methods, also known as joint inference of multiple graphs (Navarro et al., 2020), considers the setting where multiple related networks each with a subset of observations is available. In this setting, the membership of the signals to the graphs is known and the graphs are closely related to each other. This problem setting has been most widely studied for inferring the topology of time-varying networks (Kalofolias et al., 2017; Yamada et al., 2019; Baingana & Giannakis, 2016; Sardellitti et al., 2019). Assuming that the variation is smooth across time, the problem is reduced to learning multiple closely related graphs regularized with a term that promotes changes between consecutive graphs to be small in some pre-specified norm. More recently, the problem of joint inference of multiple graphs from the observed graph signals has been formulated with the assumption of graph stationarity (Navarro et al., 2020). In this formulation, the signals are assumed to be stationary, and pairwise similarity between all graphs is used to regularize the optimization.

The second class of methods focuses on the case where the data is heterogeneous and each subgroup has its own graph structure. This problem has been addressed for both the supervised and unsupervised settings. The supervised setting, also known as multi-category GL problem, assumes that the number of classes and the signals that belong to each class are known *a priori* (Saboksayr et al., 2021; Kao et al., 2017). In this case, the goal is to learn multiple graphs each associated to a class of signals such that the representation of signals within a class and discrimination of signals in different classes are both taken into consideration. In the unsupervised setting, the number of clusters is known but the membership of the different graph signals is not known. In this case, the goal is to simultaneously cluster the data and learn the representative graph for each cluster (Araghi et al., 2019; Maretic & Frossard, 2020). In (Maretic & Frossard, 2020), graph signals are modelled by a graph Laplacian mixture model (GLMM), which extends the factor analysis model of (Dong et al., 2016) to jointly model the smooth graph signals and identify the clusters through Gaussian mixture model (GMM). This model assumes that the number of clusters is known *a priori* and the distribution of the data is Gaussian. The model is fitted to data through the expectation-maximization algorithm for simultaneous graph inference and clustering. On the other hand, (Araghi et al., 2019) proposes K-graphs, which is an extension of k-means clustering where the graph signals are assigned to the clusters based on their smoothness over each cluster's representative graph. Once the signals are clustered, the representative graphs are updated with graph learning algorithms. Both of GLMM and K-Graphs algorithms assign a graph signal to a cluster based on only the smoothness of the signal with respect to the graph associated with that cluster and do not explicitly take the pairwise relationships

[1]Codes are available at the following github repository: https://github.com/SPLab-aviyente/GRASCale

between the graph signals into account.

2. Background

2.1. Notations

Scalars, vectors and matrices are denoted by lowercase (x), bold lowercase ($\mathbf{x}$) and bold uppercase ($\mathbf{X}$) letters, respectively. Entries of vectors and matrices are denoted as x_i and X_{ij}, respectively. ith row and column of $\mathbf{X}$ are indicated as $\mathbf{X}_{i\cdot}$ and $\mathbf{X}_{\cdot i}$, respectively. Superscript $^\top$ indicates transpose of vectors and matrices. Identity matrix is shown by $\mathbf{I}$. All ones and zeroes vectors and matrices are shown as $\mathbf{1}$ and $\mathbf{0}$, respectively. The operator $\mathrm{dg}()$ takes a matrix $\mathbf{X}$ and returns a vector $\mathbf{x}$ with $x_i = X_{ii}$ or takes a vector $\mathbf{x}$ and returns a diagonal matrix $\mathbf{X}$ with $X_{ii} = x_i$. Finally, δ_{ij} is Kronecker delta, which is equal to 1 if $i = j$ and 0, otherwise.

2.2. Graphs and Graph Signals

An undirected graph is denoted by $G = (V, E, \mathbf{W})$ where V is the node set with $|V| = n$, $E \subseteq V \times V$ is the edge set and $\mathbf{W} \in \mathbb{R}^{n\times n}$ is its adjacency matrix. An edge between node v_i and v_j is represented by e_{ij} and is associated with a weight w_{ij}. $\mathbf{W}$ is a symmetric matrix with $W_{ij} = w_{ij}$ if $e_{ij} \in E$ and 0, otherwise. Degree of v_i is the sum of the weights of the edges connected to it, i.e., $d_i = \mathbf{W}_{i\cdot}^\top \mathbf{1}$. Degree vector of G is $\mathbf{d} = \mathbf{W}\mathbf{1}$ and $\mathbf{D} = \mathrm{dg}(\mathbf{d})$ is the corresponding degree matrix. The Laplacian matrix of G is defined as $\mathbf{L} = \mathbf{D} - \mathbf{W}$. $\mathbf{L}$ is a positive semi-definite matrix and has eigendecomposition $\mathbf{L} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top$ where $\mathbf{\Lambda}$ is the diagonal matrix of eigenvalues and columns of $\mathbf{V}$ are the eigenvectors. Eigenvalues of $\mathbf{L}$ are assumed to be sorted in an ascending order, i.e., $0 = \Lambda_{11} \leq \Lambda_{22} \leq \cdots \leq \Lambda_{nn}$.

A graph signal $\mathbf{x} \in \mathbb{R}^n$ is a vector whose entries reside on the nodes of a graph G. Graph Fourier transform (GFT) of $\mathbf{x}$ is defined as the expansion of $\mathbf{x}$ in terms of the eigenbasis of the graph Laplacian (Shuman et al., 2013). This representation allows us to characterize $\mathbf{x}$ in terms of its graph spectral content. GFT of $\mathbf{x}$ is given by $\widehat{\mathbf{x}} = \mathbf{V}^\top \mathbf{x}$. Inverse GFT is defined as (Shuman et al., 2013):

$$\mathbf{x} = \mathbf{V}\widehat{\mathbf{x}} = \sum_{i=1}^{n} \widehat{x}_i \mathbf{V}_{\cdot i}. \tag{1}$$

Thus, $\mathbf{x}$ is the linear combination of eigenvectors of $\mathbf{L}$ with the coefficients equal to the entries of $\widehat{\mathbf{x}}$. Eigenvectors of $\mathbf{L}$ corresponding to the small eigenvalues have small variation over the graph. Thus, if most of the energy of $\widehat{\mathbf{x}}$ lies in $\widehat{x}_i$s corresponding to the small eigenvalues, then $\mathbf{x}$ is smooth with respect to the graph (Sandryhaila & Moura, 2014). The smoothness of $\mathbf{x}$ over G is then quantified as (Shuman et al., 2013):

$$\mathrm{tr}(\widehat{\mathbf{x}}^\top \mathbf{\Lambda} \widehat{\mathbf{x}}) = \mathrm{tr}(\mathbf{x}^\top \mathbf{V}\mathbf{L}\mathbf{V}^\top \mathbf{x}) = \mathrm{tr}(\mathbf{x}^\top \mathbf{L}\mathbf{x}), \tag{2}$$

which is small if $\mathbf{x}$ is smooth over G.

2.3. Spectral Clustering

Data clustering can be formulated as community detection on graphs, where the nodes correspond to data points and the edge weights indicate the pairwise similarity between data points. The problem of data clustering is then transformed to partitioning the nodes into groups with strong intra- and weak inter-group connectivity. Given a dataset $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^p$ with an associated graph $G = (V, E, \mathbf{W})$, where $|V| = p$ and v_i corresponds to $\mathbf{x}_i$ and w_{ij} indicates the similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$, clustering is then formulated as the partitioning of V into k communities, i.e., $\mathcal{C} = \{C_1, \ldots C_k\}$. Given the cluster assignment vector $\mathbf{g} \in \mathbb{R}^p$ with $g_i = s$ if $v_i \in C_s$, the graph cut is then defined as (von Luxburg, 2007):

$$\mathrm{cut}(\mathcal{C}) = \sum_{i,j=1}^{p} W_{ij}(1 - \delta_{g_i g_j}) = \mathrm{tr}(\mathbf{Z}^\top \mathbf{L}\mathbf{Z}), \tag{3}$$

where $\mathbf{Z} \in \mathbb{R}^{p\times k}$ is the cluster assignment matrix with $Z_{is} = 1$ if $v_i \in C_s$ and 0, otherwise. The graph cut is the sum of the inter-community edge weights and $\mathcal{C}$ can be found by minimizing the graph cut. This minimization is NP-hard (von Luxburg, 2007). Thus, $\mathbf{Z}$ is relaxed to take on real values, which leads to the following optimization problem:

$$\underset{\mathbf{Z}}{\mathrm{minimize}}\ \mathrm{tr}(\mathbf{Z}^\top \mathbf{L}\mathbf{Z}) \ \ \mathrm{s.t.}\ \mathbf{Z} \in \mathbb{D}, \tag{4}$$

where $\mathbf{Z}$ is constrained to be in a set $\mathbb{D}$ to ensure that $\mathbf{Z}$ preserves some properties of binary cluster membership matrix. These properties can include positivity ($\mathbf{Z} \geq 0$), row-sum constraint ($\mathbf{Z}\mathbf{1} = \mathbf{1}$) or orthogonality ($\mathbf{Z}^\top\mathbf{Z} = \mathbf{I}$) (von Luxburg, 2007; Shewchuk, 2016; Zass & Shashua, 2005). Once a real valued $\mathbf{Z}$ is learned, clustering algorithms such as k-means can be employed to identify the clusters.

2.4. Graph Learning

An unknown graph G can be learned from a set of observed graph signals defined over it with the assumption that graph signals are smooth over G. Using this assumption, (Dong et al., 2016) proposed to learn G by minimizing (2) with respect to $\mathbf{L}$ given a set of graph signals $\{\mathbf{x}_i\}_{i=1}^p$ as follows:

$$\underset{\mathbf{L}\in\mathbb{L}}{\mathrm{minimize}}\ \ \mathrm{tr}(\mathbf{X}^\top \mathbf{L}\mathbf{X}) + \alpha\|\mathbf{L}\|_F^2 \ \ \mathrm{s.t.}\ \mathrm{tr}(\mathbf{L}) = 2n, \tag{5}$$

where $\mathbf{X} \in \mathbb{R}^{n\times p}$ is the data matrix whose columns are $\mathbf{x}_i$'s and $\mathbb{L} = \{\mathbf{L} : L_{ij} = L_{ji} \leq 0\ \forall i \neq j,\ \mathbf{L}\mathbf{1} = \mathbf{0}\}$ is the set of valid Laplacian matrices. The first term in (5) measures the smoothness of the graph signals while the second term is the Frobenius norm of $\mathbf{L}$ and controls the density of the learned graph. Finally, the problem is constrained to prevent the trivial solution $\mathbf{L} = \mathbf{0}$.

3. Method

3.1. Graph Signal Clustering with Regularized Graph Cut

Assume we are given a dataset $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^p$ where $\mathbf{x}_i \in \mathbb{R}^n$ is a graph signal over a graph $G^s \in \mathcal{G} = \{G^1, \ldots, G^k\}$. All graphs in $\mathcal{G}$ are defined over the same vertex set V with $|V| = n$ and have their own edge set E^s, i.e., $G^s = (V, E^s, \mathbf{W}^s)$, $\forall G^s \in \mathcal{G}$. Let the partitioning of graph signals in $\mathcal{X}$ be defined as $\mathcal{C} = \{C_1, \ldots, C_k\}$ where C_s includes all of the graph signals defined over G^s. In this paper, it is assumed that the partitioning of the graph signals, $\mathcal{C}$, is not known *a priori*. The problem of learning $\mathcal{C}$ can be considered as a clustering problem. Let $G^c = (V^c, E^c, \mathbf{W}^c)$ be the graph that represents the similarity between the elements of $\mathcal{X}$ where V^c is the node set with $|V^c| = p$. Node $v_i^c \in V^c$ corresponds to $\mathbf{x}_i$ and w_{ij}^c is the similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$. $\mathcal{C}$ can then be learned by applying spectral clustering to G^c. However, spectral clustering as formulated in (4) does not use the fact that $\mathbf{x}_i$'s are graph signals. One can improve the clustering by incorporating information from the graphs in $\mathcal{G}$. Therefore, we propose a regularized graph cut (regcut) by assuming that the graph signals are smooth over the graphs they are defined on:

$$\text{regcut}(\mathcal{C}) = \sum_{i,j=1}^{p} W_{ij}^c (1\text{-}\delta_{g_i g_j}) + \alpha \sum_{s=1}^{K} \sum_{i=1}^{p} \delta_{g_i s} \mathbf{x}_i^\top \mathbf{L}^s \mathbf{x}_i, \quad (6)$$

where $\mathbf{x}_i^\top \mathbf{L}^s \mathbf{x}_i$ is the smoothness of $\mathbf{x}_i$ over G^s as defined in Section 2.2. By regularizing the graph cut with smoothness, we ensure that if $\mathbf{x}_i$ is assigned to the sth cluster it is smooth with respect to G^s. As in Section 2.3, this problem is NP-hard. Therefore, we relax $\mathbf{Z}$ to take on real values and obtain the following optimization problem:

$$\underset{\mathbf{Z} \in \mathbb{D}}{\text{minimize}} \; \text{tr}(\mathbf{Z}^\top \mathbf{L}^c \mathbf{Z}) + \alpha \sum_{s=1}^{k} \text{tr}(\text{dg}(\mathbf{Z}_{\cdot s}) \mathbf{X}^\top \mathbf{L}^s \mathbf{X}), \quad (7)$$

where $\mathbf{X}$ is the data matrix with $\mathbf{X}_{\cdot i} = \mathbf{x}_i$ and $\mathbf{Z}$ is constrained as in (4).

3.2. Joint Graph Signal Clustering and Graph Learning

For the optimization problem in (7), one needs to know G^c and the graphs in $\mathcal{G}$. Since these graphs are generally not available, they need to be learned. G^c can be learned from $\mathbf{X}$ using the aforementioned GL methods or more classical approaches such as k-nearest neighbor graphs. However, for graphs in $\mathcal{G}$, we cannot use these approaches as we do not know the partitioning of the graph signals. Thus, the graphs in $\mathcal{G}$ must be learned simultaneously with clustering. Therefore, we extend (7) with GL:

$$\begin{aligned} \underset{\mathbf{Z}, \mathbf{L}^1, \ldots, \mathbf{L}^k}{\text{minimize}} \; & \text{tr}(\mathbf{Z}^\top \mathbf{L}^c \mathbf{Z}) + \alpha_1 \sum_{s=1}^{k} \Big[\text{tr}(\text{dg}(\mathbf{Z}_{\cdot s}) \mathbf{X}^\top \mathbf{L}^s \mathbf{X}) \\ & + (\mathbf{Z}_{\cdot s}^\top \mathbf{1}) \alpha_2 \|\mathbf{L}^s\|_F^2 \Big] \qquad (8) \\ \text{s.t.} \quad & \mathbf{Z} \in \mathbb{D},\ \mathbf{L}^s \in \mathbb{L},\ \text{tr}(\mathbf{L}^s) = 2n \; \forall s \in \{1, \ldots, k\}, \end{aligned}$$

where each $\mathbf{L}^s$ is learned by assuming that graph signals in the sth cluster are smooth over G^s. As in (5), the Frobenius norm controls the sparsity of the learned graphs such that large values of α_2 result in denser graphs. However, in this setting we weigh this sparsity term with $\mathbf{Z}_{\cdot s}^\top \mathbf{1}$ which corresponds to the number of signals in cluster s to ensure that the sparsity levels of the learned graphs are similar for a given α_2. As the value of the smoothness term increases with the number of signals in the cluster, multiplying the sparsity term with $\mathbf{Z}_{\cdot s}^\top \mathbf{1}$ ensures that the relative importance of the sparsity term with respect to smoothness term remains similar across s. Finally, we set $\mathbb{D} = \{\mathbf{Z} \in \mathbb{R}^{p \times k} \mid \mathbf{Z} \geq 0, \mathbf{Z1} = \mathbf{1}\}$.

3.3. Optimization

The problem in (8) is a multi-convex problem, i.e., it is convex in each variable separately but non-convex when all variables are considered together. Therefore, we employ block coordinate descent (BCD) to solve (8) (Shi et al., 2017). At each iteration of BCD, the problem is solved cyclically over each variable while fixing the remaining variables. When solving with respect to a variable, we perform inexact minimization with prox-linear update as it results in easy-to-solve problems with fast convergence when extrapolation is used (Xu & Yin, 2013). Before applying BCD, we first vectorize (8) where we learn the upper triangular part of $\mathbf{L}^s$. Let $\boldsymbol{\ell}^s \in \mathbb{R}^m$ be the upper triangular part of $\mathbf{L}^s$ where $m = n(n-1)/2$. Define the operator mt with $\text{mt}(\boldsymbol{\ell}^s) = \mathbf{L}^s$ and define the matrix $\mathbf{P} \in \mathbb{R}^{m \times n}$ with $\mathbf{P}\boldsymbol{\ell}^s = -\text{dg}(\mathbf{L}^s)$. Then, (8) can be rewritten as:

$$\begin{aligned} \underset{\mathbf{Z}, \mathbf{L}^1, \ldots, \mathbf{L}^k}{\text{minimize}} \; & \text{tr}(\mathbf{Z}^\top \mathbf{L}^c \mathbf{Z}) + \alpha_1 \sum_{s=1}^{k} \Big[\text{tr}(\text{dg}(\mathbf{Z}_{\cdot s}) \mathbf{X}^\top \text{mt}(\boldsymbol{\ell}^s) \mathbf{X}) \\ & + (\mathbf{Z}_{\cdot s}^\top \mathbf{1}) \alpha_2 (2\langle \boldsymbol{\ell}^s, \boldsymbol{\ell}^s \rangle + \langle \mathbf{P}\boldsymbol{\ell}^s, \mathbf{P}\boldsymbol{\ell}^s \rangle) \Big] \qquad (9) \\ \text{s.t.} \quad & \mathbf{Z} \geq 0, \mathbf{Z1} = \mathbf{1}, \boldsymbol{\ell}^s \leq 0, \mathbf{1}^\top \boldsymbol{\ell}^s = -n \; \forall s, \end{aligned}$$

where $\langle \cdot, \cdot \rangle$ is the inner product. Prox-linear updates at the tth iteration of BCD can then be found as follows:

$$\mathbf{Z}^{(t+1)} = \underset{\substack{\mathbf{Z} \geq 0, \\ \mathbf{Z1} = \mathbf{1}}}{\text{argmin}} \; \langle \widehat{\mathbf{G}}_Z^{(t)}, \mathbf{Z} \text{-} \widehat{\mathbf{Z}}^{(t)} \rangle + \frac{\lambda_Z}{2} \|\mathbf{Z} \text{-} \widehat{\mathbf{Z}}^{(t)}\|_F^2, \quad (10)$$

$$\boldsymbol{\ell}^{s(t+1)} = \underset{\substack{\boldsymbol{\ell}^s \leq 0, \\ \mathbf{1}^\top \boldsymbol{\ell}^s = -n}}{\text{argmin}} \; \langle \widehat{\mathbf{g}}_s^{(t)}, \boldsymbol{\ell}^s \text{-} \widehat{\boldsymbol{\ell}^s}^{(t)} \rangle + \frac{\lambda_s}{2} \|\boldsymbol{\ell}^s \text{-} \widehat{\boldsymbol{\ell}^s}^{(t)}\|_F^2, \quad (11)$$

Algorithm 1 GS Clustering with Simultaneous GL

```
Input: $\mathbf{X}$, $\mathbf{L}^s$, $\alpha_1$, $\alpha_2$, $k$ and max_iter
Set $t \leftarrow 1$
Initialize $\mathbf{Z}^{(t)}$, $\mathbf{Z}^{(t-1)}$, $\boldsymbol{\ell}^{s\,(t)}$ and $\boldsymbol{\ell}^{s\,(t-1)}$
repeat
    Update $\mathbf{L}^{s\,(t+1)}$ with (11) for $s \in \{1, \dots, k\}$
    Update $\mathbf{Z}^{(t+1)}$ with (10)
    Set $t \leftarrow t + 1$
until convergence or $t \geq$ max_iter
Output: $\mathbf{Z}^{(t)}$, $\mathbf{L}^{1\,(t)}, \dots, \mathbf{L}^{k\,(t)}$
```

where $\widehat{\mathbf{G}}_Z^{(t)}$ is the gradient of the objective function in (9) with respect to $\mathbf{Z}$ evaluated at $\widehat{\mathbf{Z}}^{(t)}$, $\widehat{\mathbf{g}}_s^{(t)}$ is the gradient with respect to $\boldsymbol{\ell}^s$ evaluated at $\widehat{\boldsymbol{\ell}^s}^{(t)}$, and:

$$\widehat{\mathbf{Z}}^{(t)} = \mathbf{Z}^{(t-1)} + w(\mathbf{Z}^{(t-1)} - \mathbf{Z}^{(t-2)}), \tag{12}$$

$$\widehat{\boldsymbol{\ell}^s}^{(t)} = \boldsymbol{\ell}^{s\,(t-1)} + w(\boldsymbol{\ell}^{s\,(t-1)} - \boldsymbol{\ell}^{s\,(t-2)}), \tag{13}$$

where $0 \leq w \leq 1$ is the extrapolation parameter. Finally, λ_Z and λ_s are step sizes and can be set to the Lipschitz constants of the gradient of the objective function in (9) with respect to $\mathbf{Z}$ and $\boldsymbol{\ell}^s$. Solutions of both (10) and (11) are projections onto simplex, as shown in the Appendix. Overall optimization procedure is given in Algorithm 1.

(Xu & Yin, 2013) show that BCD with prox-linear update converges for multi-convex problems, when the objective function consists of smooth and separable non-smooth terms. The problem in (9) satisfies these assumptions; thus, Algorithm 1 is guaranteed to converge.

3.4. Initialization

BCD type algorithms may converge to poor local minima (Shi et al., 2017). To overcome this problem, one can run the algorithm multiple times and consider the solution with the smallest objective value. One can also initiate the algorithm at a better point such that it converges to a solution with lower objective value. In this section, we describe a procedure to select better initializations for the proposed BCD algorithm.

Consider the set $\mathcal{Z} = \{\mathbf{Z}^1, \dots, \mathbf{Z}^b\}$ which is obtained by running Algorithm 1 b times. Each $\mathbf{Z}^i$ indicates a possible partitioning of the graph signals. One can obtain a better clustering by combining information from all Z^i's using consensus clustering (Strehl & Ghosh, 2002), an ensemble learning method to combine multiple clusterings. We follow the consensus clustering procedure described in (Lancichinetti & Fortunato, 2012), where the consensus clustering $\mathbf{Z}^0$ is found from an association matrix $\mathbf{A}$ whose entries A_{ij} are equal to the number of times graph signals $\mathbf{x}_i$ and $\mathbf{x}_j$ are assigned to the same cluster in $\mathcal{Z}$. This association matrix can be used as the input to spectral clustering to find $\mathbf{Z}^0$. Once $\mathbf{Z}^0$ is found, we rerun the Algorithm 1 one more time, where $\mathbf{Z}$ is initialized at $\mathbf{Z}^0$ (the rest of the variables are initialized randomly). The clustering and learned graphs obtained from this run are used as the final result. This initialization procedure is given in Algorithm 2.

Algorithm 2 Initialization Procedure

```
Input: $b$
Initialize $\mathcal{Z}$ as an empty set
for $i \leq b$ do
    Run Algorithm 1 and add learned $\mathbf{Z}$ to $\mathcal{Z}$
end for
Find $\mathbf{Z}^0$ by applying consensus clustering to $\mathcal{Z}$
Run Algorithm 1 with initial point set to $\mathbf{Z}^0$
Output: Solutions of the last run
```

In our experiments, we set $b = 9$ and we set the maximum number of iterations for each run to a small number, e.g., 100, since even sub-optimal solutions can result in a good consensus clustering.

3.5. Hyperparameter Selection

The proposed method requires the selection of three hyperparameters: number of clusters k, α_1 and α_2. In literature, various methods have been proposed to determine the number of clusters in spectral clustering. These methods generally define a quality function Q and find the number of clusters as the value that optimizes Q. Possible choices of Q are eigengap (Von Luxburg, 2007), modularity (Newman, 2006), Bayesian information criterion (BIC) (Saldana et al., 2017), integrated completed likelihood (ICL) (Daudin et al., 2008). α_2 controls the sparsity level of the learned graphs such that larger values of α_2 result in denser graphs. We set it to a value that results in graphs with a pre-determined sparsity level. This approach is similar to previous graph construction schemes, such as in k-NN graphs, where one wants to construct a graph with each node having at least k neighbors. The selection of α_1 is explained in detail through parameter sensitivity analysis in Section 4.1

4. Results

In this section, the performance of GRASCale is evaluated on synthetic and real datasets and is compared to various state-of-the-art clustering and graph learning algorithms. We compare methods based on the quality of the resulting clustering as well as the accuracy of the learned graphs associated with each cluster. For the first comparison, we consider normalized spectral clustering (SC), GLMM and K-Graphs. For the latter comparison, GL (see Section 2.4), GLMM and K-Graphs are considered. As mentioned in Section 2.3, SC clusters signals only based on their pairwise similarities. Thus, by comparing GRASCale to SC, we can illustrate the benefits of considering graph signal smoothness. GLMM and K-Graphs perform simultaneous graph signal clustering and graph learning similar to the proposed method. However, they only rely on the smoothness of the signals with respect to graphs associated with each cluster. By comparing GRASCale against them, we can illustrate the benefits of incorporating pairwise similarities. Finally, when applying GL, we assume the partitioning of the signals is known; thus the performance of GL provides an upper bound for the performance of GRASCale in the graph learning task. We used the formulation of (Kalofolias, 2016a) for implementing GL.

Parameter Selection: SC, GLMM, K-Graphs and GRASCale require the number of clusters k as an input. We provided the ground truth k as an input to all methods. GL, GLMM and K-graphs require a hyperparameter that controls the sparsity of the learned graphs similar to α_2 in (8). For all methods, we set this hyperparameter to a value that results in graphs with sparsity levels between 0.1 and 0.15[2]. GLMM and K-Graphs algorithms are based on alternating minimization, which causes their results to vary across runs. Therefore, we run each algorithm 10 times and report the average performance. For GRASCale, we set $b = 9$ as mentioned in Section 3.5. Thus, each algorithm is run 10 times. Finally, SC is applied to a binary k-nearest neighbor graph with the number of neighbors set to 5. The same graph is used as $\mathbf{L}^c$ for the proposed method.

Performance Metrics: Normalized mutual information (NMI) (Danon et al., 2005) is used to quantify the performance of clustering. For the graph learning task, F1-score is used to quantify how close the learned graphs are to the ground truth graphs. We measure F1-score for all s and report the average.

4.1. Synthetic Data

Data Generation: Given a graph G with Laplacian $\mathbf{L} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^\top$, we can generate a graph signal $\mathbf{x}$ that is smooth with respect to G by filtering a given signal $\mathbf{x}_0$ with a low-pass graph filter (Dong et al., 2016; Kalofolias, 2016a). Mathematically, this is equivalent to $\mathbf{x} = h(\mathbf{L})\mathbf{x}_0$ where $h(\mathbf{L}) = \mathbf{V}h(\boldsymbol{\Lambda})\mathbf{V}^\top$ is a low-pass graph filter. Based on this, we generate the synthetic data as follows. We first generate k graphs $\mathcal{G} = \{G^1, \dots, G^k\}$ based on a random graph model, such as Erdős–Rényi (ER) (Gilbert, 1959) or Barabási–Albert (BA) models (Albert & Barabási, 2002), where each G^s has n nodes. For each G^s, we generate p_s smooth graph signals as described above with $h(\boldsymbol{\Lambda}) = \sqrt{\boldsymbol{\Lambda}}^\dagger$ and $\mathbf{x}_0 \sim \mathcal{P}$, where † is the pseudo-inverse operator and $\mathcal{P}$ is a probability distribution to be determined. The graph signals are then used to construct data matrices $\mathbf{X}^s \in \mathbf{R}^{n \times p_s}$, from which we build $\mathbf{X} = [\mathbf{X}^1, \dots, \mathbf{X}^s] \in \mathbb{R}^{n\times p}$ where $p = p_1 + \dots + p_s$. White Gaussian noise with variance equal to 10% of the signal power is added to the data matrix. Finally, we generate 20 different realizations of each dataset in all experiments and report the average performance across realizations.

[2]Real-world graphs are generally sparse, so it is desirable to learn sparse graphs. Therefore, we learn graphs at this range of sparsity level. In our experiments, we observed smaller values sparsity level can result in disconnected graphs. To prevent this, we did not consider smaller values.

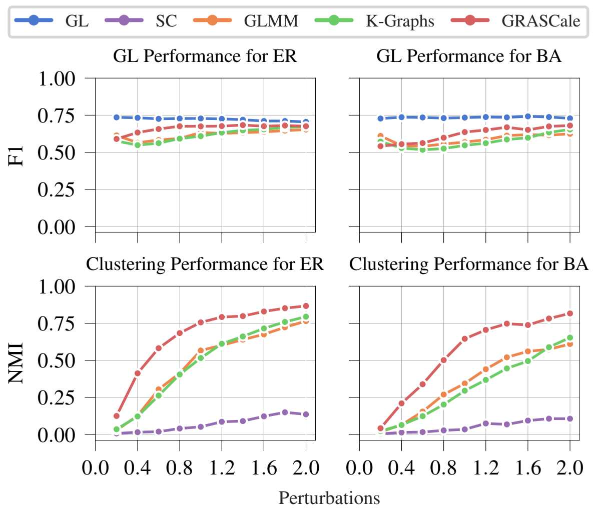

Figure 2. Results for Experiment 1 when cluster sizes are equal. Upper row illustrates the graph learning performance and the bottom row shows the clustering performance. Left and right columns are performances for ER and BA graph models, respectively.

Experiment 1: In this experiment, we generate signals from $\mathcal{G} = \{G^1, G^2, G^3\}$ where each G^s is generated by swapping the edges of a given graph G $\lceil m_G \times pert \rceil$ times. m_G is the number of edges in G and $pert > 0$ refers to the amount of perturbation. Smaller values of $pert$ causes graphs in $\mathcal{G}$ to be highly correlated; thus, clustering the graph signals generated from these graphs becomes a harder task. We generated G with 50 nodes from two random graph models: ER with edge probability $p_{ER} = 0.1$ and BA model with $m_{BA} = 3$. We generated $\mathbf{X}$ as described above with $\mathcal{P} = \mathcal{N}(0, \mathbf{I})$.

In Figure 2, we report the results when the cluster sizes are equal, i.e., $p_s = 200$ for all s. It can be observed that the clustering performance for all methods increases with the amount of perturbation. This is due to the fact that as the perturbation level increases, the different clusters become more distinct. GRASCale performs better than GLMM and K-Graphs for both ER and BA models. SC is observed to perform very poorly as the signals are generated independently from each other. Thus, pairwise similarities between signals that are in the same cluster are not strong, resulting in low NMI values for SC. In terms of graph learning, GL

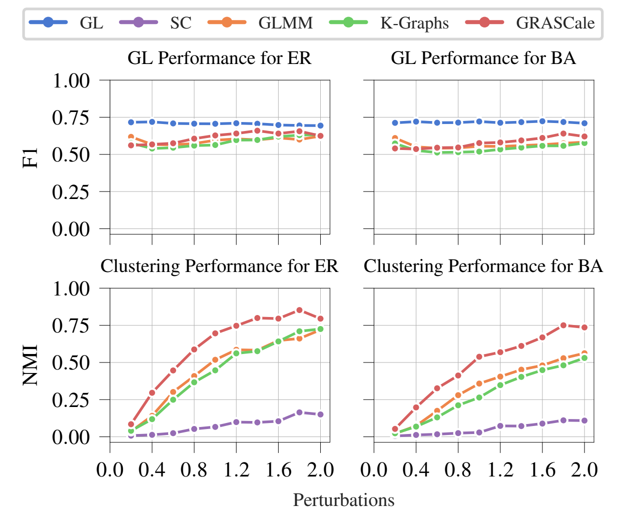

Figure 3. Results for Experiment 1 when cluster sizes are different. Upper row shows graph learning performance and bottom row shows clustering performance. Left and right columns are performances for ER and BA graph models, respectively.

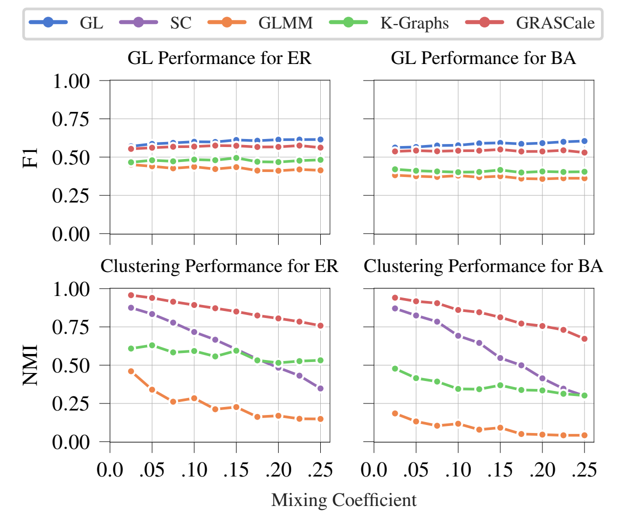

Figure 4. Results for Experiment 2. Upper row shows graph learning performance and bottom row shows clustering performance. Left and right columns are performances for ER and BA graph models, respectively.

performs the best as expected since it assumes that the cluster membership of the signals is known *a priori*. There is a slight improvement in the graph learning performances of GLMM, K-Graphs and GRASCale as perturbation level increases and their performances converge to that of GL. Graph learning performances of GLMM, K-Graphs and the proposed method for small perturbation levels may seem counter-intuitive considering their low NMI values. However, graphs in $\mathcal{G}$ are very correlated for small values of perturbation, thus graph signals in a given cluster carry information about other graphs too. Therefore, methods can still perform well for graph inference even though the graph clusters may not be accurately identified.

Figure 3 illustrates the results for the same simulation setting when there is heterogeneity in cluster sizes, i.e., $p_1 = 300$, $p_2 = 200$, and $p_3 = 100$. Results are very similar to that of Figure 2. There is a slight drop in the performance of all algorithms compared to Figure 2 across all perturbation levels and graph models.

Experiment 2: In the previous experiment, signals were generated independently; thus they do not have any explicitly imposed pairwise relations. In this experiment, we generate graph signals that have pairwise relations and are also smooth with respect to graphs associated with clusters. In order to achieve this goal, we first generate a data matrix $\mathbf{Y} \in \mathbb{R}^{n \times p}$ with $n = 50$ and $p = 600$. Rows of $\mathbf{Y}$ are generated by filtering a signal $\mathbf{y} \in \mathbb{R}^p$ through a low-pass graph filter defined on G^c. The signal similarity graph G^c has p nodes and $\mathbf{y} \sim \mathcal{N}(0, \mathbf{I})$. If there is an edge between nodes v_i and v_j in G^c, columns $\mathbf{Y}_{\cdot i}$ and $\mathbf{Y}_{\cdot j}$ will be similar to each other. We construct G^c from a planted partition model (Condon & Karp, 2001) whose nodes are partitioned into three equal sized clusters: $C_1 = \{v_1, \ldots, v_{200}\}$, $C_2 = \{v_{201}, \ldots, v_{400}\}$, and $C_3 = \{v_{401}, \ldots, v_{600}\}$. Planted partition model has two parameters p_{in} and p_{out}, which determine the intra- and inter-cluster connectivity, respectively. We set $p_{in} = 0.05(1 - \mu)$ and $p_{out} = 0.05\mu$, where $\mu > 0$ is the mixing coefficient. Larger values of μ causes the clusters to be less distinguishable. For the low-pass filter, we used a heat kernel $h(\mathbf{\Lambda}^c) = \exp(-5\mathbf{\Lambda}^c)$ where $\mathbf{\Lambda}^c$ is the eigenvalue matrix corresponding to the Laplacian matrix of G^c (Kalofolias, 2016a). We generated graphs in $\mathcal{G}$ as in the first experiment with $pert$ set to 2. Once $\mathbf{Y}$ and $\mathcal{G}$ are generated, columns of $\mathbf{Y}$ in C_s are filtered by the graph filter corresponding to $G^s \in \mathcal{G}$ for all s to construct $\mathbf{X}$.

Figure 4 shows the performance of the different algorithms. With the introduction of pairwise similarity within clusters, the performance of SC is observed to improve significantly. However, its NMI value is still lower than GRASCale since the latter benefits from both pairwise relations and smoothness of the graph signals. GLMM and KGraphs have lower performance than the proposed method, as these methods employ only smoothness of the graph signals. Increasing the mixing coefficient causes a decrease in the performance of all methods, as larger values of μ result in less distinguishable clusters. The decrease in NMI values for SC and GRASCale with increasing μ values follows a similar trend. This indicates that the proposed method indeed uses the pairwise relations between signals. For the graph learning task, F1 score of the proposed method is higher than those of GLMM and K-Graphs and is very close to that of GL due to its high clustering performance.

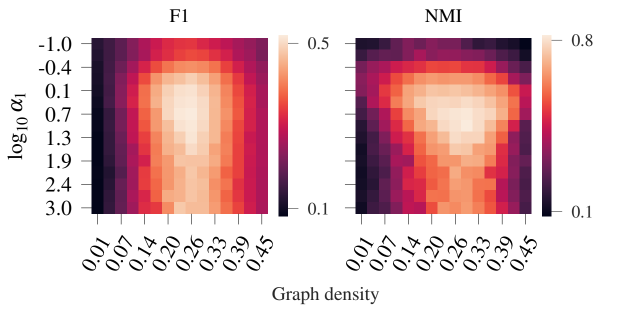

Figure 5. Sensitivity of F1 and NMI values to varying values of α_1 and density of learned graphs. Left panel shows the graph learning performance and the right panel shows the clustering performance.

Parameter Sensitivity: We study the sensitivity of the performance of GRASCale to the selection of α_1 and α_2 on a dataset from Experiment 2. We consider a dataset generated from the BA graph model with μ set to 0.25. The ground truth graph has a density around 0.12 in this dataset. We apply our algorithm to this dataset with varying α_1 and α_2 values and the performances are reported in Figure 5. For the x-axis, densities of the learned graph are used rather than the values of α_2. Figure 5 shows that the density of learned graphs is important for the performance. In particular, low density graphs have poor performance in terms of F1 and NMI, as these graphs are very sparse and do not contain enough information. Similarly, high density values also result in low performance, since learned graphs include many false positive edges. Finally, this figure also shows that the proposed method is not sensitive to the value of α_1 as long as the learned graphs have a reasonable density. In particular, there is a large range of α_1 values, where F1-score and NMI are stable. Based on this observation, we set $\alpha_1 = 10$ in all of our data analysis without any fine-tuning.

4.2. Real Data

In this section, the proposed method is applied to a real world data clustering problem, where the aim is to cluster the digits of MNIST dataset while learning a graph for each digit. More specifically, we selected 400 images corresponding to digits 0, 1, 2 and 3. After vectorizing each image, we obtain a data matrix of size 400×1600, where

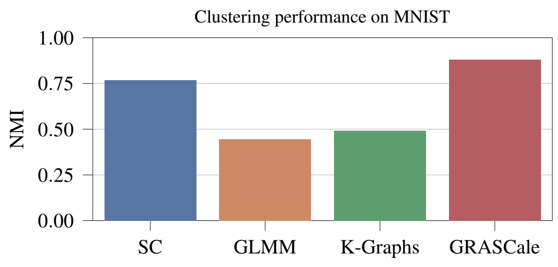

Figure 6. Clustering performance for MNIST dataset.

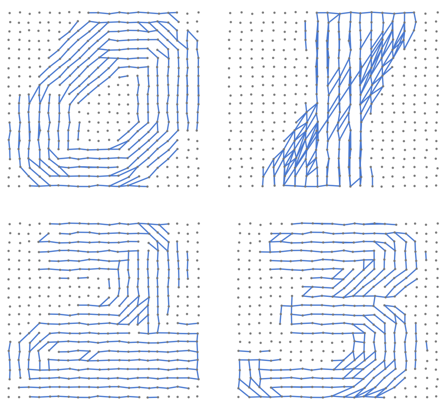

Figure 7. Graph structures learned for each digit by the proposed method. Points correspond to pixels, while lines indicate the inferred edges between pixels. Only top 300 edges are shown.

the rows and columns correspond to pixels and images, respectively. SC, GLMM, K-Graphs and GRASCale are applied to the constructed data matrix and the clustering performance is reported in Figure 6. The best performing method is GRASCale, and it is followed by SC; while GLMM and K-Graphs have significantly lower performance. These results indicate that using pairwise similarities of the signals and their smoothness together improve the clustering performance.

As mentioned in (Maretic & Frossard, 2020), learning a graph for each cluster can be helpful for the interpretablity of clustering. By analyzing the graph structure learned for each cluster, one can deduce why a set of graph signals are assigned to the same cluster; which leads to explainable data science (Roscher et al., 2020). In Figure 7, we plot the graphs learned for each digit by GRASCale. It can be seen that the method learns very interpretable graph structures. The learned graphs for digits 0, 2, 3 have high resemblance to the digits themselves. Although the graph found for digit 1 has a meaningful structure, it is noisier than the other graphs. This is due to the fact that there is a lot of variation across samples for writing digit 1. This means that while we tend to cluster digits based on their numerical values, it might be the case that there is also a clustering within each digit based on the writing style.

5. Conclusions

In this paper, we presented GRASCale for simultaneous graph signal clustering and graph learning. Compared to previous methods developed for the same task, GRASCale uses two types of information: pairwise relations between graph signals and their smoothness with respect to graphs associated with clusters. Our results on synthetic and real datasets indicate that incorporating these complementary pieces of information within the same framework improves clustering and graph learning performance significantly.

In the presented formulation of GRASCale, we assumed $\mathbf{L}^c$ is constructed *a priori*; however, this graph can also be learned along with clusters and graphs associated with clusters. In future work, we will consider this extension of jointly learning $\mathbf{L}^c$ along with the individual graphs, $\mathbf{L}^s$.

Acknowledgements

This work was supported by the National Science Foundation grant: CCF 2006800.

References

Albert, R. and Barabási, A.-L. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47, January 2002. doi: 10.1103/RevModPhys.74.47.

Araghi, H., Sabbaqi, M., and Babaie-Zadeh, M. k-graphs: An algorithm for graph signal clustering and multiple graph learning. *IEEE Signal Processing Letters*, 26(10): 1486–1490, 2019.

Baingana, B. and Giannakis, G. B. Tracking switched dynamic network topologies from information cascades. *IEEE Transactions on Signal Processing*, 65(4):985–997, 2016.

Banerjee, O., El Ghaoui, L., and d'Aspremont, A. Model selection through sparse maximum likelihood estimation for multivariate gaussian or binary data. *The Journal of Machine Learning Research*, 9:485–516, 2008.

Berger, P., Hannak, G., and Matz, G. Efficient graph learning from noisy and incomplete data. *IEEE Transactions on Signal and Information Processing over Networks*, 6: 105–119, 2020.

Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A., and Vandergheynst, P. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4): 18–42, 2017.

Condon, A. and Karp, R. M. Algorithms for graph partitioning on the planted partition model. *Random Structures & Algorithms*, 18(2):116–140, 2001.

Danon, L., Diaz-Guilera, A., Duch, J., and Arenas, A. Comparing community structure identification. *Journal of statistical mechanics: Theory and experiment*, 2005(09): P09008, 2005.

d'Aspremont, A., Banerjee, O., and El Ghaoui, L. First-order methods for sparse covariance selection. *SIAM Journal on Matrix Analysis and Applications*, 30(1):56–66, 2008.

Daudin, J.-J., Picard, F., and Robin, S. A mixture model for random graphs. *Statistics and computing*, 18(2):173–183, 2008.

Dong, X., Thanou, D., Frossard, P., and Vandergheynst, P. Learning Laplacian Matrix in Smooth Graph Signal Representations. *IEEE Transactions on Signal Processing*, 64(23):6160–6173, October 2016. ISSN 1053-587X, 1941-0476. doi: 10.1109/TSP.2016.2602809.

Dong, X., Thanou, D., Rabbat, M., and Frossard, P. Learning graphs from data: A signal representation perspective. *IEEE Signal Processing Magazine*, 36(3):44–63, 2019.

Duchi, J., Shalev-Shwartz, S., Singer, Y., and Chandra, T. Efficient projections onto the l 1-ball for learning in high dimensions. In *Proceedings of the 25th international conference on Machine learning*, pp. 272–279, 2008.

Gilbert, E. N. Random graphs. *The Annals of Mathematical Statistics*, 30(4):1141–1144, 1959.

Hsieh, C.-J., Dhillon, I., Ravikumar, P., and Sustik, M. Sparse inverse covariance matrix estimation using quadratic approximation. *Advances in neural information processing systems*, 24, 2011.

Kalofolias, V. How to learn a graph from smooth signals. In *Artificial Intelligence and Statistics*, pp. 920–929. PMLR, May 2016a.

Kalofolias, V. How to learn a graph from smooth signals. In *Artificial Intelligence and Statistics*, pp. 920–929, 2016b.

Kalofolias, V., Loukas, A., Thanou, D., and Frossard, P. Learning time varying graphs. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2826–2830. Ieee, 2017.

Kao, J.-Y., Tian, D., Mansour, H., Ortega, A., and Vetro, A. Disc-glasso: Discriminative graph learning with sparsity regularization. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2956–2960. IEEE, 2017.

Lancichinetti, A. and Fortunato, S. Consensus clustering in complex networks. *Scientific Reports*, 2(1):336, March 2012. ISSN 2045-2322. doi: 10.1038/srep00336.

Maretic, H. P. and Frossard, P. Graph Laplacian Mixture Model. *IEEE Transactions on Signal and Information Processing over Networks*, 6:261–270, April 2020. ISSN 2373-776X, 2373-7778. doi: 10.1109/TSIPN.2020.2983139.

Mateos, G., Segarra, S., Marques, A. G., and Ribeiro, A. Connecting the dots: Identifying network structure via graph signal processing. *IEEE Signal Processing Magazine*, 36(3):16–43, 2019.

Mazumder, R. and Hastie, T. The graphical lasso: New insights and alternatives. *Electronic journal of statistics*, 6:2125, 2012.

Navarro, M., Wang, Y., Marques, A. G., Uhler, C., and Segarra, S. Joint inference of multiple graphs from matrix polynomials. *arXiv preprint arXiv:2010.08120*, 2020.

Newman, M. E. Modularity and community structure in networks. *Proceedings of the national academy of sciences*, 103(23):8577–8582, 2006.

Pasdeloup, B., Gripon, V., Mercier, G., Pastor, D., and Rabbat, M. G. Characterization and inference of graph diffusion processes from observations of stationary signals. *IEEE transactions on Signal and Information Processing over Networks*, 4(3):481–496, 2017.

Roscher, R., Bohn, B., Duarte, M. F., and Garcke, J. Explainable machine learning for scientific insights and discoveries. *Ieee Access*, 8:42200–42216, 2020.

Saboksayr, S. S., Mateos, G., and Cetin, M. Online discriminative graph learning from multi-class smooth signals. *Signal Processing*, 186:108101, 2021.

Saldana, D. F., Yu, Y., and Feng, Y. How many communities are there? *Journal of Computational and Graphical Statistics*, 26(1):171–181, 2017.

Sandryhaila, A. and Moura, J. M. F. Discrete Signal Processing on Graphs: Frequency Analysis. *IEEE Transactions on Signal Processing*, 62(12):3042–3054, May 2014. ISSN 1053-587X, 1941-0476. doi: 10.1109/TSP.2014.2321121.

Sardellitti, S., Barbarossa, S., and Di Lorenzo, P. Enabling prediction via multi-layer graph inference and sampling. In *2019 13th International conference on Sampling Theory and Applications (SampTA)*, pp. 1–4. IEEE, 2019.

Segarra, S., Marques, A. G., Mateos, G., and Ribeiro, A. Network topology inference from spectral templates. *IEEE Transactions on Signal and Information Processing over Networks*, 3(3):467–483, 2017.

Shafipour, R., Segarra, S., Marques, A. G., and Mateos, G. Identifying the topology of undirected networks from diffused non-stationary graph signals. *IEEE Open Journal of Signal Processing*, 2:171–189, 2021.

Shewchuk, J. R. Allow Me to Introduce Spectral and Isoperimetric Graph Partitioning. pp. 69, April 2016.

Shi, H.-J. M., Tu, S., Xu, Y., and Yin, W. A Primer on Coordinate Descent Algorithms. *arXiv:1610.00040 [math, stat]*, January 2017.

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, April 2013. ISSN 1053-5888. doi: 10.1109/MSP.2012.2235192.

Strehl, A. and Ghosh, J. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *Journal of machine learning research*, 3(Dec):583–617, 2002.

Thanou, D., Dong, X., Kressner, D., and Frossard, P. Learning heat diffusion graphs. *IEEE Transactions on Signal and Information Processing over Networks*, 3(3):484–499, 2017.

Von Luxburg, U. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

von Luxburg, U. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, December 2007. ISSN 0960-3174, 1573-1375. doi: 10.1007/s11222-007-9033-z.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Philip, S. Y. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

Xu, Y. and Yin, W. A Block Coordinate Descent Method for Regularized Multiconvex Optimization with Applications to Nonnegative Tensor Factorization and Completion. *SIAM Journal on Imaging Sciences*, 6(3):1758–1789, September 2013. ISSN 1936-4954. doi: 10.1137/120887795.

Yamada, K., Tanaka, Y., and Ortega, A. Time-varying graph learning based on sparseness of temporal variation. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5411–5415. IEEE, 2019.

Zass, R. and Shashua, A. A unifying approach to hard and probabilistic clustering. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, pp. 294–301 Vol. 1, Beijing, China, October 2005. IEEE. ISBN 978-0-7695-2334-7. doi: 10.1109/ICCV.2005.27.

Zhang, Z., Cui, P., and Zhu, W. Deep learning on graphs: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 2020.

A. Solutions of BCD Steps

In this section, the solutions of BCD steps are provided. We start with (11), which can be rewritten as follows:

$$\boldsymbol{\ell}^{s\,(t+1)} = \underset{\boldsymbol{\ell}^s}{\operatorname{argmin}} \, \|\boldsymbol{\ell}^s - \widehat{\boldsymbol{\ell}^s}^{(t)} + \frac{1}{\lambda_s}\widehat{\mathbf{g}}_s^{(t)}\|_F^2 \;\; \mathrm{s.\,t.}\;\; \boldsymbol{\ell}^s \leq 0, \mathbf{1}^\top\boldsymbol{\ell}^s = -n,$$
$$= \underset{\boldsymbol{\ell}^s}{\operatorname{argmin}} \, \|\boldsymbol{\ell}^s - \widehat{\boldsymbol{\ell}^s}^{(t)} + \frac{2\mathbf{k} - \mathbf{P}^\top\mathbf{d} + ({\mathbf{Z}_{\cdot s}^{(t)}}^\top\mathbf{1})\alpha_2(4\widehat{\boldsymbol{\ell}^s}^{(t)} + 2\mathbf{P}^\top\mathbf{P}\widehat{\boldsymbol{\ell}^s}^{(t)})}{\lambda_s}\|_F^2 \;\; \mathrm{s.\,t.}\;\; \boldsymbol{\ell}^s \leq 0, \mathbf{1}^\top\boldsymbol{\ell}^s = -n, \quad (14)$$

where, in the second line we substitute $\widehat{\mathbf{g}}_s^{(t)} = 2\mathbf{k} - \mathbf{P}^\top\mathbf{d} + ({\mathbf{Z}_{\cdot s}^{(t)}}^\top\mathbf{1})\alpha_2(4\widehat{\boldsymbol{\ell}^s}^{(t)} + 2\mathbf{P}^\top\mathbf{P}\widehat{\boldsymbol{\ell}^s}^{(t)})$, $\mathbf{k} \in \mathbb{R}^m$ is the upper triangular part of $\mathbf{X}\mathrm{dg}(\mathbf{Z}_{\cdot s}^{(t)})\mathbf{X}^\top$, and $\mathbf{d} \in \mathbb{R}^n$ is the diagonal of $\mathbf{X}\mathrm{dg}(\mathbf{Z}_{\cdot s}^{(t)})\mathbf{X}^\top$. The solution of (14) is the projection of $\widehat{\boldsymbol{\ell}^s}^{(t)} - \frac{2\mathbf{k}-\mathbf{P}^\top\mathbf{d}+({\mathbf{Z}_{\cdot s}^{(t)}}^\top\mathbf{1})\alpha_2(4\widehat{\boldsymbol{\ell}^s}^{(t)}+2\mathbf{P}^\top\mathbf{P}\widehat{\boldsymbol{\ell}^s}^{(t)})}{\lambda_s}$ onto the negative simplex, which can be performed efficiently using the algorithm described in (Duchi et al., 2008). To solve (10), we rewrite it as follows:

$$\mathbf{Z}^{(t+1)} = \underset{\mathbf{Z}}{\operatorname{argmin}} \, \|\mathbf{Z} - \widehat{\mathbf{Z}}^{(t)} + \frac{1}{\lambda_Z}\widehat{\mathbf{G}}_Z^{(t)}\|_F^2 \;\; \mathrm{s.\,t.}\;\; \mathbf{Z} \geq 0, \mathbf{Z}\mathbf{1} = \mathbf{1},$$
$$= \underset{\mathbf{Z}}{\operatorname{argmin}} \, \|\mathbf{Z} - \widehat{\mathbf{Z}}^{(t)} + \frac{2}{\lambda_Z}\mathbf{L}^c\widehat{\mathbf{Z}}^{(t)} + \frac{\alpha_1}{\lambda_Z}\mathbf{Q}_1 + \frac{\alpha_1\alpha_2}{\lambda_Z}\mathbf{Q}_2\|_F^2 \;\; \mathrm{s.\,t.}\;\; \mathbf{Z} \geq 0, \mathbf{Z}\mathbf{1} = \mathbf{1}, \quad (15)$$

where, in the second line we substitute $\widehat{\mathbf{G}}_Z^{(t)} = 2\mathbf{L}^c\widehat{\mathbf{Z}}^{(t)} + \alpha_1\mathbf{Q}_1 + \alpha_1\alpha_2\mathbf{Q}_2$, $\mathbf{Q}_1 \in \mathbb{R}^{p\times k}$ whose sth column is $\mathbf{Q}_{1,s\cdot} = \mathrm{dg}(\mathbf{X}^\top\mathrm{mt}(\boldsymbol{\ell}^{s\,(t+1)})\mathbf{X})$, and $\mathbf{Q}_2 \in \mathbb{R}^{p\times k}$ whose sth column is $(2\langle\boldsymbol{\ell}^{s\,(t+1)}, \boldsymbol{\ell}^{s\,(t+1)}\rangle + \langle\mathbf{P}\boldsymbol{\ell}^{s\,(t+1)}, \mathbf{P}\boldsymbol{\ell}^{s\,(t+1)}\rangle)\mathbf{1}$. The problem in (15) can be solved separately with respect to rows of $\mathbf{Z}$. Let $\mathbf{A} = \widehat{\mathbf{Z}}^{(t)} + \frac{2}{\lambda_Z}\mathbf{L}^c\widehat{\mathbf{Z}}^{(t)} + \frac{\alpha_1}{\lambda_Z}\mathbf{Q}_1 + \frac{\alpha_1\alpha_2}{\lambda_Z}\mathbf{Q}_2$, then the subproblem of (15) with respect to ith row of $\mathbf{Z}$ is:

$$\mathbf{Z}_{i\cdot}^{(t+1)} = \underset{\mathbf{Z}_i}{\operatorname{argmin}} \, \|\mathbf{Z}_{i\cdot} - \mathbf{A}_{i\cdot}\|_2^2 \;\; \mathrm{s.\,t.}\;\; \mathbf{Z}_{i\cdot} \geq 0, \mathbf{Z}_{i\cdot}^\top\mathbf{1} = 1, \quad (16)$$

whose solution is the projection of $\mathbf{A}_{i\cdot}$ onto the positive simplex, which can be performed efficiently using the algorithm described in (Duchi et al., 2008).