OrthoNet: Multilayer Network Data Clustering

Mireille El Gheche, Giovanni Chierchia, and Pascal Frossard

***Abstract*—Network data appears in very diverse applications, like biological, social, or sensor networks. Clustering of network nodes into categories or communities has thus become a very common task in machine learning and data mining. Network data comes with some information about the network edges. In some cases, this network information can even be given with multiple views or layers, each one representing a different type of relationship between the network nodes. Increasingly often, network nodes also carry a feature vector. We propose in this paper to extend the node clustering problem, that commonly considers only the network information, to a problem where both the network information and the node features are considered together for learning a clustering-friendly representation of the feature space. Specifically, we design a generic two-step algorithm for multilayer network data clustering. The first step aggregates the different layers of network information into a graph representation given by the geometric mean of the network Laplacian matrices. The second step uses a neural net to learn a feature embedding that is consistent with the structure given by the network layers. We propose a novel algorithm for efficiently training the neural net via gradient descent, which encourages the neural net outputs to span the leading eigenvectors of the aggregated Laplacian matrix, in order to capture the pairwise interactions on the network, and provide a clustering-friendly representation of the feature space. We demonstrate with an extensive set of experiments on synthetic and real datasets that our method leads to a significant improvement w.r.t. state-of-the-art multilayer graph clustering algorithms, as it judiciously combines nodes features and network information in the node embedding algorithms.**

***Index Terms*—Multilayer Graph, Multiview Network, SPD Manifold, Spectral Clustering, Unsupervised Learning.**

I. Introduction

NETWORK data is getting increasingly popular in machine learning and data science, as it corresponds to a natural data representation form in biological, social, computer, or sensor network applications, to cite a few examples. Network data can be mathematically described by graphs whose vertices and edges correspond to the network nodes and links respectively. Even in applications where the network information is not explicit, graphs can be used to model the pairwise relationships between data points. Moreover, applications often rely on multiple sources of information to characterize the relationships between data. This leads to multilayer network representations, where nodes are shared across network layers, each one describing a different type of relationship between network nodes. In addition, it is often possible to associate attributes with the network nodes, which may represent different forms of measurements or signals. For example, a public transport system can be represented by a multilayer graph, where the nodes are transportation hubs, each layer describes a different mean of transportation (a bus line, a metro line, etc.), and the node attribute is the number of travelers at each hub. This obviously leads to very rich datasets, and it becomes important to devise machine learning methods that are able to consider altogether the information of both the multilayer network and the node features.

Multilayer networks are considered in many machine learning and data mining tasks, including inference of mixture models, multi-view learning, processing, clustering, and community detection. We focus here on the multilayer network data clustering problem, where the goal is to assign each network node (shared across different layers) to a cluster, by taking into account both the feature vectors on the nodes and the connectivity patterns in each layer. Multilayer network data clustering differs from common classes of clustering methods in two main aspects: (i) the information about cluster membership must be estimated from multiple network layers, while classical network clustering only considers a single layer; (ii) the clusters are formed by considering both node features and network information, while graph clustering algorithms usually only rely on network information.

In this paper, we present a general approach for multilayer network data clustering, which exploits both the Riemannian geometry of the symmetric positive definite (SPD) manifold and the power of neural nets to learn a proper node embedding. Given the intrinsic difficulty of jointly considering both node features and network information, we propose a constructive solution that works in two consecutive steps. Firstly, we compute the geometric mean of Laplacian matrices associated to each layer of the network. Aggregating the multilayer network into a graph representation with the form of a SPD matrix allows us to properly take into account the topology shared across layers. Secondly, we use the aggregated SPD matrix and the node features to perform deep spectral clustering. Unlike the standard approach of Laplacian matrix eigendecomposition, we reformulate spectral clustering as a trace optimization problem subject to an orthogonality constraint, and we devise a new algorithm to solve it. The peculiarity of our approach is that the orthogonality constraint is enforced implicitly, leading to a differentiable cost function that can be optimized via gradient descent. This allows us to use a neural net for learning the node feature embedding, which is thereby trained without supervision. Similarly to spectral clustering, the goal is to find a nonlinear mapping of the node features that penalizes the pairwise interactions provided by the aggregated SPD matrix, while enforcing the orthogonality constraint in the low-dimensional space to avoid trivial solutions.

Experimental results on diverse datasets show that the pro-

Mireille El Gheche and Pascal Frossard are with Signal Processing Laboratory (LTS4), Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland. E-mail: mireille.elgheche@epfl.ch, pascal.frossard@epfl.ch

Giovanni Chierchia is with Université Paris-Est, LIGM UMR 8049, CNRS, ESIEE Paris, Noisy-le-Grand, France. E-mail: giovanni.chierchia@esiee.fr

Giovanni Chierchia was supported by the French CNRS INS2I with a PEPS JCJC project funded under the grant 2019OSCI.

posed approach has a better clustering performance compared to baseline multilayer network data clustering approaches, due to the effective combination of network and node feature information. We expect that our algorithm can provide a new generic solution for the effective processing of rich network datasets with combinations of different forms of information.

The remaining of this paper is organized as follows. Section II reviews the literature on multilayer network data clustering. Section III describes the problem formulation. Section V details our approach for node feature embedding based on a learning objective inspired from spectral clustering. Section IV presents our approach for layer aggregation based on the geometric mean of SPD matrices. Section VI provides an experimental validation of the proposed approach on synthetic and real multilayer graphs. Section VII draws the conclusion.

II. Related work

A wide panel of approaches were proposed to combine the information from multilayer networks, and an intense research effort was dedicated to clustering methods. In this section, we review the literature on multilayer network aggregation and graph representation learning.

A. Multilayer network aggregation

The most straightforward way to summarize the information from a multilayer graph is to perform a linear or convex combination of its layers [1]–[3]. While convex combinations can be efficient in some cases, they may not be able to capture the specificity present in each layer. In this regard, more effective ways to merge the graph layers is to make use of the family of matrix power mean [4], or to see them as points of a Grassmann manifold [5].

A different aggregation strategy consists of integrating the information from individual layers directly into the optimization problem underlying the learning process. Examples include the co-EM clustering algorithm [6], the clustering approach in [7] based on co-training [8] and co-regularization [9], as well as the joint fusion and clustering approach in [10]. These methods can be useful when a unified representation for the multiple views is not easy to find in the data. In [11], each graph layer is modeled as a subspace on a Grassman manifold, and they are combined by finding the subspace that minimizes the sum of projection distances to all layers.

Closer to this paper, the work in [11] performs the aggregation in the Grassmann manifold. However, it lacks a meaningful summarization of the information contained in graph layers, and neglects any attribute that may be assigned to the graph nodes. The main novelty of the proposed approach w.r.t. [11] lies in the introduction of a new numerical algorithm to combine the characteristics of graph layers in the SPD manifold, and the design of a new approach that takes into account features carrying relevant information about the nodes.

B. Graph representation learning

In the study of graphs and networks, community detection refers to the problem of grouping together nodes that are more densely connected internally than with the rest of the network [12]–[15]. In this paper, we are mainly interested in graph clustering based on spectral analysis [16]. Spectral clustering can be linked to dimensionality reduction, which aims at representing graphs and/or high-dimensional data into low-dimensional spaces (also called embedding), while preserving both the graph topological structure and the node content information. In this regard, one of the most popular techniques consists of embedding the graph nodes into the subspace spanned by the eigenvectors of the graph Laplacian matrix corresponding to the K smallest eigenvalues [17], [18], where clusters can be easily detected via K-means algorithm [19]. Extensions of this approach consider the introduction of suitable constraints into the problem formulation, with the aim of conveying some prior knowledge on the cluster analysis [20], [21]. Alternatively, one can use the first K eigenvectors of the graph Laplacian matrix in a modularity maximization problem [22], [23]. Another approach hinges around the interpretation of Principal Component Analysis (PCA) on graphs [24], which again links the graph structure to a subspace spanned by the top eigenvectors of the graph Laplacian matrix. Moreover, numerous methods have been proposed in the literature for representation learning on graphs, such as multidimentional scaling (MDS) [25], Laplacian eigenmap [26], IsoMap [27], LLE [28], matrix factorization [29], [30], random walks [31], and deep learning approaches [32]–[35].

The works in [33], [35] propose to learn a nonlinear map that embeds data points into the eigenspace of their associated graph Laplacian matrix, and subsequently clusters them. Differently from [33], [35], we use a multilayer graph signal, and we propose a new algorithm for learning the nonlinear map. In this respect, the originality of our approach lies in the reformulation of the optimization problem, in which we replace the orthogonality constraint with a differentiable operation injected directly into the cost function. In this regard, the main advantage of our approach is to avoid the complexity of alternating between a projection step and a gradient step like in [33], as the latter may slow down the training process.

III. OrthoNet Framework

A. Problem Formulation

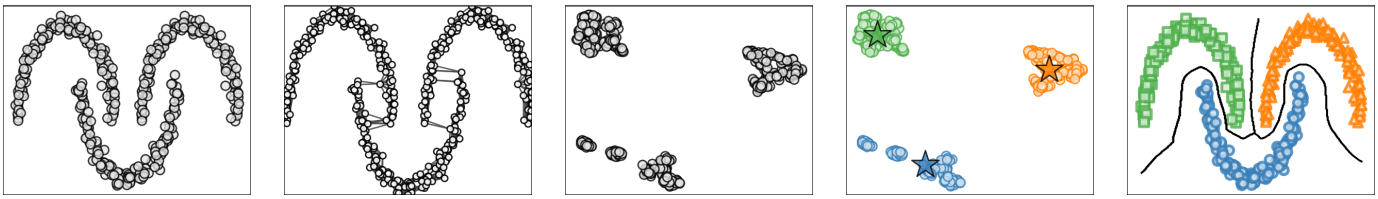
We are interested in clustering a multilayer graph

$$\mathcal{G} = \left\{\mathcal{G}^s(V, E^s)\right\}_{1 \leq s \leq S} \tag{1}$$

defined on a set V of N vertexes shared across $S \geq 1$ layers of edges.[1] For each layer $s \in \{1, \ldots, S\}$, there is a graph $\mathcal{G}^s(V, E^s)$ with (non-negative) similarity edge weights. We denote by $W^s = [w^s_{i,j}] \in \mathbb{R}^{N \times N}$ the weighted adjacency matrix of $\mathcal{G}^s$, and by $D^s = \mathrm{diag}(d^s_1, \ldots, d^s_N)$ the diagonal matrix of vertex degrees $d^s_i = \sum_j w^s_{ij}$ for all $i \in \{1, \ldots, N\}$. The Laplacian matrix of $\mathcal{G}^s$ is thus defined as

$$L^s = D^s - W^s. \tag{2}$$

[1]In this paper, the terms graph and network, vertex and node, as well as multilayer, multiview, and multiplex are used interchangeably.



(a) Features (b) Nearest-neighbor graph (c) Feature embedding (d) K-means clustering (stars mark the cluster centers) (e) Classification (lines mark the decision boundaries)

Fig. 1. Illustration of the proposed framework. Given the features (a) and the graph (b) representing their interactions, a mapping is learned so as to transform the features of strongly connected nodes into close vectors of a latent space (c). Doing so, the mapped features yield a representation that can be easily clustered with K-means (d). In addition, as the learned mapping can be applied to any feature vector, it is possible to define a classifier (e) that takes its decision based on the distance of a latent vector to the cluster centers computed on the feature embedding. The illustration is given for $N = 500, M = 2, K = 3, S = 1$.

We further assume that each vertex of the multilayer graph $\mathcal{G}$ is associated with M-dimensional features, and we denote such network data as

$$X = \begin{bmatrix} x_1^\top \\ \vdots \\ x_N^\top \end{bmatrix} \in \mathbb{R}^{N \times M}. \tag{3}$$

Our goal is to cluster the graph vertices by taking into account both the multilayer structure of $\mathcal{G}$ and the node features X, without any a priori information about the actual relationship between the graph layers and the features. We however rely on three minimal assumptions, listed below.

1) **Node connectivity.** Nodes that are connected in multiple graph layers are more likely to belong to the same cluster.
2) **Layer complementarity.** Individual layers provide at least a partial information on the clustering structure.
3) **Feature correlation.** Features for nodes within the same clusters are likely to be more correlated than features for nodes in different clusters.

This setting is especially useful in scenarios where the topology shared across layers provides information that is not fully present neither in the data, nor in each layer alone.

As we do not assume any a priori model between the graph layers and the node features, we aim at discovering their relationship from the data. To this end, we propose a learning approach that exploits the topology shared across layers to drive the unsupervised training of a feature mapping. We define a learning objective that encourages the features of strongly connected nodes to be mapped to close vectors of a latent space. By doing so, the learned mapping bears similarity with spectral clustering, and yields a clustering-friendly representation of the node features.

In cases where node connectivity implies feature correlation (see our assumptions), the mapping actually learns to represent correlated features as close vectors of the latent space, thus yielding a clustering-friendly representation of the whole feature space. This makes it possible to obtain a classifier that generalizes to any feature vector, included but not limited to those associated with the graph nodes.

The proposed learning framework can be formulated as the joint optimization problem of finding the graph that is representative of all layers, and the mapping that allows for the clustering of its nodes. In general, this task is complex to solve, especially since we have no assumption on the interactions between graph layers and node features. We present a constructive solution to this problem in the next section.

B. Proposed approach

Our approach is based on the idea of using the multilayer graph information, especially the information that appears consistently across layers, to drive the unsupervised learning of a mapping on the node features. Given the intrinsic difficulty of this joint optimization problem, we propose an alternative solution in two consecutive steps, detailed in the following.

1) **Layer aggregation**. In the first step, we merge the individual layers into a representative graph G given by its Laplacian matrix L. This operation is performed directly on the Laplacian matrices through an operator $\Phi\colon (\mathbb{R}^{N\times N})^S \to \mathbb{R}^{N\times N}$, that is
$$L = \Phi(L^1, \ldots, L^S). \tag{4}$$
We propose to compute L as the geometric mean (in the SPD manifold) of the Laplacian matrices L^s given by the layers $\mathcal{G}^s$. This allows us to summarize the topology shared across layers into a single Laplacian matrix, as we shall explain in Section IV.
2) **Feature embedding**. In the second step, we estimate the parameters $\theta \in \mathbb{R}^B$ of a nonlinear mapping defined as
$$f_\theta\colon \mathbb{R}^M \to \mathbb{R}^K, \tag{5}$$
which embeds the node features in a latent space of dimension equal to the number K of desired clusters. We perform this task using a learning objective inspired from spectral clustering, where the representative graph given by the Laplacian matrix L drives the unsupervised learning of the mapping on the node features, as we will present in Section V.

Once the mapping f_θ has been learned from the multilayer graph, the node features are transformed as

$$Y_\theta = f_\theta(X) = \begin{bmatrix} f_\theta(x_1)^\top \\ \vdots \\ f_\theta(x_N)^\top \end{bmatrix} \in \mathbb{R}^{N\times K}. \tag{6}$$

The matrix Y_θ provides a clustering-friendly representation of the graph nodes. This is ensured by our learning objective,

which encourages the mapping f_θ to represent the features of strongly connected nodes as close vectors of $\mathbb{R}^K$, while enforcing the orthogonality of such vectors to split them into separate groups. Therefore, the rows of Y_θ can be clustered with K-means to group together the nodes that are the most strongly connected by the representative graph.

Moreover, since the features of connected nodes are correlated (see our assumptions), the mapping actually learns to represent correlated features as close vectors of the latent space. This leads to a clustering-friendly representation of the whole feature space, because the learned mapping can be applied to any feature vector, and not only to those associated with the graph nodes. A generic feature vector $x \in \mathbb{R}^M$ can be thus classified based on the distance of its embedding $f_\theta(x)$ to the cluster centers $\{c_1, \dots, c_K\}$ computed on the graph at training time, yielding the classifier defined as

$$(\forall x \in \mathbb{R}^M) \quad p_{\text{class}}(x) = \operatorname*{argmin}_{k \in \{1,\dots,K\}} \|f_\theta(x) - c_k\|. \tag{7}$$

Fig. 1 presents an overview of the proposed framework.[2] More details about the two main steps of the proposed framework will be provided in the next sections.

IV. Layer aggregation

A. Problem formulation

We now present the formulation for layer aggregation, which exploits the Riemannian geometry of SPD manifold to merge the Laplacian matrices L^s of individual layers $\mathcal{G}^s$ into a single SPD matrix that describes the representative graph G.

The notions of arithmetic and geometric means, typically used to average positive numbers, generalize naturally to a finite set of SPD matrices. This generalization is based on a variational characterization of the mean operation, which consists in finding the SPD matrix L that minimizes its distance to a set of SPD matrices $\{L^s\}_{1 \le s \le S}$, that is

$$L = \Phi(L^1, \dots, L^S) = \operatorname*{argmin}_{L \in \mathcal{P}(N)} \sum_{s=1}^{S} \mathcal{D}(L, L^s), \tag{8}$$

where $\mathcal{P}(N)$ denotes the manifold of $N \times N$ SPD matrices, and $\mathcal{D}\colon \mathcal{P}(N) \times \mathcal{P}(N) \to \mathbb{R}$ is a suitable distance.

B. Geometric mean

When the dissimilarity between SPD matrices is computed via the Euclidean distance, the solution to Problem (8) is the arithmetic mean of $\{L^s\}_{1 \le s \le S}$. The latter is however suboptimal to merge information from different layers, and a better alternative for graph clustering is given by the matrix power mean [4], which can perfectly recover the clusters of complementary layers sampled from the stochastic block model when the power goes to $-\infty$.

A different family of matrix power means can be defined based on the Riemannian distance [40]. In this context, the geometric mean arises as a special case of interest in machine learning [41], [42]. The geometric mean of SPD matrices $\{L^s\}_{1 \le s \le S}$ corresponds to the solution to Problem (8) with the Riemann distance[3]

$$\mathcal{D}(L, L^s) = \left\| \operatorname{Log}\left(L^{-\frac{1}{2}} L^s L^{-\frac{1}{2}}\right) \right\|_F^2, \tag{9}$$

where Log denotes the principal logarithm of a SPD matrix.

C. Numerical computation

We propose to aggregate the graph layers by computing the geometric mean of the respective Laplacian matrices. Formally, the geometric mean arises as the solution to Problem (8) in the case when $\mathcal{D}$ is the Riemann distance given in (9). The problem however admits no close-form solution for $S > 2$. We thus resort to numerically compute the geometric mean through the Fréchet-Karcher gradient flow [43], whose iterations are defined as follows

$$(\forall t \in \mathbb{N}) \quad L_{t+1} = L_t^{\frac{1}{2}} \operatorname{Exp}\Big(\beta \sum_{s=1}^{S} \operatorname{Log}\big(L_t^{-\frac{1}{2}} L^s L_t^{-\frac{1}{2}}\big)\Big) L_t^{\frac{1}{2}}. \tag{10}$$

Here above, $L_0 = \sum_{s=1}^{S} L^s$ is the initialization, $\beta > 0$ is the step size, and Exp denotes the exponential of a symmetric matrix. Riemannian gradient descent converges to the optimal solution with a rate of $O(1/k)$ for the geodesically convex problem considered here [44]. In practice, one iteration with $\beta = 1$ suffices to find a good approximation of the solution. This operation has a time complexity of $O(N^3)$, due to the presence of matrix logarithms and exponentials.

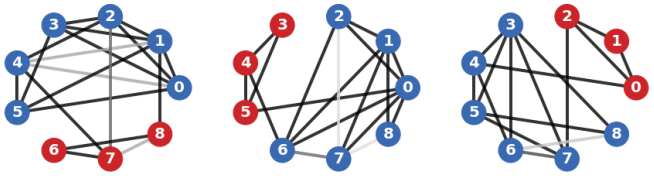
D. Illustrative example

Fig. 2 presents an example of layer aggregation, where the proposed geometric mean is compared to the arithmetic mean and the projection mean [11]. In this example, the original graph is composed of three layers, which only provide a partial information on the clustering structure. The geometric mean yields a graph that is representative of all the layers, as spectral clustering manages to recover from it the three blocks appearing in the respective layers. On the contrary, the arithmetic mean tends to underestimate the importance of edge $(7, 8)$ despite that it appears in two layers, whereas the projection mean tends to overestimate the importance of edge $(0, 8)$ despite that it appears in only one layer, leading to the incorrect assignment of nodes 2 and 8.

In the example of Fig. 2, the merging of edges $(2, 7)$, $(7, 8)$, $(8, 0)$, and $(6, 8)$ is critical for correctly recovering the three clusters. Small changes of their weights result in different clustering for both the arithmetic and the projection mean.

[2]The minimal requirement of our framework is to have at least a single-layer graph with an attribute for each node (in which case the first step is dropped), but additional layers and attributes are usually beneficial for the performance, as long as they are at least partially informative. In cases where the node features are not given, a one-hot indicator vector can be assigned to each node [36]. Conversely, if no graph is given, one or more layers can be built directly from the data, e.g., by computing several nearest-neighbor graphs on feature subsets. Although layers derived from the data might seem like redundant information, this is a common practice in image processing [37]–[39] for example, where the nonlocal graph of similar patches is effectively used as a prior information to capture long-range correlations in the data.

[3]Note that the principal logarithm is not well defined on Laplacian matrices, as they are just positive semi-definite. To circumvent this issue, we add a small diagonal shift to ensure positive definiteness. In other terms, we implicitly assume that $L^s = \bar{L}^s + \epsilon I$, where ϵ is a small positive constant.



(a) Individual layers composed by two complemetary blocks. The magnitude of edge weights is gray colored from white (small) to black (large). Node coloring is the result of spectral clustering with $K = 2$ on each layer.

(b) Representative graphs obtained by aggregating the Laplacian matrices of individual layers with different techniques. *Left:* Arithmetic mean. *Middle:* Projection mean [11]. *Right:* Geometric mean (proposed). Node coloring is the result of spectral clustering with $K = 3$ on each aggregated graph.

Fig. 2. Illustrative example of layer aggregation. The original graph is composed of three layers, which only provide partial information on the clustering structure. The geometric mean successfully merges the partial views into a representative graph with three distinct blocks (one from each layer).

This is not the case for the geometric mean, which provides a consistent aggregation for a wide range of edge weights across different layers. Indeed, the Riemann distance is known to give equal importance to all eigenvalues, regardless of their magnitude. As a result, the geometric mean is more robust to small fluctuations of edge weights, and thus better suited to preserve the structural information of multilayer graphs.

V. Node feature embedding

A. Problem formulation

We now present the formulation for feature embedding, which relies on both the node attributes X and the Laplacian matrix $L = [L_{ij}]$ of the representative graph G.

The goal is to estimate a mapping f_θ that represents the features of strongly connected nodes as close points in the latent space. That is, for large similarities $w_{ij} = -L_{ij}$, we want the distance between $f_\theta(x_i)$ and $f_\theta(x_j)$ to be small, which amounts to minimizing

$$\mathrm{Tr}(Y_\theta^\top L Y_\theta) = \sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}\|f_\theta(x_i) - f_\theta(x_j)\|^2, \qquad (11)$$

where $Y_\theta = f_\theta(X)$ is defined in (6). This objective is however not sufficient alone for learning an embedding that would result in effective clustering, as the sum of pairwise distances is trivially minimized by mapping all points to the same output vector. To avoid trivial solutions, we can borrow from spectral clustering the idea that the embedded points must be orthogonal to each other [33], [35], yielding

$$\underset{\theta\in\mathbb{R}^B}{\text{minimize}} \quad \mathrm{Tr}(Y_\theta^\top L Y_\theta) \quad \text{s.t.} \quad Y_\theta^\top Y_\theta = \mathbf{I_{K\times K}}. \qquad (12)$$

The matrix Y_θ represents the graph nodes as vectors of the latent space $\mathbb{R}^K$. Intuitively, nodes within the same cluster should be mapped to close vectors, while nodes from different clusters should be spaced out from each other, so that the latent space can be easily clustered. By optimizing the sum of pairwise distances over the orthogonality constraint, the rows of Y_θ tend to be split into K clusters, which are formed by grouping together the more strongly connected vertexes in the graph. This is indeed similar to spectral clustering,[4] which uses the spectrum of the Laplacian matrix to perform dimensionality reduction before clustering [17], [18], [45]. An illustrative example of node feature embedding is presented in Fig. 1, where the mapping f_θ is implemented by a neural net with four fully-connected layers and ReLU activations.

Note that many formulations have been proposed for representation learning on graphs [32]. There is however a clear distinction between classification scenarios, where metric learning methods are well established [46]–[50], and clustering scenarios. In the latter context, most approaches adopt a model based on two mapping functions: an *encoder* that embeds each node into a low-dimensional vector, and a *decoder* that recovers high-dimensional graph information (e.g., the node positions) from the learned embeddings. In particular, the decoder is needed for the definition of a self-supervised loss function that measures the discrepancy between the decoded similarity values and the true similarity values in the graph.

The peculiarity of Problem (12) is that the decoder is replaced by the orthogonality constraint. The advantage of this solution is that the mapping f_θ can directly learn the structural information provided by the graph, rather than indirectly using it through self-supervision. In addition, the mapping f_θ can be implemented by any parametric function from $\mathbb{R}^M$ to $\mathbb{R}^K$. This includes neural nets, whose only requirement is to end up with a layer of K units. In practice, a small neural net with few fully-connected or graph-convolutional layers [32] is sufficient to effectively disentangle the data in a low-dimensional space.

B. Proposed optimization algorithm

We propose to solve Problem (12) with an optimization algorithm based on gradient descent. The main difficulty of this optimization problem arises from the orthogonality constraint, since it is enforced on the mapping to be estimated, rather than the optimization parameters, ruling out standard techniques based on alternating optimization [51]. While this problem was recently tackled in [33], [35], we propose an alternative algorithm based on implicitly constrained optimization, which extends our preliminary work [52].

Our idea is that the orthogonality constraint in Problem (12) can be enforced implicitly by using the upper triangular matrix $R_\theta^\top \in \mathbb{R}^{K\times K}$ of the QR decomposition of Y_θ, defined as

$$Y_\theta = Q_\theta R_\theta^\top, \qquad (13)$$

with $Q_\theta \in \mathbb{R}^{N\times K}$ being a semi-orthogonal matrix. Indeed, when Y_θ is full rank, or equivalently $Y_\theta^\top Y_\theta$ is positive definite, its QR decomposition is unique, and R_θ is equal to the lower triangular factor of the Cholesky decomposition

$$Y_\theta^\top Y_\theta = R_\theta R_\theta^\top. \qquad (14)$$

[4]The connection to spectral clustering becomes apparent by setting $\theta \in \mathbb{R}^{N\times K}$, $X = \mathbf{I_{N\times N}}$, and $f_\theta(X) = X\theta$, so as to have $Y_\theta = \theta$.

Algorithm 1 Gradient descent for Problem (16)

Require: $L \in \mathbb{R}^{N\times N}$ ▷ Laplacian matrix
Require: $X \in \mathbb{R}^{N\times M}$ ▷ Network data
Require: $f_\theta \colon \mathbb{R}^M \mapsto \mathbb{R}^K$ ▷ Neural net
Require: $\theta_0 \in \mathbb{R}^B$ ▷ Initialization
Require: $\gamma > 0$ ▷ Step size
for $t = 0, 1, \dots$ **do**
  $Y_t = f_{\theta_t}(X)$
  $R_t = \mathrm{Cholesky}(Y_t^\top Y_t)$
  $\overline{D}_t = R_t^{-T} R_t^{-1}$
  $\overline{Y}_t = 2\big(LY_t\overline{D}_t - Y_t\overline{D}_t Y_t^\top LY_t\overline{D}_t\big)$
  $g_t =$ gradient of J at θ_t based on $\overline{Y}_t$ ▷ See (18)
  $\theta_{t+1} =$ gradient-step(θ_t, g_t, γ) ▷ See [53]
return $f_{\theta_*}(X)R_*^{-\top}$

Therefore, the semi-orthogonal factor Q_θ can be extracted from the matrix Y_θ by multiplying it with $R_\theta^{-\top}$, namely

$$Q_\theta = Y_\theta R_\theta^{-\top} \quad \Rightarrow \quad Q_\theta^\top Q_\theta = \mathbf{I}_{\mathbf{K}\times\mathbf{K}}. \tag{15}$$

This consideration allows us to rewrite Problem (12) as

$$\underset{\theta\in\mathbb{R}^B}{\text{minimize}} \;\; J(\theta) := \mathrm{Tr}(R_\theta^{-1} Y_\theta^\top LY_\theta R_\theta^{-\top}) \tag{16}$$

$$\text{with} \quad R_\theta := \mathrm{Cholesky}(Y_\theta^\top Y_\theta). \tag{17}$$

In the above reformulation, the term Y_θ is no longer a semi-orthogonal matrix. The constraint is now enforced implicitly through the factor R_θ derived from the Cholesky decomposition of $Y_\theta^\top Y_\theta$, which ensures that the product $Y_\theta R_\theta^{-\top}$ is a semi-orthogonal matrix of $\mathbb{R}^{N\times K}$. This makes it possible to steer the embedding Y_θ away from trivial solutions.

C. Gradient descent

All operations involved in Problem (16) are differentiable, provided that the mapping f_θ is defined by a differentiable operator, such as a neural net. Specifically, the gradient of the cost function J defined in (16) can be decomposed as

$$\nabla J(\theta) = \left[\mathrm{Tr}\left(\frac{\partial J(\theta)}{\partial Y_\theta}^\top \frac{\partial Y_\theta}{\partial \theta^{(b)}}\right)\right]_{1\le b\le B} \tag{18}$$

where the Jacobian w.r.t. Y_θ (derived in the appendix) reads

$$\frac{\partial J(\theta)}{\partial Y_\theta} = 2\big(\mathbf{I}_{\mathbf{N}\times\mathbf{N}} - Y_\theta R_\theta^{-T} R_\theta^{-1} Y_\theta^\top\big) LY_\theta R_\theta^{-T} R_\theta^{-1}. \tag{19}$$

Thanks to this result, a solution to Problem (16) can be found via gradient descent, whose iterations are summarized in Algorithm 1. There are several advantages in solving Problem (16) with this approach. We avoid the complexity of alternating between a projection step and a gradient step [33], as the alternating approach may slow down the convergence to the optimal solution. Moreover, we can reduce the complexity of gradient updates via stochastic approximations [35], [52].

A question remains on the equivalence between the original problem (12) and the proposed reformulation (16). By the Courant-Fischer theorem, the solution $Y_{\bar\theta}$ to Problem (12) closely approximates the K smallest eigenvectors of the matrix L, up to the representational capacity of the mapping $f_{\bar\theta}$. This is however not true for the solution $Y_{\bar\theta}R_{\bar\theta}^{-\top}$ derived from Problem (16), which only spans the smallest K eigenvectors of the matrix L [54]. To see this, note that we can rewrite the cost function reformulated in (16) as follows:

$$\begin{aligned}\mathrm{Tr}(R_\theta^{-1} Y_\theta^\top LY_\theta R_\theta^{-\top}) &= \mathrm{Tr}(R_\theta^{-\top} R_\theta^{-1} Y_\theta^\top LY_\theta) &(20)\\ &= \mathrm{Tr}\left((R_\theta R_\theta^{-\top})^{-1} Y_\theta^\top LY_\theta\right) &(21)\\ &= \mathrm{Tr}\left((Y_\theta^\top Y_\theta)^{-1} Y_\theta^\top LY_\theta\right). &(22)\end{aligned}$$

In the case $K = 1$, this boils down to the Rayleigh quotient

$$\mathcal{Q}(y) = \frac{y^\top Ly}{y^\top y}, \tag{23}$$

whose minimizer is the smallest eigenvector of L. For $K > 1$, the function (22) is invariant to right-multiplications of Y_θ, and thus the minimum is achieved by any basis that spans the K smallest eigenvectors of L. This is not critical in our context, as a basis change in the embedding space does not affect clustering, allowing us to learn a valid mapping Y_θ without the need to explicitly compute the eigenvectors of L. Note also that we could directly minimize (22) as in [35]. In this regard, the proposed cost function J may be better suited for optimization, because it leads to a symmetric gradient, which improves stability during training.

VI. Experimental results

A. Preliminaries

We compare our approach with six clustering algorithms. The methods referred to as SC-ML [11], MIMOSA [3], and PLM [4] work in two steps: they aggregate the Laplacian matrices of the multilayer graph, and then perform the spectral clustering of the resulting (single-layer) graph.[5] The difference lies in the aggregation step, which is performed in Grassman manifold, via a convex combination, or using the power Laplacian mean, respectively. Moreover, the method called GMC [10] jointly performs layer aggregation and spectral embedding, and the resulting embedding matrix is clustered with K-means. The method called SpectralNet [33] builds a graph from the data points (features), estimates a nonlinear mapping by solving Problem (12), and then embeds the features in a low-dimensional space, where they are clustered with K-means. We also report the performance of K-means clustering solely on the node features, without using the structural information carried by the multilayer graph.

All methods are used with their default hyperparameters. As for our method, referred to as OrthoNet, we implement the mapping as a neural net with four dense layers of size 400-200-100-K and PReLU activation [55], where K is the number of desired clusters, and the optimization is carried out with AmsGrad method [53] using a learning rate $\gamma = 10^{-3}$. We use three criteria to measure the clustering performance: Purity, Normalized Mutual Information (NMI), and adjusted Rand Index (RI). They measure the agreement of two partitions, ignoring permutations and with no requirement to have the

[5] For a single-layer graph with N nodes, a K-means clustering is computed on the rows of the matrix $U \in \mathbb{R}^{N\times K}$ formed by the eigenvectors associated to the K smallest eigenvalues of the graph Laplacian matrix, so as to group together the vertexes that are the most strongly connected by the graph.

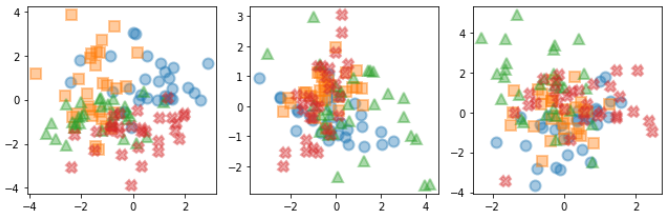


Fig. 3. Synthetic data generated with $N = 100$ vectors, $K = 4$ clusters, $S = 3$ layers, and $d = 2$ dimensions. Each panel shows the feature vectors of a layer, colored by the cluster they belong to. A k-NN graph is build on each set of vectors. The layer alignment is purely based on the clusters.

same number of clusters. Values close to zero indicate two assignments that are largely independent, while values close to one indicate significant agreement. All the experiments are conducted in Python/Numpy/PyTorch on a 40-core Intel Xeon CPU at 2.5 GHz with 128GB of RAM.

B. Datasets

In our experiments, we consider several synthetic and real datasets. The synthetic dataset consists of S point clouds of arbitrary size N, each generated from a d-dimensional Gaussian mixture model with K components having different means and covariance matrices, as shown in Fig. 3. We build a 20-nearest-neighbor (k-NN) graph on each point cloud, and we set the edge weights to the reciprocal of the Euclidean distance between pairs of neighbors. This give us S layers. Then, we concatenate the data points across the clouds, so as to form a feature vector of size $M = dS$ for each node in the multilayer graph, where the alignment of nodes across layers is known by construction. The goal is to recover the K components from which the data points are generated.

We then consider several real datasets. The IMDB database allows access to the movie's actors, directors, writers and production company, the movie's awards (wins and nominations), its box office, as well as the directors/actors/writers box office. Without having access to budget figures, the goal is to cluster the movies into $K = 5$ budget ranges: low cost (below 0.1 USD millions), low-medium cost (below 10 USD millions), medium cost (below 40 USD millions), medium-high cost (below 100 USD millions), high cost (above 100 USD millions). To build a multilayer graph on IMDB data, we connect the movies sharing one or more actors, directors, or writers, leading to $S = 3$ graph layers. Moreover, each movie is associated to $M = 3$ attributes: box office, awards, and director box office.

Moreover, Yelp is a popular website for reviewing and rating local businesses. In our experiments, we only extract star ratings, text reviews, and review evaluations (users can mark reviews as "cool", "useful", and "funny"), ignoring the other information in the dataset. The goal is to cluster the businesses into $K = 3$ rating levels: low (1 or 2 stars), medium (3 stars), high (4 or 5 stars). To build a multilayer graph on Yelp data, we proceed as follows. The text reviews are preprocessed using sentiment analysis. This yields a polarity score within the range $[-1, 1]$ on which we build a 20-NN graph. We also build a 20-NN graph on the review evaluations, leading to $S = 2$ graph layers. Moreover, each business is associated to $M = 2$ attributes: the sentiment analysis score, and the review evaluation score.

Another dataset, the "100 leaves", contains $N = 1600$ samples of $M = 192$ features for $K = 100$ plant species. We build a 5-NN graph on $S = 3$ different feature subsets: shape descriptor, fine scale margin, and texture histogram. The goal is to cluster the observations according to their plant species.

Finally, the "Mfeat" handwritten digit dataset contains $N = 2000$ samples of $M = 650$ features for $K = 10$ digits (0-9). We build a 5-NN graph on $S = 6$ different feature groups. The goal is to cluster the observations according to their digits.

C. Graph clustering: single layer versus multiple layers

We start our analysis by assessing the importance of building a multilayer graph on several subsets of features, as opposed to constructing a one-layer graph on the whole features. To this end, we compare two alternative strategies to build the graph on a given dataset.

- *Single-layer graph*. For each sample, we concatenate all the available features into one vector. Then, we use those vectors to build a single k-NN graph.
- *Multilayer graph*. For each sample, we split the available features into different subsets (see Section VI-B). Then, we build a separate k-NN graph on each subset, and we stack them into a multilayer graph.

Table I reports the clustering performance on two datasets (synthetic and Mfeat) using the above graph building strategies. We directly perform spectral clustering and OrthoNet clustering on single-layer graphs, leading to the two sets of indicators reported in the first and third line of Table I. On multilayer graphs, we first aggregate the layers using three different methods: the arithmetic mean (average), the projection mean in Grassman manifold (SC-ML), and the geometric mean in SPD manifold (proposed). Then, we perform spectral clustering or OrthoNet clustering on each aggregated graph, leading to the six sets of indicators reported in the second and fourth line of Table I. In both scenarios, spectral clustering relies only on the graph, whereas OrthoNet relies on both the graph and the features.

Among the results obtained with spectral clustering (left side of the table) and OrthoNet clustering (right side of the table), the proposed approach (SPD+OrthoNet) achieves the best performance. This empirically confirms that

- multilayer graphs carry a richer information than single-layer graphs for the purpose of node clustering,
- the geometric mean in SPD manifold is an appropriate and effective choice for layer aggregation,
- the presence of features on graph nodes can improve the clustering of multilayer graphs.

D. Multilayer graph clustering: performance assessment

Table II reports a broader comparison with the state-of-the-art methods mentioned in the previous subsection.[6] On

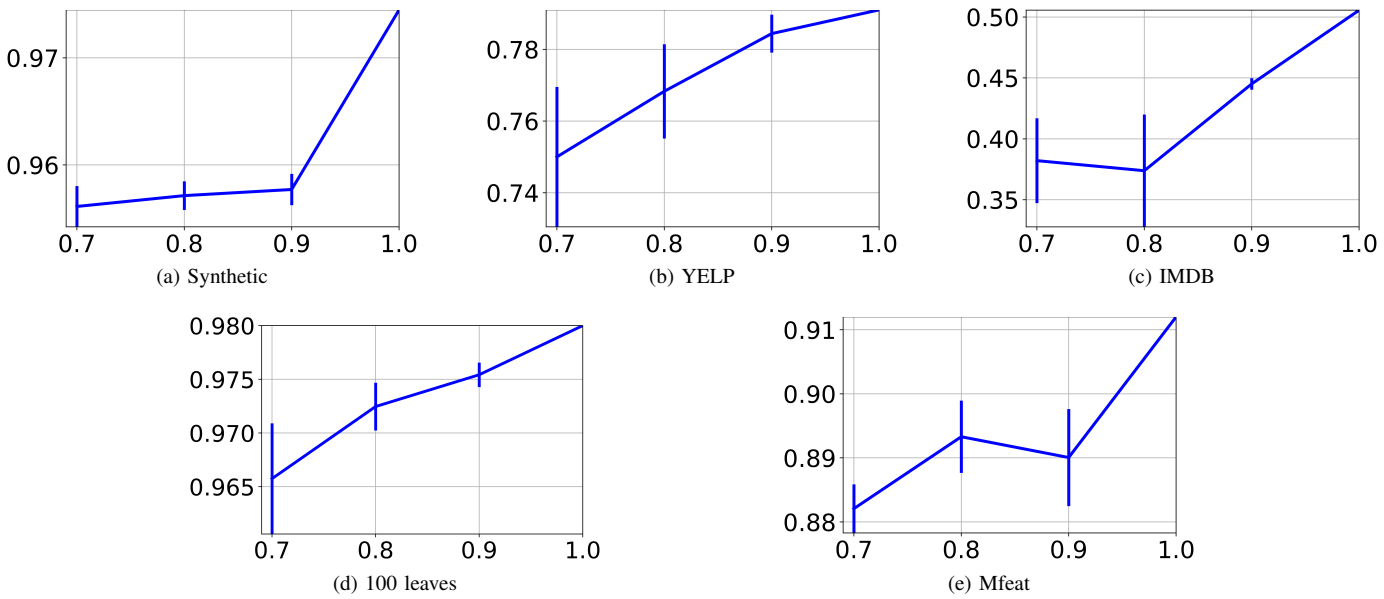
[6]We were unable to run MIMOSA on "100 leaves" dataset, due to the high number (100) of clusters.

TABLE I
ONE-LAYER GRAPH CLUSTERING VERSUS MULTILAYER GRAPH CLUSTERING

Dataset	Layer aggregation	SPECTRAL CLUSTERING (graph only) Purity	NMI	RI	ORTHONET (graph + features) Purity	NMI	RI
SYNTHETIC Nodes: 2000 – Clusters: 5 **Layers: 1** – Features: 8	None	0.9375	0.8626	0.8568	0.9395	0.8647	0.8610
SYNTHETIC Nodes: 2000 – Clusters: 5 **Layers: 4** – Features: 8	Average SC-ML SPD	0.9475 0.9555 0.9705	0.8437 0.8634 0.9075	0.8731 0.8919 0.9275	0.9535 0.9580 0.9720	0.8614 0.8718 0.9113	0.8873 0.8977 0.9312
MFEAT Nodes: 2000 – Clusters: 6 **Layers: 1** – Features: 650	None	0.7900	0.7489	0.6558	0.8000	0.7631	0.6752
MFEAT Nodes: 2000 – Clusters: 6 **Layers: 6** – Features: 650	Average SC-ML SPD	0.8395 0.8775 0.9130	0.8410 0.8780 0.8953	0.7622 0.8339 0.8575	0.8515 0.8920 0.9555	0.8371 0.8860 0.9128	0.7761 0.8438 0.9057

TABLE II
PERFORMANCE OF MULTILAYER GRAPH CLUSTERING ALGORITHMS.

Data	K-means	SC-ML	GMC	PLM	MIMOSA	SpectralNet	OrthoNet
SYNTHETIC	(Nodes: 10000, Features: 8, Layers: 4, Clusters: 5)						
Purity	0.9901	0.9858	0.3996	0.9886	0.9868	**0.9940**	**0.9948**
NMI	0.9664	0.9545	0.7299	0.9621	0.9566	**0.9720**	**0.9756**
RI	0.9756	0.9652	0.4763	**0.9910**	0.9896	0.9840	0.9827
Time	0.10 s	153.84 s	1345 s	408.62 s	1418 s	464.31 s	785.84 s
YELP	(Nodes: 1600, Features: 2, Layers: 2, Clusters: 3)						
Purity	0.8656	0.7748	0.7007	0.7616	0.7168	0.9460	**0.9570**
NMI	0.6265	0.3329	0.3925	0.6754	0.1395	0.6810	**0.7910**
RI	0.5912	0.5192	0.6010	0.8433	0.6603	0.7322	**0.9044**
Time	0.016 s	1.54 s	7.3 s	5.10 s	115.67 s	82.04 s	6.42 s
IMDB	(Nodes: 550, Features: 3, Layers: 3, Clusters: 5)						
Purity	0.8280	0.8817	0.7007	0.8495	0.5358	0.7310	**0.8925**
NMI	0.2331	0.3614	0.1563	0.2260	0.0953	0.2200	**0.5056**
RI	0.0219	0.2680	0.5245	**0.7437**	0.4347	0.1810	0.6909
Time	0.04 s	0.31 s	2.21 s	20.89 s	50.6 s	30.35 s	19.68 s
100 LEAVES	(Nodes: 1600, Features: 192, Layers 3, Clusters: 100)						
Purity	0.6987	0.9487	0.8237	0.8229	–	0.7840	**0.9712**
NMI	0.8452	0.9717	0.9292	0.9079	–	0.8370	**0.9812**
RI	0.5578	0.9129	0.4974	**0.9819**	–	0.3530	0.9478
Time	1.12 s	1.86 s	8.09 s	4.06 s	–	770 s	4.17 s
MFEAT	(Nodes: 2000, Features: 650, Layers: 6, Clusters: 10)						
Purity	0.5470	0.8775	0.8820	0.8780	0.2215	0.7685	**0.9555**
NMI	0.5744	0.8780	0.9041	0.8807	0.3549	0.7480	**0.9128**
RI	0.4293	0.8339	0.8496	0.9692	**0.9960**	0.6343	0.9057
Time	0.71 s	3.45 s	24.05 s	11.82 s	13.04 s	95.12 s	10.85 s



(a) Synthetic (b) YELP (c) IMDB

(d) 100 leaves (e) Mfeat

Fig. 4. OrthoNet performance (y-axis) in terms of NMI evaluated on all the data, after the training is performed on a fraction (x-axis) of data and graph.

the synthetic dataset, OrthoNet and SpectralNet are the best performers, whereas the aggregation-based techniques are practically equivalent. This may be related to the fact that signals are more relevant for this kind of data, than the graphs alone (which are built from the signals). On the other datasets, OrthoNet is by far the best performer, especially in terms of the NMI score. This result is probably due to the richer information carried by graph layers, which cannot be translated into features. As SpectralNet builds a similarity distance on the feature vectors (using nearest neighbors or Siamese network), it cannot rely on the benefit brought by the multilayer graph. Conversely, the proposed approach can take advantage of the information carried by both the multilayer graph and the feature vectors, leading to a better clustering performance. Such improvement is however achieved with an increase of the execution time, due to the high computational cost for computing the geometric mean of SPD matrices.

E. Generalization to new data

In our approach, we train a neural net to find a clustering-friendly representation of the feature space. While so far we focused on the clustering of graph node features, we now wish to assess the capacity to classify new feature vectors never seen before (i.e., not associated to any graph node). The difficulty here is that we deal with a completely unsupervised scenario, so the experimentation protocol based on splitting the data in train and test sets is not really meaningful. To allow for a fair comparison with the techniques reported in Table II, we train f_θ on a subset of the available graph nodes, and then we evaluate the clustering performance of f_θ on all the available feature vectors. In particular, Fig. 4 reports the NMI score obtained by training OrthoNet on 70%, 80%, 90%, and 100% of the graph nodes. For all the fractions less than 100%, we repeat the training 10 times on random subsets of the graph, and we report both mean and standard deviation of the scores. The results show a moderate drop of performance when training is performed of smaller graphs. This suggests that the learned neural net has the ability to generalize to new data, provided that the graph carries sufficient information.

VII. Conclusion

We have proposed a framework for multilayer network data clustering based on a two-step approach. We first compute the geometric mean of Laplacian matrices in the SPD manifold, and then we use the resulting graph to train a neural net on the node features in a unsupervised manner, using a formulation inspired from spectral clustering. The latter step is tackled with a new optimization algorithm that deals with the orthogonality constraint of the neural net outputs in an implicit way, so as to span the leading eingenvectors of the aggregated Laplacian matrix without the need to explicitly compute them. Experimental results show a better clustering performance of this approach on diverse datasets compared to state-of-the-art multilayer network clustering, as well as the ability of the trained neural net to generalize to new data. Interesting perspectives for future work include a better modeling of node features through a general approach to simultaneously aggregate the multilayer information with the network data.

Appendix
Derivation of the Gradient

To derive the gradient of the cost function J defined in (16), we first apply the chain rule and obtain the expression in (18). Then, the Jacobian of J w.r.t. Y_θ can be derived by reverse-mode algorithmic differentiation [56], [57]. The step-by-step computation is reported in Table III, where the standard algorithmic differentiation terminology is used: if the matrix A is an intermediate variable within the cost function

TABLE III
STEP-BY-STEP REVERSE-MODE DIFFERENTIATION

Operation	Differential of a variable	Differential of J	Jacobian of J w.r.t. a variable
$J = \mathrm{Tr}(C)$	-	$\mathrm{d}J = \mathrm{Tr}(\overline{C}^\top dC)$	$\overline{C} = I$
$C = ADA^\top$	$\mathrm{d}C = \mathrm{d}A\, DA^\top + A\,\mathrm{d}D\, A^\top + AD\,\mathrm{d}A^\top$	$\mathrm{d}J = \mathrm{Tr}\big(\overline{A}^\top \mathrm{d}A + \overline{D}^\top \mathrm{d}D\big)$	$\overline{A} = 2\,AD$ $\overline{D} = A^\top A$
$D = Y^\top LY$	$\mathrm{d}D = \mathrm{d}Y^\top LY + Y^\top L\,\mathrm{d}Y$	$\mathrm{d}J = \mathrm{Tr}\big(\overline{A}^\top \mathrm{d}A + \widetilde{Y}^\top \mathrm{d}Y\big)$	$\widetilde{Y} = 2\,LY\overline{D}$
$A = R^{-1}$	$\mathrm{d}A = -A\,\mathrm{d}R\,A$	$\mathrm{d}J = \mathrm{Tr}\big(\overline{R}^\top \mathrm{d}R + \widetilde{Y}^\top \mathrm{d}Y\big)$	$\overline{R} = -R^{-\top}\overline{A}R^{-\top}$
$R = \mathrm{cholesky}(P)$	$\mathrm{d}R = R\,\Phi\big(R^{-1}\mathrm{d}P\,R^{-\top}\big)$	$\mathrm{d}J = \mathrm{Tr}\big(\overline{P}^\top \mathrm{d}P + \widetilde{Y}^\top \mathrm{d}Y\big)$	$\overline{P} = \frac{1}{2}(\mathcal{S} + \mathcal{S}^\top)$ $\mathcal{S} = R^{-\top}\Phi(R^\top\overline{R})R^{-1}$ $\Phi(\cdot) = \cdot - \mathrm{triu}(\cdot) + \frac{1}{2}\mathrm{diag}(\cdot)$
$P = Y^\top Y$	$\mathrm{d}P = \mathrm{d}Y^\top Y + Y^\top \mathrm{d}Y$	$\mathrm{d}J = \mathrm{Tr}\big(\overline{Y}^\top \mathrm{d}Y\big)$	$\overline{Y} = 2\,Y\overline{P} + \widetilde{Y}$

J, then $\overline{A}$ denotes the derivative of J w.r.t. each element of A. From Table III, we deduce that the derivative w.r.t. Y_θ reads

$$\overline{Y}_\theta = 2\,Y_\theta\overline{P} + 2LY_\theta R_\theta^{-T}R_\theta^{-1} \tag{24}$$

where

$$\begin{aligned}\overline{P} &= -\frac{1}{2}R_\theta^{-\top}\big(\Phi(2R_\theta^{-1}DR_\theta^{-T}) + \Phi(2R_\theta^{-1}DR_\theta^{-T})^\top\big)R_\theta^{-1}\\ &= -R_\theta^{-\top}R_\theta^{-1}DR_\theta^{-T}R_\theta^{-1}\\ &= -R_\theta^{-\top}R_\theta^{-1}Y_\theta^\top LY_\theta R_\theta^{-T}R_\theta^{-1}.\end{aligned} \tag{25}$$

By putting together (24) and (25), we arrive at the final expression of the Jacobian given in (19).

REFERENCES

[1] A. Argyriou, M. Herbster, and M. Pontil, "Combining graph laplacians for semi-supervised learning," in *Advances in Neural Information Processing Systems 18*, Y. Weiss, B. Schölkopf, and J. C. Platt, Eds. Montréral, canada: MIT Press, Dec. 2006, pp. 67–74.

[2] L. Tang, X. Wang, and H. Liu, "Community detection via heterogeneous interaction analysis," *Data Mining and Knowledge Discovery*, vol. 25, no. 1, pp. 1–33, Jul. 2012.

[3] P.-Y. Chen and A. O. Hero, "Multilayer spectral graph clustering via convex layer aggregation: Theory and algorithms," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 3, no. 3, pp. 553–567, Sep. 2017.

[4] P. Mercado, A. Gautier, F. Tudisco, and M. Hein, "The power mean laplacian for multilayer graph clustering," in *International Conference on Artificial Intelligence and Statistics*, Playa Blanca, Lanzarote, Canary Islands, Spain, Apr. 2018, pp. 1828–1838.

[5] X. Wang, W. Bian, and D. Tao, "Grassmannian regularized structured multi-view embedding for image classification," *IEEE Transactions on Image Processing*, vol. 22, no. 7, pp. 2646–2660, Jul. 2013.

[6] S. Bickel and T. Scheffer, "Multi-view clustering," in *Fourth IEEE International Conference on Data Mining*, Washington, DC, USA, Nov. 2004, pp. 19–26.

[7] A. Kumar and H. Daumé, "A co-training approach for multi-view spectral clustering," in *Proceedings of International Conference on Machine Learning*, Bellevue, Washington, USA, Jun. 2011, pp. 393–400.

[8] A. Blum and T. Mitchell, "Combining labeled and unlabeled data with co-training," in *Proceedings of the eleventh annual conference on Computational learning theory*, New York, NY, USA, 1998, pp. 92–100.

[9] V. Sindhwani and P. Niyogi, "A co-regularization approach to semi-supervised learning with multiple views," in *Proceedings of the ICML Workshop on Learning with Multiple Views*, Bonn, Germany, Aug. 2005.

[10] H. Wang, Y. Yang, and B. Liu, "GMC: Graph-based multi-view clustering," *IEEE Transactions on Knowledge and Data Engineering (to appear)*, 2019.

[11] X. Dong, P. Frossard, P. Vandergheynst, and N. Nefedov, "Clustering on multi-layer graphs via subspace analysis on grassmann manifolds," *IEEE Transactions on Signal Processing*, vol. 62, no. 4, pp. 905–918, Feb. 2014.

[12] A. Clauset, C. Moore, and M. E. J. Newman, "Hierarchical structure and the prediction of missing links in networks," *Nature*, vol. 453, no. 7191, p. 98, 2008.

[13] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, no. 3, pp. 75–174, Feb. 2010.

[14] J. Kim and J.-G. Lee, "Community detection in multi-layer graphs: A survey," *ACM SIGMOD Record*, vol. 44, no. 3, pp. 37–48, Dec. 2015.

[15] C. W. Loe and H. J. Jensen, "Comparison of communities detection algorithms for multiplex," *Physica A: Statistical Mechanics and its Applications*, vol. 431, p. 2945, 2015.

[16] E. Schaeffer, "Survey: Graph clustering," *Computer Science Review*, vol. 1, no. 1, pp. 27 – 64, Aug. 2007.

[17] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, Aug. 2000.

[18] A. Y. Ng, M. I. Jordan, and W. Yair, "On spectral clustering: Analysis and an algorithm," *Advances in Neural Information Processing Systems 14*, pp. 849–856, 2002.

[19] J. B. MacQueen, "Some methods for classification and analysis of multivariate observations," *Berkeley Symposium on Mathematical Statistics and Probability*, 1967.

[20] Q. Xu, M. Desjardins, and K. Wagstaff, "Constrained spectral clustering under a local proximty structure," *Proceedings of the 18th International Florida Artificial Intelligence Research Society*, May 2005.

[21] X. Wang and I. Davidson, "Flexible constrained spectral clustering," in *International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: ACM, Oct. 2010, pp. 563–572.

[22] M. E. J. Newman, "Finding community structure in networks using the eigenvectors of ma- trices," *Physical Review E*, vol. 74, no. 3, Sep. 2006.

[23] ——, "Modularity and community structure in networks," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 103, no. 23, pp. 8577–8582, Jun. 2006.

[24] M. Saerens, F. Fouss, L. Yen, and P. Dupont, "The principal components analysis of a graph, and its relationships to spectral clustering," in *Proceedings of European Conference on Machine Learning*, Berlin, Heidelberg, 2004, pp. 371–383.

[25] J. B. Kruskal and M. Wish, "Multidimensional scaling," in *Sage Publications, Beverely Hills,California*, 1978.

[26] M. Belkin and N. Partha, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *Advances in Neural Information Processing Systems 14*, T. G. Dietterich, S. Becker, and Z. Ghahramani, Eds. Vancouver CANADA: MIT Press, Dec. 2002, pp. 585–591.

[27] J. B. Tenenbaum, V. De Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[28] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, Dec. 2000.

[29] H. Yang, S. Pan, P. Zhang, L. Chen, D. Lian, and C. Zhang, "Binarized attributed network embedding," in *IEEE International Conference on Data Mining*, Singapore, Nov. 2018, pp. 1476–1481.

[30] X. Shen, S. Pan, W. Liu, Y.-S. Ong, and Q.-S. Sun, "Discrete network embedding," in *International Joint Conference on Artificial Intelligence*, vol. 7, Stockholm, Sweden, Jul. 2018, pp. 3549–3555.

[31] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *ACM SIGKD Dinternational conference on*

Knowledge discovery and data mining, New York, New York, USA, Aug. 2014, pp. 701–710.

[32] W. L. Hamilton, R. Ying, and J. Leskovec, "Representation learning on graphs: Methods and applications," *IEEE Data Engineering Bulletin*, 2017. [Online]. Available: http://arxiv.org/abs/1709.05584

[33] U. Shaham, K. Stanton, H. Li, B. Nadler, R. Basri, and Y. Kluger, "Spectralnet: Spectral clustering using deep neural networks," in *International Conference on Learning Representations*, Vancouver, Canada, May 2018.

[34] A. Bojchevski and S. Günnemann, "Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking," in *International Conference on Learning Representations*, Vancouver, Canada, Apr. 2018.

[35] D. Pfau, S. Petersen, A. Agarwal, D. Barrett, and K. Stachenfeld, "Spectral inference networks: Unifying deep and spectral learning," in *Proc. of ICLR*, 2019.

[36] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *The Semantic Web*, Heraklion, Crete, Greece, Jun. 2018, pp. 593–607.

[37] G. Chierchia, N. Pustelnik, B. Pesquet-Popescu, and J.-C. Pesquet, "A nonlocal structure tensor based approach for multicomponent image recovery problems," *IEEE Trans. Image Proces.*, vol. 23, no. 12, pp. 5531–5544, Dec. 2014.

[38] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *Proc. CVPR*, June 2018.

[39] Y. Tao, Q. Sun, Q. Du, and W. Liu, "Nonlocal neural networks, nonlocal diffusion and nonlocal modeling," in *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., 2018, pp. 496–506.

[40] Y. Lim and M. Pálfia, "Matrix power means and the karcher mean," *Journal of Functional Analysis*, vol. 262, no. 4, pp. 1498–1514, 2012.

[41] P. Zadeh, R. Hosseini, and S. Sra, "Geometric mean metric learning," in *Proc. ICML*, M. F. Balcan and K. Q. Weinberger, Eds., vol. 48, New York, New York, USA, 20–22 Jun 2016, pp. 2464–2471.

[42] P. Mercado, F. Tudisco, and M. Hein, "Clustering signed networks with the geometric mean of laplacians," in *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds., 2016, pp. 4421–4429.

[43] H. Karcher, "Riemannian center of mass and mollifier smoothing," *Communications on pure and applied mathematics*, vol. 30, no. 5, pp. 509–541, Sep. 1977.

[44] H. Zhang and S. Sra, "First-order methods for geodesically convex optimization," *Journal of Machine Learning Research*, vol. 49, pp. 1–22, 2016.

[45] U. V. Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, Dec. 2007.

[46] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, Jun. 2005, pp. 539–546.

[47] K. Q. Weinberger and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," *J. Mach. Learn. Res.*, vol. 10, pp. 207–244, Jun. 2009.

[48] E. Hoffer and N. Ailon, "Deep metric learning using triplet network," in *International Workshop on Similarity-Based Pattern Recognition*, A. Feragen, M. Pelillo, and M. Loog, Eds., 2015, vol. 9370, pp. 84–92.

[49] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," in *IEEE conference on computer vision and pattern recognition*, 2015, pp. 815–823.

[50] M. Bontonou, C. Lassance, G. B. Hacene, V. Gripon, J. Tang, and J. Tang, "Introducing graph smoothness loss for training deep learning architectures," in *IEEE Data Science Workshop*, Jun. 2019, pp. 160–164.

[51] R. Lai and S. Osher, "A splitting method for orthogonality constrained problems," *J. Sci. Comput.*, vol. 58, no. 2, pp. 431–449, Feb. 2014.

[52] M. El Gheche, G. Chierchia, and P. Frossard, "Stochastic gradient descent for spectral embedding with implicit orthogonality constraint," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, Brighton, UK, May 2019, pp. 3567–3571. [Online]. Available: https://arxiv.org/abs/1812.05721

[53] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of adam and beyond," in *International Conference on Learning Representations*, Vancouver, Canada, May 2018.

[54] A. Edelman, T. A. Arias, and S. T. Smith, "The geometry of algorithms with orthogonality constraints," *SIAM J. Matrix Anal. Appl.*, vol. 20, no. 2, pp. 303–353, Apr. 1999.

[55] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proc. ICCV*, Washington, DC, USA, 2015, pp. 1026–1034.

[56] M. Giles, "An extended collection of matrix derivative results for forward and reverse mode automatic differentiation," *Numerical Analysis Report, Oxford University Computing Laboratory*, vol. 08, no. 01, 2008.

[57] I. Murray, "Differentiation of the Cholesky decomposition," *Eprint arXiv:1602.07527*, 2016.