Density-Based Distance Preserving Graph: Theoretical and Practical Analyses

Li Wang, Haian Yin, and Jin Zhang

***Abstract*— This brief aims to provide theoretical guarantee and practical guidance on constructing a type of graphs from input data via distance preserving criterion. Unlike the graphs constructed by other methods, the targeted graphs are hidden through estimating a density function of latent variables such that the pairwise distances in both the input space and the latent space are retained, and they have been successfully applied to various learning scenarios. However, previous work heuristically treated the multipliers in the dual as the graph weights, so the interpretation of this graph from a theoretical perspective is still missing. In this brief, we fill up this gap by presenting a detailed interpretation based on optimality conditions and their connections to neighborhood graphs. We further provide a systematic way to set up proper hyperparameters to prevent trivial graphs and achieve varied levels of sparsity. Three extensions are explored to leverage different measure functions, refine/reweigh an initial graph, and reduce computation cost for medium-sized graph. Extensive experiments on both synthetic and real datasets were conducted and experimental results verify our theoretical findings and the showcase of the studied graph in semisupervised learning provides competitive results to those of compared methods with their best graph.**

***Index Terms*— Density estimation, distance preservation, graph construction and learning, sparse graph.**

I. Introduction

Graphs as ubiquitous and informative structures have been extensively employed to represent data. They play crucially important roles in approximating intrinsic manifolds of data in feature extraction [1], [2], semisupervised learning [3], and clustering [4]. However, graphs are often unknown and improper graphs can degrade the learning performance, so learning proper graphs from input data is important.

For a given measure (distance, dissimilarity, or similarity) function over any two data points, a fully connected weighted graph can be directly constructed from a set of data points, such as Gaussian kernel [3] and Pearson's correlation coefficient [5]. Although the fully connected weighted graphs have demonstrated great success in many learning problems, sparse graphs are often more preferred due to their robustness to data noise, efficiency, and interpretability [6].

The majority of graph construction and learning methods concentrate on generating a sparse graph from data. Based on the existence of graph elements such as graph topology (node connectivity) and edge weights, we can roughly classify the existing graph construction methods into three categories: 1) generating a sparse graph topology; 2) estimating edge weights of a given graph topology; and 3) learning both graph topology and edge weights.

In this brief, we are particularly interested in the latent sparse graph constructed as a byproduct by maximum posterior manifold embedding (MPME) [7] for dimensionality reduction. MPME was originally proposed to learn the posterior density function of embedded points in a latent space from input data such that the pairwise distances in both the input space and the latent space are retained. Although MPME does not explicitly contain a graph as an optimized variable, its dual problem is formulated with multipliers that are heuristically regarded as the graph weights. We name this special hidden graph as density-based distance preserving graph (DDPG). Hence, MPME as a graph construction method belongs to category 3), and its resulting DDPG is equipped with various unique properties compared to graphs constructed by other methods.

1) Different from other graphs, DDPG is hidden to preserve the pairwise distances.
2) DDPG is flexible to be learned from data in terms of various distance measures, while most existing methods, e.g., local linear reconstruction (LLR)-related models [8]–[10], are inherently modeled in the Euclidean space.
3) DDPG can be learned by MPME to naturally incorporate any prior distribution of latent variables.
4) The dual problem with respect to DDPG is strongly convex, so the global optimal graph is obtainable.
5) MPME and its variants have been successfully used for various learning problems, including dimensionality reduction [7], clustering [11], and feature selection [12].

Although DDPG enjoys the above-mentioned good properties, it is heuristically regarded as a similarity graph, so the interpretation of DDPG as a graph to approximate the intrinsic manifold of input data is still missing. The lack of deep understanding prevents DDPG from being used for other learning problems, such as semisupervised learning. Moreover, the sparsity of DDPG is simultaneously controlled by multiple hyperparameters in MPME, which makes constructing DDPG from input data impractical from model selection perspective, especially for medium-size data. In this brief, we aim to provide the theoretical analysis of DDPG, simplify the model selection process, and further explore three extensions of DDPG constructed from different settings. The contributions of this brief are summarized as follows.

1) We provide the interpretation of DDPG as a similarity graph from the perspective of optimality conditions for the primal and dual problems of MPME and build the connections to K-nearest neighbor (K-NN) graphs and uncover ϵ-neighborhood (ϵ-N) graph as the extreme case.
2) We conduct hyperparameter analyses, including the prevention of trivial solutions, the impact of graph sparsity, and reparameterizing parameters for simplification. With the help of these analyses, we can safely tune a single hyperparameter instead of three ones to reach different levels of sparsity in DDPG.
3) Three extensions for learning DDPG are explored, including various measure functions, graph refinement, and scalability consideration.

Extensive experiments on both synthetic and real datasets are conducted to verify that: 1) the different levels of sparsity in DDPG can be controllable by a single hyperparameter; 2) DDPG can be used to refine a given graph or speed up the learning with a precomputed graph; and 3) DDPG used in graph-based semisupervised learning can achieve competitive results against the best graphs learned by compared methods.

Manuscript received January 15, 2021; revised July 25, 2021; accepted October 23, 2021. The work of Li Wang was supported in part by NSF under Grant DMS-2009689. The work of Jin Zhang was supported in part by NSFC under Grant 11971220, in part by the Guangdong Basic and Applied Basic Research Foundation under Grant 2019A1515011152, and in part by the Shenzhen Science and Technology Program under Grant RCYX20200714114700072. *(Corresponding authors: Li Wang; Jin Zhang.)*

Li Wang is with the Department of Mathematics and the Department of Computer Science and Engineering, University of Texas at Arlington, Arlington, TX 76019 USA (e-mail: li.wang@uta.edu).

Haian Yin and Jin Zhang are with the SUSTech International Center for Mathematics and the Department of Mathematics, Southern University of Science and Technology, Shenzhen 518055, China (e-mail: 11930905@mail.sustech.edu.cn; zhangj9@sustech.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2021.3123089.

Digital Object Identifier 10.1109/TNNLS.2021.3123089

The rest of this brief is organized as follows. We first briefly review various existing graph construction methods in Section II and then introduce MPME in detail as a graph construction method in Section III. The interpretation of DDPG and its connections to neighborhood graphs are presented in Section IV. In Section V, we conduct a detailed analysis of hyperparameters and further simplify them into one. Extensions are shown in Section VI. Extensive experiments are conducted in Section VII. We conclude this work in Section VIII.

II. Related Work

Existing sparse graph construction methods can be roughly divided into three categories based on the existence of graph elements, i.e., graph topology and edge weights.

A. Generating a Sparse Graph Topology

Euclidean distance is often used to construct the K-NN graph and ϵ-N graph [1]. Both types of graphs are studied in terms of the influence on graph-based clustering [13]. However, ϵ-N graphs are less used in label propagation methods due to its deteriorating performance since it has a tendency to fragment the data into many disconnected components [14]. Due to the greedy construction process of connecting K closest points to one point, the K-NN graph is formally directed. Postprocessing steps, such as mutual K-NN, symmetric K-NN, and symmetry-favored K-NN, are generally used [15]. The b-matching (BM) method [6] can directly form a regular graph with each node having b neighbors. Several other approaches have been reviewed in [16].

B. Estimating Edge Weights of a Given Graph Topology

Given a graph topology, edge reweighing is more often used to capture the varying similarity among different pairs of nodes. Certain measure functions can be used to define the edge weights such as the Gaussian kernel function [1] and the inverse of distance [17], which can be different from the measure function used to learn the topology. In addition to the predefined weight functions, learning edge weights from data for a given graph topology has also been explored in the literature [8]–[10]. LLR [8] aims to reconstruct each input data point via a linear combination of its neighbors, and the combination coefficients are used as graph weights. To prevent negative weights, the nonnegative weights are enforced in LLR [9]. Parametric weighting functions, such as automatic relevance determination kernel, are also explored in LLR [10].

C. Learning Both Graph Topology and Edge Weights

Learning a weighted graph from data is generally formed as the problem of optimizing a sparse matrix. The ℓ_1 norm on the graph is popularly used to promote the sparsity of the learned graph including sparse manifold clustering and embedding [18] and ℓ_1-graph [19]. As both methods are similar to ℓ_1 regularized LLR, the learned graphs are directed and their weights can be negative, so additional postprocessing is required to convert them to undirected similarity graphs. Learning an undirected weighted graph from data has also been explored [20]–[24]. In [20], a sparse precision matrix is optimized by maximizing the likelihood of a Gaussian distribution with ℓ_1 norm over the precision matrix. In [21], the precision matrix in the Gaussian–Markov random fields is parameterized as a Laplacian matrix on the graph of data points. Since graph Laplacian and the graph matrix share the same sparse patterns, a sparse graph Laplacian matrix is learned in [22]. Moreover, several special types of graphs were studied, including spanning trees [23], low-rank graph [24], and DDPG for preserving pairwise distances [7], [11], [12]. Learning graphs for graph neural networks is also studied [25], [26]. A graph generator with Bernoulli variables [25] is optimized specifically for graph convolutional network, while the inference is done by randomly drawing a sufficient number of graphs from the generator. Due to the nonlinear transformations, there is no deterministic graph to be obtained and the sparsities of these sampled graphs are not controllable. The graph parameterized by the Mahalanobis distance function [26] is dense and only applicable to graph data.

III. MPME as a Graph Construction Approach

As discussed in Section I, DDPG learned by MPME has various advantages compared to graphs by existing methods. In MPME [7], preserving pairwise distances of input data in a latent space has been explored to learn an undirected weighted graph for unsupervised dimensionality reduction. In order to conduct theoretical and practical analyses on DDPG, we need to introduce MPME in detail.

Let $\{(\mathbf{x}_i)\}_{i=1}^n$ be the n input data points with $\mathbf{x}_i \in \mathbb{R}^d$. Define by $X = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{d\times n}$ the input dataset. MPME looks for a density function over latent representations of the input data by preserving pairwise distances of input data in a latent space. The latent representations of the input data X are represented by a matrix of random variables $Z = [z_{r,i}] \in \mathbb{R}^{m\times n}, r = 1, \ldots, m, i = 1, \ldots, n$, following some unknown density $p(Z)$, where m is the dimension of the latent space. Define by $\mathbf{f}_r = [z_{r,1}, \ldots, z_{r,n}]^{\mathrm{T}} \in \mathbb{R}^n$ and $\mathbf{z}_i = [z_{1,i}, \ldots, z_{m,i}]^{\mathrm{T}} \in \mathbb{R}^m$, the rth row of Z and the ith column of Z, respectively. Thus, we have the following relationships $Z = [\mathbf{z}_1, \ldots, \mathbf{z}_n] = [\mathbf{f}_1, \ldots, \mathbf{f}_m]^{\mathrm{T}}$, and $\mathbf{z}_i$ is the latent counterpart to $\mathbf{x}_i, \forall i = 1, \ldots, n$. As $\{\mathbf{f}_r\}_{r=1}^m$ are the bases of the latent representations, independence is often assumed, that is, $p(Z) = \prod_{r=1}^m p(\mathbf{f}_r)$. The pairwise distance of two random variables $\mathbf{z}_i$ and $\mathbf{z}_j$ can be calculated as the expectation of their Euclidean distance over $p(Z)$, given by, $\forall i, j = 1, \ldots, n$

$$\mathbb{E}\left[\|\mathbf{z}_i - \mathbf{z}_j\|^2\right] = \sum_{r=1}^{m} \int \left(z_{r,i} - z_{r,j}\right)^2 p(\mathbf{f}_r) \mathrm{d}\mathbf{f}_r. \tag{1}$$

The density-based pairwise distance takes the density $p(Z)$ into account in the framework of Bayesian averaging [27], so it is more robust to data noise and outliers compared to the deterministic approach. The pairwise distance in a latent space is analogous to the distance of its counterpart in the input space, for example, the Euclidean distance

$$\phi_{i,j} = \|\mathbf{x}_i - \mathbf{x}_j\| \quad \forall i, \ j = 1, \ldots, n. \tag{2}$$

Given a prior distribution $\pi(Z) = \prod_{r=1}^m \pi(\mathbf{f}_r)$, MPME aims to obtain the optimal $p(Z)$ by minimizing Kullback–Leibler (KL) divergence between $p(Z)$ and its prior $\pi(Z)$ under the constraints of preserving the pairwise distance but allowing a small violation

$$\begin{aligned} \min_{p(Z),\{\zeta_{i,j}\}} \ & \lambda \mathrm{KL}(p(Z)||\pi(Z)) + \sum_{i=1}^{n}\sum_{j=1}^{n} \zeta_{i,j} \\ \text{s.t. } & \mathbb{E}\left[\|\mathbf{z}_i - \mathbf{z}_j\|^2\right] \le \phi_{i,j} + \zeta_{i,j}, \ \zeta_{i,j} \ge 0 \quad \forall i, j \end{aligned} \tag{3}$$

where λ is the regularization parameter and the KL divergence is further written as $\mathrm{KL}(p(Z)||\pi(Z)) = \sum_{r=1}^m \int p(\mathbf{f}_r) \log(p(\mathbf{f}_r)/\pi(\mathbf{f}_r)) \mathrm{d}\mathbf{f}_r$. Apparently, it is difficult to directly solve (3) with respect to $p(Z)$, so we seek its dual problem according to its convexity.

By introducing multipliers $\{\alpha_{i,j} \ge 0\}$ and $\{\tau_{i,j} \ge 0\}$, the Lagrangian function is

$$\begin{aligned} & g\left(\{p(\mathbf{f}_r)\}, \{\zeta_{i,j}\}, \{\alpha_{i,j}\}, \{\tau_{i,j}\}\right) \\ & = \lambda \sum_{r=1}^{m} \int p(\mathbf{f}_r) \log \frac{p(\mathbf{f}_r)}{\pi(\mathbf{f}_r)} \mathrm{d}\mathbf{f}_r + \sum_{i,j} \zeta_{i,j} - \sum_{i,j} \alpha_{i,j} \tau_{i,j} \\ & \quad + \sum_{i,j} \alpha_{i,j} \left(\sum_{r=1}^{m} \int \left(z_{r,i} - z_{r,j}\right)^2 p(\mathbf{f}_r) \mathrm{d}\mathbf{f}_r - \|\mathbf{x}_i - \mathbf{x}_j\|^2 - \zeta_{i,j} \right). \end{aligned}$$

We have the following Karush–Kuhn–Tucker (KKT) conditions:

$$\lambda(1+\log p(\mathbf{f}_r)-\log\pi(\mathbf{f}_r))+\sum_{i,j}\alpha_{i,j}\left(z_{r,i}-z_{r,j}\right)^2=0\quad\forall r=1,\ldots,m, \tag{4}$$

$$1-\alpha_{i,j}-\tau_{i,j}=0\quad\forall i,\ j=1,\ldots,n, \tag{5}$$

$$\tau_{i,j}\zeta_{i,j}=0\quad\forall i,\ j=1,\ldots,n, \tag{6}$$

$$\alpha_{i,j}\left(\mathbb{E}\left[\|\mathbf{z}_i-\mathbf{z}_j\|^2\right]-\phi_{i,j}-\zeta_{i,j}\right)=0\quad\forall i,\ j=1,\ldots,n, \tag{7}$$

$$\mathbb{E}\left[\|\mathbf{z}_i-\mathbf{z}_j\|^2\right]\le\phi_{i,j}+\zeta_{i,j},\ \zeta_{i,j}\ge 0\quad\forall i,\ j=1,\ldots,n. \tag{8}$$

According to (4), we have

$$p(\mathbf{f}_r)\propto\pi(\mathbf{f}_r)\exp\left(-\tfrac{2}{\lambda}\mathbf{f}_r^{\mathrm{T}}L\mathbf{f}_r\right)\quad\forall r=1,\ldots,m \tag{9}$$

where graph Laplacian $L=\mathrm{diag}(A1)-A$ and $A=[\alpha_{i,j}]\in\mathbb{R}^{n\times n}$ is the matrix representation of the learned undirected weighted graph. The dual problem is

$$\max_{A\in\mathcal{A}}-\lambda\sum_{r=1}^{m}u_r-\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_{i,j}\phi_{i,j} \tag{10}$$

where the log partition term is

$$u_r=\log\left\{\int\pi(\mathbf{f}_r)\exp\left(-\frac{2}{\lambda}\mathbf{f}_r^{\mathrm{T}}L\mathbf{f}_r\right)\mathrm{d}\mathbf{f}_r\right\}\quad\forall r \tag{11}$$

and the feasible set of A is

$$\mathcal{A}=\left\{[\alpha_{i,j}]|A=A^{\mathrm{T}},0\le\alpha_{i,j}\le 1,\alpha_{i,i}=0,\forall i,j\right\}. \tag{12}$$

Note that $\phi_{i,i}=0$ and L is irrelevant to the diagonal part of A, so we set $\alpha_{i,i}=0,\forall i=1,\ldots,n$. This means that the self-connected edges of the learned graph are ignored. Moreover, the pairwise distance constraints are symmetric, so the matrix A of multipliers is symmetric too.

Specifically, we assume that $\pi(\mathbf{f}_r)$ is a normal distribution with mean 0 and covariance $\sigma^2 I_n$. The log partition term has an explicit expression as

$$u_r=-\frac{1}{2}\log\det\left(\frac{1}{\sigma^2}I_n+\frac{4}{\lambda}L\right) \tag{13}$$

and multivariate normal distribution according to (9)

$$p(\mathbf{f}_r)\sim\mathcal{N}\left(0,\left(\frac{1}{\sigma^2}I_n+\frac{4}{\lambda}L\right)^{-1}\right). \tag{14}$$

In MPME [7], A is heuristically regarded as a graph, which is obtained by solving convex problem (10) with (13) using the L-BFGS-B method [28]. The latent embeddings of the input data are then obtained by the maximum posterior estimation.

According to [7], solving problem (10) takes approximately $O(n^{2.37})$ for computing logdet and an inversion of matrix $(1/\sigma^2)I_n+(4/\lambda)L$ at each iteration of the L-BFGS-B solver. Kernel principal component analysis (KPCA) for obtaining the latent embeddings takes $O(n^3)$. Thus, the time complexity of MPME takes $O(n^3)$.

IV. Graph Interpretation

The interpretation of the learned latent graph in (10) is not well explored in the existing work [7]. To fill up this gap, we will provide some explanation for the learned graph from various perspectives, including the optimality conditions and the relationships to neighborhood graphs.

A. Optimality Conditions

According to KKT conditions (4)–(8), we can study the relationships between the pairwise distance $\phi_{i,j}$ and its latent counterpart $\mathbb{E}[\|\mathbf{z}_i-\mathbf{z}_j\|^2]$ in terms of the optimal $\alpha_{i,j}$. As $\tau_{i,j}\ge 0$, we consider the following four cases.

1) $\tau_{i,j}\in(0,1)$. According to (5), $\alpha_{i,j}\in(0,1)$. According to (6), $\zeta_{i,j}=0$. Combining with (7), we have $\mathbb{E}[\|\mathbf{z}_i-\mathbf{z}_j\|^2]=\phi_{i,j}$.
2) $\tau_{i,j}=0$. According to (5), $\alpha_{i,j}=1$. According to (6), we have $\mathbb{E}[\|\mathbf{z}_i-\mathbf{z}_j\|^2]-\phi_{i,j}-\zeta_{i,j}=0$. Since $\zeta_{i,j}\ge 0$, thus, $\mathbb{E}[\|\mathbf{z}_i-\mathbf{z}_j\|^2]\ge\phi_{i,j}$.
3) $\tau_{i,j}=1$. According to (5), $\alpha_{i,j}=0$. According to (6), $\zeta_{i,j}=0$. Thus, we have $\alpha_{i,j}(\mathbb{E}[\|\mathbf{z}_i-\mathbf{z}_j\|^2]-\phi_{i,j})=0$. As the constraint $\mathbb{E}[\|\mathbf{z}_i-\mathbf{z}_j\|^2]\le\phi_{i,j}+\zeta_{i,j}$ is required, and we have $\mathbb{E}[\|\mathbf{z}_i-\mathbf{z}_j\|^2]\le\phi_{i,j}$.
4) $\tau_{i,j}>1$. According to (5), $\tau_{i,j}=1-\alpha_{i,j}\le 1$, which leads to a contradiction.

In summary, we have the following optimality conditions:

$$\phi_{i,j}-\mathbb{E}\left[\|\mathbf{z}_i-\mathbf{z}_j\|^2\right]\begin{cases}\ge 0, & \alpha_{i,j}=0\\ =0, & 0<\alpha_{i,j}<1\\ \le 0, & \alpha_{i,j}=1.\end{cases} \tag{15}$$

As a result, we have recovered the relationships between the pairwise distances in latent space and their corresponding pairwise dissimilarities of the input data points.

B. Multipliers Interpreted as Pairwise Similarity

According to (10), the objective term

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_{i,j}\phi_{i,j}=\sum_{(i,j):\alpha_{i,j}\ne 0}\alpha_{i,j}\phi_{i,j} \tag{16}$$

is minimized. At the optimum, $\alpha_{i,j}$ is expected to be smaller if $\phi_{i,j}$ is bigger. Thus, $\alpha_{i,j}$ can be interpreted as certain similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$ when $\alpha_{i,j}\ne 0$ and $\phi_{i,j}\le\mathbb{E}[\|\mathbf{z}_i-\mathbf{z}_j\|^2]$. For $\alpha_{i,j}=0$, we have $\phi_{i,j}>\mathbb{E}[\|\mathbf{z}_i-\mathbf{z}_j\|^2]$ from the optimality condition (15). Combining all together, we can see that A is a similarity matrix to characterize the pairwise relations of input data points and $\alpha_{i,j}=0$ for the two input points of large distance. However, this interpretation is not strictly true since each $\alpha_{i,j}$ reaches its optimum based on the rest of others since they are coupled in (10) in the logdet term. In the following, we will explore the relationships of the graph matrix A learned by model (10) to two popularly used graphs: ϵ-N graph and K-NN graph.

C. Connection to the K-NN Graph

In [6], BM graph can be considered as a proper replacement of the K-NN graph for semi-supervised learning (SSL). It solves the following integer programming:

$$\min_{\{g_{i,j}\in\{0,1\}}}\sum_{i=1}^{n}\sum_{j=1}^{n}g_{i,i}\phi_{i,j}\quad\text{s.t.}\ \sum_{j=1}^{n}g_{i,j}=b,g_{i,i}=0,g_{i,j}=g_{j,i}\quad\forall i,j \tag{17}$$

where integer b is a parameter. In the following, we uncover its connection to our proposed model (10).

With the normal prior and (13), problem (10) can be rewritten as

$$\min_{A\in\mathcal{A}}-\frac{m\lambda}{2}\log\det\left(\frac{1}{\sigma^2}I_n+\frac{4}{\lambda}L\right)+\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_{i,j}\phi_{i,j}. \tag{18}$$

Suppose that $\sum_{j=1}^{n}\alpha_{i,j}=b$. Thus, we have $L=bI-A$ and

$$\begin{aligned}&\log\det\left(\frac{1}{\sigma^2}I_n+\frac{4}{\lambda}L\right)\\&\approx n\log\frac{4}{\lambda}+\log\det\left(\left(\frac{\lambda}{4\sigma^2}+b\right)I_n\right)-\mathrm{tr}\left(\left(\frac{\lambda}{4\sigma^2}+b\right)^{-1}A\right)\\&=n\log\frac{4}{\lambda}+n\log\left(\frac{\lambda}{4\sigma^2}+b\right)\end{aligned}$$

where the first-order approximation of logdet with respect to A at $A = 0$ is used [29]. Since the self-connected edges are ignored by setting $\alpha_{i,i} = 0, \forall i = 1, \ldots, n$, $\mathrm{tr}(A) = 0$. With the above approximation of logdet term, our model (18) reduces to

$$\begin{aligned} &\min_{A} \ \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_{i,j}\phi_{i,j} \\ &\text{s.t.} \ \sum_{j=1}^{n} \alpha_{i,j} = b, \quad 0 \le \alpha_{i,j} \le 1, \ \alpha_{i,j} = \alpha_{j,i}, \ \alpha_{i,i} = 0 \quad \forall i, j. \end{aligned} \tag{19}$$

As a result, the approximate solution of (18) is a relaxation of the BM problem (17). On the other hand, the equivalence requires special parameters m, σ, and λ such that

$$-\frac{m\lambda}{2}\log\det\left(\frac{1}{\sigma^2}I_n + \frac{4}{\lambda}L\right) = \begin{cases} \text{const}, & \sum_{j=1}^{n}\alpha_{i,j} = b \quad \forall i \\ \infty, & \text{otherwise} \end{cases}$$

with the constraints in (18).

D. Extreme Case: The ϵ-N Graph

Let us consider an extreme case $\lambda = \infty$ for applying the above optimality conditions. According to (9), the density $p(\mathbf{f}_r) = \pi(\mathbf{f}_r)$. Suppose that the prior $\pi(\mathbf{f}_r)$ is a normal distribution with mean 0 and covariance $\sigma^2 I_n$. We can compute the pairwise distance of latent variables $\mathbf{z}_i$ and $\mathbf{z}_j$ as

$$\begin{aligned} \mathbb{E}\big[\|\mathbf{z}_i - \mathbf{z}_j\|^2\big] &= \mathbb{E}\big[\|\mathbf{z}_i\|^2\big] + \mathbb{E}\big[\|\mathbf{z}_j\|^2\big] - 2\mathbb{E}\big[\mathbf{z}_i^{\mathrm{T}}\mathbf{z}_j\big] \\ &= \sum_{r=1}^{m}\big(\mathbb{E}\big[z_{r,i}^2\big] + \mathbb{E}\big[z_{r,j}^2\big] - 2\mathbb{E}\big[z_{r,i}z_{r,j}\big]\big) \\ &= 2m\sigma^2. \end{aligned}$$

Now, we have the following rule to construct the graph A:

$$\alpha_{i,j} = 0 \quad \text{if } \phi_{i,j} > 2m\sigma^2 \quad \text{else } \alpha_{i,j} \neq 0. \tag{20}$$

Let $\epsilon = \sqrt{2m}\sigma$. Equation (20) is analogous to the ϵ-N graph, which means that if the pairwise distance of two input data is larger than ϵ, the edge weight is set to zero. The two graphs become equivalent if $\alpha_{i,j} = 1$ for all $\alpha_{i,j} \neq 0$. However, the covariance of $p(\mathbf{f}_r)$ is not simply an identity matrix for $\lambda > 0$, so the learned graph via (10) can be very different from the ϵ-N graph, i.e., the extreme case $\lambda = \infty$.

V. Hyperparameter Analysis

Denote $Q = (1/\sigma^2)I_n + (4/\lambda)L$. The dual problem (18) can be rewritten as a convex optimization with box constraints

$$\min_{0\le\mathbf{a}\le 1} g(\mathbf{a}) := -\frac{m\lambda}{2}\log\det(Q) + \sum_{i,j}\alpha_{i,j}\phi_{i,j} \tag{21}$$

where $\mathbf{a}$ is a vector of upper triangular part of A, e.g., $\{\alpha_{i,j} : \forall i, j > i\}$ due to the symmetry of A and $\alpha_{i,i} = 0, \forall i$. As $g(\mathbf{a})$ is continuously differentiable, for the global minimizer $\mathbf{a}^*$ of (21), we have the following optimality conditions [30]:

$$\frac{\partial g(\mathbf{a}^*)}{\partial \alpha_{i,j}} \begin{cases} \ge 0, & \alpha_{i,j}^* = 0 \\ = 0, & 0 < \alpha_{i,j}^* < 1 \\ \le 0, & \alpha_{i,j}^* = 1. \end{cases} \quad \forall i, \ j > i \tag{22}$$

According to the optimality conditions (22), we analyze the behaviors of hyperparameters in (18) for the practical use.

A. Prevention of a Trivial Solution

Intuitively, $\mathbf{a}^* = 0$ is a trivial solution since all nodes of the graph are disconnected. From (22), this trivial solution can be obtained in the condition of

$$\frac{\partial g(\mathbf{0})}{\partial \alpha_{i,j}} \ge 0 \quad \forall i, \ j > i. \tag{23}$$

To prevent the trivial solution, we can impose the following condition that:

$$\exists (i, j > i), \quad \frac{\partial g(0)}{\partial \alpha_{i,j}} < 0. \tag{24}$$

The gradient of $g(\mathbf{a})$ with respect to $\alpha_{i,j}$ can be written as

$$\begin{aligned} \frac{\partial g(\mathbf{a})}{\partial \alpha_{i,j}} &= \mathrm{tr}\left(\left\{-\frac{m\lambda}{2}Q^{-1}\right\}\frac{4}{\lambda}S_{i,j}\right) + 2\phi_{i,j} \\ &= 2\phi_{i,j} - \left[U_{i,i} + U_{j,j} - U_{i,j} - U_{j,i}\right] \quad \forall i, \ j > i \end{aligned} \tag{25}$$

where

$$S_{i,j}(s,t) = \begin{cases} 1, & s = i = j = t \\ -1, & s = i \neq j = t \\ 0, & \text{otherwise} \end{cases} \quad \forall, s, t \tag{26}$$

$$U = 2mQ^{-1}. \tag{27}$$

We have $g(\mathbf{0}) = 2\phi_{i,j} - 4m\sigma^2$ since $Q^{-1} = \sigma^2 I_n$ at $A = 0$. The condition (24) becomes

$$\exists (i, j > i), \quad 2\phi_{i,j} - 4m\sigma^2 < 0. \tag{28}$$

Fortunately, this condition can be easily satisfied by letting

$$\sigma > \sqrt{\frac{\min_{i,j}\phi_{i,j}}{2m}}. \tag{29}$$

This is consistent with the analysis in Section IV-D for the ϵ-N graph with $\epsilon = \sqrt{2m}\sigma$. If $\epsilon = \min_{i,j}\sqrt{\phi_{i,j}}$, the ϵ-N graph has no edges, that is $A = 0$.

According to (14), we have

$$\begin{aligned} \mathbb{E}\big[\|\mathbf{z}_i - \mathbf{z}_j\|^2\big] &= m\big(Q^{-1}(i,i) + Q^{-1}(j,j) - 2Q^{-1}(i,j)\big) \\ &= \frac{1}{2}\mathrm{tr}\big(US_{i,j}\big) \end{aligned}$$

which is different from the extreme case ϵ-N. Hence, we have

$$\frac{\partial g(\mathbf{a}^*)}{\partial \alpha_{i,j}} = 2\phi_{i,j} - 2\mathbb{E}\big[\|\mathbf{z}_i - \mathbf{z}_j\|^2\big] \tag{30}$$

for the optimal A and $p(Z)$. This verifies that the optimal conditions (22) and (15) are equivalent.

B. Impact on Graph Sparsity

Another important factor is the sparsity of A. According to (22), the number of zeros depends on the number of positive gradients at optimum. To investigate the impact of hyperparameters on the sparsity of A, we further rewrite the gradient

$$\begin{aligned} \frac{\partial g(\mathbf{a})}{\partial \alpha_{i,j}} &= 2\phi_{i,j} - 2m\mathbf{e}_{i,j}^{\mathrm{T}}Q^{-1}\mathbf{e}_{i,j} \\ &= 2\phi_{i,j} - 2m\mathbf{e}_{i,j}^{\mathrm{T}}V\,\mathrm{diag}\left(\left[\frac{1}{\frac{1}{\sigma^2} + \frac{4}{\lambda}\gamma_i}\right]\right)V^{\mathrm{T}}\mathbf{e}_{i,j} \end{aligned}$$

where $\mathbf{e}_{i,j} \in \mathbb{R}^n$ with $\mathbf{e}_{i,j}(i) = 1$ and $\mathbf{e}_{i,j}(j) = -1$ and 0 for other entries, and $L = V\Gamma V^{\mathrm{T}}$ are eigenvalue decomposition with eigenvalues $\Gamma = \mathrm{diag}(\gamma_1, \ldots, \gamma_n)$ and eigenvectors as the columns of V. Denote $V^{\mathrm{T}} = [v_{k,i}] = [\mathbf{v}_1, \ldots, \mathbf{v}_n]$, and we have $V^{\mathrm{T}}\mathbf{e}_{i,j} = \mathbf{v}_i - \mathbf{v}_j$. As a result, we have

$$\frac{\partial g(\mathbf{a})}{\partial \alpha_{i,j}} = 2\phi_{i,j} - 2m\sum_{k=1}^{n}\frac{1}{\frac{1}{\sigma^2} + \frac{4}{\lambda}\gamma_k}\big(v_{k,i} - v_{k,j}\big)^2. \tag{31}$$

This implies that the smaller m, σ, or λ is, the sparser the A is. However, they control the graph sparsity at different levels. Our graph Laplacian matrix L is analogous to the regularized graph Laplacian, which has also been explored in [31].

C. Reparameterization

To facilitate the setup of the hyperparameters λ and σ, we introduce two parameters $\phi > 0$ and $\lambda_* > 0$ such that

$$\sigma^2 = \frac{\phi}{2m}, \quad \lambda = 4\sigma^2\lambda_* = \frac{2\phi\lambda_*}{m}.$$

According to (29), we have

$$\phi > \min_{i,j} \phi_{i,j}. \tag{32}$$

With this setup, we can rewrite the weighting

$$\frac{2m}{\frac{1}{\sigma^2} + \frac{4}{\lambda}\gamma_k} = \frac{\phi}{1 + \frac{\gamma_k}{\lambda_*}}. \tag{33}$$

Given an m, the smaller ϕ and λ leads to sparser A. In general, we can set a large ϕ, such as the maximum value of $\{\phi_{i,j}\}$, and then tune λ_* to get different levels of sparsity. And σ^2 is the variance of the prior normal distribution, so it is preferred to be large, so that the density value on the real space is more flat or uniform. This is preferred for the data without strong assumptions available. Given ϕ, m can be large to allow a smaller σ^2. Suppose that $\sigma^2 = 10^6$, we have $m = (\phi/2\sigma^2)$ and $\lambda = 4\sigma^2\lambda_*$. In this setup, λ_* is the only hyperparameter for DDPG learning. We will use and verify this hyperparameter setup throughout all our experiments in Section VII.

VI. Extensions

In this section, we will explore three simple extensions of (3) for DDPG construction based on the theoretical analyses in Section IV, including: 1) the choice of distance metric to be preserved; 2) the graph refinement; and 3) scalability concern.

A. Transformation of Pairwise Dissimilarity to a Metric

Model (3) is built on pairwise dissimilarities $\{\phi_{i,j}\}$ derived from the input data X and transforms them to the expectation of pairwise Euclidean distance of two random variables. The Euclidean distance (2) is used in [7]. Any dissimilarity function between two input data can be used. Some dissimilarity functions used for various purposes are showcased.

1) Minkowski distance with parameter $p \geq 1$ is defined as $\phi_{i,j} = (\sum_{r=1}^{d} |x_{r,i} - x_{r,j}|^p)^{2/p}$, where $p = 2$ for the Euclidean distance.
2) Kernel distance. Let $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \langle \psi(\mathbf{x}_i), \psi(\mathbf{x}_j) \rangle_{\mathcal{H}}$ be the kernel function over $\mathbf{x}_i$ and $\mathbf{x}_j$ with induced map $\psi(\mathbf{x}_i)$ and $\psi(\mathbf{x}_j)$ in the reproducing kernel Hilbert space $\mathcal{H}$. The Euclidean distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ in the space $\mathcal{H}$ is defined as $\phi_{i,j} = \|\psi(\mathbf{x}_i) - \psi(\mathbf{x}_j)\|_{\mathcal{H}}^2 = \kappa(\mathbf{x}_i, \mathbf{x}_i) + \kappa(\mathbf{x}_j, \mathbf{x}_j) - 2\kappa(\mathbf{x}_i, \mathbf{x}_j)$. This metric is the key in maximum variance unfolding (MVU) [32] for nonlinear kernel learning in dimensionality reduction.
3) Cosine distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ is defined as $\phi_{i,j} = 1 - (\mathbf{x}_i^{\mathrm{T}}\mathbf{x}_j / (\|x_i\|_2 \|x_j\|_2))$, which is often used for text data.
4) χ^2 distance is useful to compute the distance between two histograms, defined as $\phi_{i,j} = \sum_{r=1}^{d} ((\mathbf{x}_i(r) - \mathbf{x}_j(r))^2 / (x_i(r) + x_j(r)))$, which is a special measure for histogram data.

It is worth noting that the model (3) takes $\phi_{i,j}$ as the input and output of a weighted graph, so it is not related to how these dissimilarities are computed. More complicated measures, e.g., geodesic distance [33] and conditional probability [34] involving a significant amount of computations can be used to learn DDPG, but they are infeasible to be used in other graph learning methods such as LLR [8]. Given a dissimilarity function, K-NN, BM, and ϵ-N can be constructed, but their outputs are connectivity graph, so they need additional reweighting step with the choice of weighting function to get a weighted graph.

B. Graph Refinement

Given an initial graph G and its node features X, we can refine the graph and its edge weights by learning a new weighted graph. Let $\mathcal{N}_i$ be the index set of the neighbors of node i in G. The extension of model (3) to include an initial graph is formulated as

$$\begin{aligned} \min_{p(Z), \{\zeta_{i,j}\}} \ & \lambda \mathrm{KL}(p(Z) || \pi(Z)) + \sum_{i, j \in \mathcal{N}_i} \zeta_{i,j} \\ \text{s.t. } & \mathbb{E}\left[\|\mathbf{z}_i - \mathbf{z}_j\|^2\right] \leq \phi_{i,j} + \zeta_{i,j}, \ \zeta_{i,j} \geq 0 \quad \forall i, \ j \in \mathcal{N}_i. \end{aligned} \tag{34}$$

With the normal prior and (13), the dual problem of (34) can be rewritten as

$$\begin{aligned} \min_{A} & -\frac{m\lambda}{2} \log\det\left(\frac{1}{\sigma^2} I_n + \frac{4}{\lambda} L\right) + \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_{i,j}\phi_{i,j} \\ \text{s.t. } & 0 \leq \alpha_{i,j} \leq 1 \quad \forall i, \ j \in \mathcal{N}_i \\ & \alpha_{i,i} = 0 \quad \forall i, \ j \notin \mathcal{N}_i, \ldots, n \\ & L = \mathrm{diag}(A1) - A, \ A = A^{\mathrm{T}}. \end{aligned} \tag{35}$$

The optimal $p(Z)$ in (9) remains as well as the optimality conditions in (15) for $\alpha_{i,j}, \forall i, j \in \mathcal{N}_i$ hold. In other words, model (34) prefers to learn a sparser weighted graph according to (15).

C. Learning a Sparse Graph in Medium Scale

Motivated by (34), we can construct an initial graph using K-NN or ϵ-N graph from X if no graph is available as a prior. Then, we solve (35) for a refined weighted graph. It is reasonable since the optimal $\alpha_{i,j} = 0$ for the case that $\phi_{i,j}$ is larger than the expected distance in latent space. A K-NN graph means that the edges corresponding to a large distance are removed. This is consistent with setting $\alpha_{i,j} = 0$ in model (35). Another merit of this approach is that the number of variables to optimize reduced to $O(\sum_{i=1}^{n} |\mathcal{N}_i|)$ from $O(n(n-1)/2)$. If K-NN with $k \ll n$ is used, the number of variables is about $O(kn)$ linear with the number of data points. Hence, this approach can be scaled to a medium size of data for weighted graph learning. The bottleneck of computing logdet term makes it infeasible for large-scale graph, although the L-BFGS-B method can be scaled to millions of variables. However, the hybrid approach might be used to combine DDPG with a bipartite graph as did in [35].

VII. Experiments

In this section, we conduct extensive experiments on both synthetic and real datasets to verify that: 1) the different levels of sparsity in DDPG can be controlled by a single parameter λ_*; 2) DDPG can be used to refine a given graph provided by attributed graph data or speed up the learning with a precomputed graph constructed by K-NN; and 3) DDPG applied to the local and global consistency (LGC) model for semisupervised learning can achieve competitive results to the best graphs learned by compared methods.

A. Learning Sparse Graph From a Dense Dissimilarity Matrix

In Section V, we have theoretically analyzed the impact of hyperparameters on the sparsity of the learned graph of (18). In this section, we conduct experiments to empirically illustrate the capability of learning graphs in various sparsity levels by varying $\lambda_* \in [10^{-4}, 10^5]$ with $\sigma^2 = 10^6$, $\phi = \max_{i,j} \phi_{i,j}$, $m = (\phi/2\sigma^2)$, and $\lambda = 4\sigma^2\lambda_*$. Two datasets are used in the experiments: synthetic three-moon data of 1500 points in 2-D space with appended 98 noisy features i.i.d. drawn from a normal distribution with mean 0 and standard deviation 0.14, and real-world teapot data of 400 images consisting of 76 $\times$ 101 red green blue (RGB) pixels. The Euclidean distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ is used to calculate $\phi_{i,j}$.

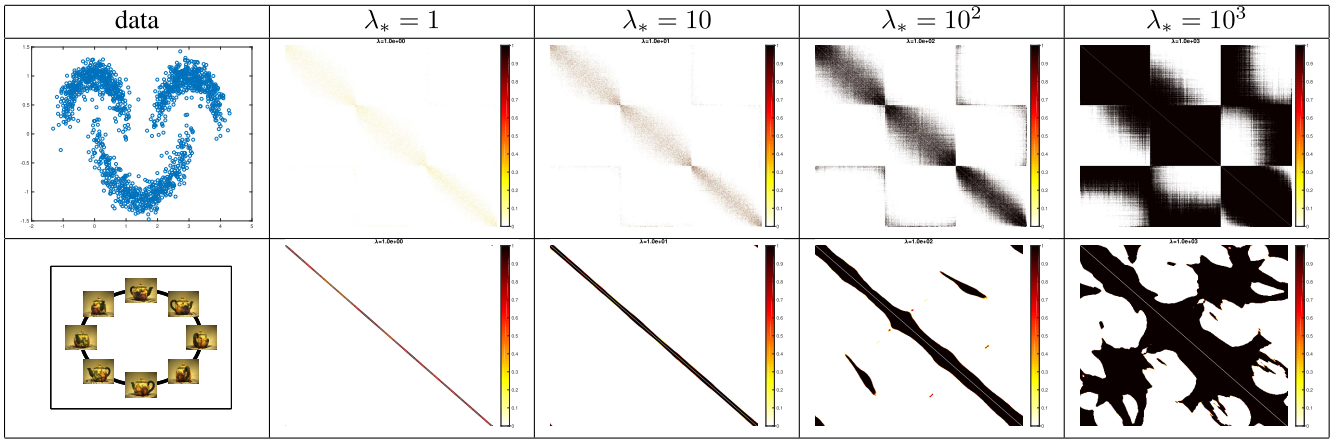

Fig. 1. Weighted undirected graphs learned by the proposed model with varied λ_* on three-moons (top row) and teapot (bottom row).

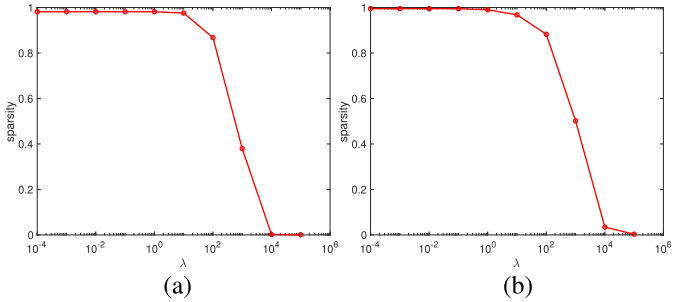

Fig. 2. Sparsity of the learned graph by varying λ_* on (a) three-moons and (b) teapot datasets.

Fig. 1 shows the learned graph A in terms of $\lambda_* \in [1, 10^3]$ on three-moon and teapot datasets. It is observed that all learned graphs can properly capture the manifold structure of two data: three half circles and one circle. In Fig. 2, we present the sparsity ratio as the number of zero entries divided by n^2 with λ_* varied in a large range $[10^{-4}, 10^5]$. From the experimental results on two datasets, we observed that the smaller λ_* is, the sparser the learned graph A is. For example, the sparsity ratio can reach 99.5% if $\lambda_* = 10^{-4}$ and 0.3% if $\lambda_* = 10^5$. Hence, tuning only λ_* would be enough to get various sparsity levels.

B. Graph Refinement and Scalability Analysis

We conduct two different experiments for graph refinement: 1) the input data are graph data with node attributes and 2) a preconstructed graph from input data.

To demonstrate the graph refinement capability of our model, Wiki data are used as an attributed graph data, which contains 2405 documents from 17 classes and 17 981 links, and each document is represented by term frequency–inverse document frequency (TFIDF) features [36] of 4973 words. The cosine distance is used due to its good performance for learning from text data. Fig. 3 shows the input data with link connections and the cosine similarity between any two documents, and the graph learned by the proposed method with $\lambda_* = 1$ and others as the setting used in Section VII-A. By the comparison, it is observed that most connections between two documents of different classes in the input graph are removed after graph refinement; the major connections remained are the connections of documents within the same class, even though quite an amount of connections in the same class are removed too. The refined graph is consistent with the cosine similarity matrix.

The graph refinement can be considered as a strategy of learning a sparse graph with side information, such as the cannot-link priors

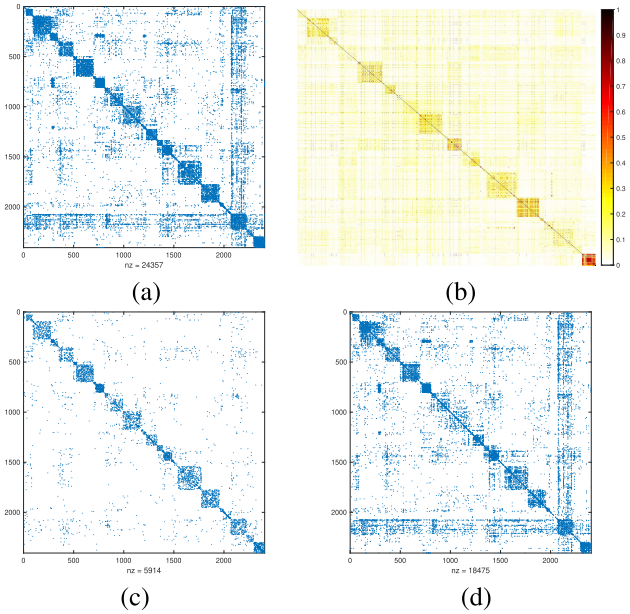

Fig. 3. Graph refinement on Wiki data with (a) link connectivity, (b) cosine distance on TFIDF features of any two documents, (c) learned graph with $\lambda_* = 1$, and (d) removed edges from G after graph refinement.

in clustering methods. For nongraph input data, learning a graph from a preconstructed neighborhood graph (such as K-NN graph) can not only capture the local manifold structure of the input data in high-dimensional space but also speed up the learning process with a fewer number of optimized variables. This strategy has been widely used in locally linear embedding (LLE) and MVU. In Fig. 4, we report the sparsity and central processing unit (CPU) times of the proposed methods on data letter, which contains 20 000 samples with 16 features. First, we show that our model is feasible for learning sparse graphs on 20 000 data points with fixed $k = 10$. Second, the sparser the learned graph is, the more CPU time our model will take. On 10 000 subsamples, we also vary the parameter K in the K-NN graph to form different predefined graphs. A large K decreases the sparsity linearly in the predefined graph, but a small λ_* can promote more sparsity while sacrificing the convergence speed.

C. Graph-Based Semisupervised Learning

We previously have demonstrated in [7] that the graph learned by the proposed model can work well for dimensionality reduction and clustering problems. In these experiments, we will show that our

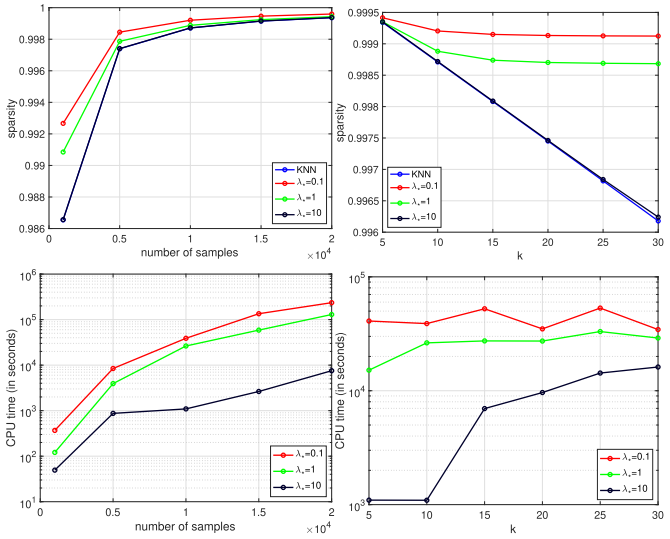

Fig. 4. Sparisty and CPU time of the proposed methods on data letter with respect to varied sizes of samples, predefined K-NNs, and $\lambda_* \in [0.1, 1, 10]$.

TABLE I

OUR GRAPH LEARNING USING DIFFERENT TYPE OF INPUT DISSIMILARITY FUNCTIONS AND THE NORMALIZED AND UN-NORMALIZED LAPLACIAN MATRICES ON THESE GRAPHS

Laplacian	Eucliean	Gaussian	Cosine	χ^2
Un-normalized	DDPG-EU-U	DDPG-GK-U	DDPG-CS-U	DDPG-χ^2-U
Normalized	DDPG-EU-N	DDPG-GK-N	DDPG-CS-N	DDPG-χ^2-N

learned graph can also be well used for graph-based semisupervised learning. We compare the graphs learned by the proposed model with the other graph construction methods, including K-NN, BM, and LLR graph in terms of graph-based semisupervised learning. LGC [3] as the representative method serves as the base learning model to demonstrate the usefulness of the graphs learned by the proposed model. Following the same experimental settings in [6], we run our graph learning model on two datasets USPS and TEXT [37] to get the learned graphs, and then, LGC is evaluated on both 10 and 100 labeled samples with the learned graphs. Four different dissimilarity functions as shown in Section VI-A were evaluated with both normalized and un-normalized graph Laplacian matrices. The shorthand notations for these variants are shown in Table I. In our experiments, we apply the Gaussian kernel as the function to compute kernel distance. As our graph learning model learns a weighted graph, we do not need to construct a connectivity graph and then apply weighting schema such as Gaussian kernel or LLR. For the bandwidth of the Gaussian kernel, we use the average of the pairwise Euclidean distance matrix. The other three dissimilarity functions in Table I do not have any parameters. The same setting of our graph learning model in Section VII-A is applied, where $\lambda_* \in [10^{-2}, 10^2]$ with $\sigma^2 = 10^6$. The mean and standard deviation of error rates over 12 random splits provided in the datasets are reported in percentage.

In Table II, we compared LGC with graphs constructed by our method to those with K-NN and BM method, as well as the five best performers reported in [37]. Note that the Euclidean distance and kernel distance are applied to USPS, while cosine distance and χ^2 for TEXT due to the properties of their features. Also, LGC-BM and LGC-KNN take the Euclidean distance and χ^2 distance for USPS and TEXT, respectively [6]. From Table II, we have the following observations.

1) LGC with our graphs sometimes can achieve the best results such as local and global consistency–density-based distance preserving graph- Gaussian kernel–normalized (LGC-DDPG-GK-N) on USPS with 100 labels and obtain very competitive results to the best performer.
2) With the base learner LGC, our graph is consistently better than K-NN and comparable to the best one local and global consistency-b-matching-Gaussian kernel (LGC-BM-GK) of the three different BM graphs.

TABLE II

ERROR RATE IN PERCENTAGE OF COMPARED METHODS ON TWO BENCHMARK DATASETS AND LGC USING DIFFERENT GRAPH CONSTRUCTION METHODS

Data set	USPS		TEXT	
# of labels	10	100	10	100
QC + CMN	13.61	6.36	40.79	25.71
TSVM	25.20	9.77	31.21	24.52
LDS	17.57	4.96	27.15	23.15
LapRLS	18.99	4.68	33.68	23.57
CHM (normed)	20.53	7.65	-	-
LGC-KNN-BN	14.99	12.34	48.63	43.33
LGC-KNN-GK	12.34	5.49	49.06	41.51
LGC-KNN-LLR	15.88	13.63	44.88	37.52
LGC-BM-BN	14.62	11.71	40.88	26.19
LGC-BM-GK	11.92	5.21	41.32	23.85
LGC-BM-LLR	14.67	12.19	40.27	24.92
LGC-DDPG-EU-U	13.02±2.57	4.90±0.92	-	-
LGC-DDPG-GK-U	12.24±2.51	4.52±1.08	-	-
LGC-DDPG-CS-U	-	-	36.00±4.34	25.09±1.49
LGC-DDPG-χ^2-U	-	-	36.68±5.16	23.95±2.50
LGC-DDPG-EU-N	13.00±2.51	5.48±0.80	-	-
LGC-DDPG-GK-N	12.24±2.51	4.39±1.07	-	-
LGC-DDPG-CS-N	-	-	36.96±4.43	25.32±1.75
LGC-DDPG-χ^2-N	-	-	36.33±4.81	23.89±3.95

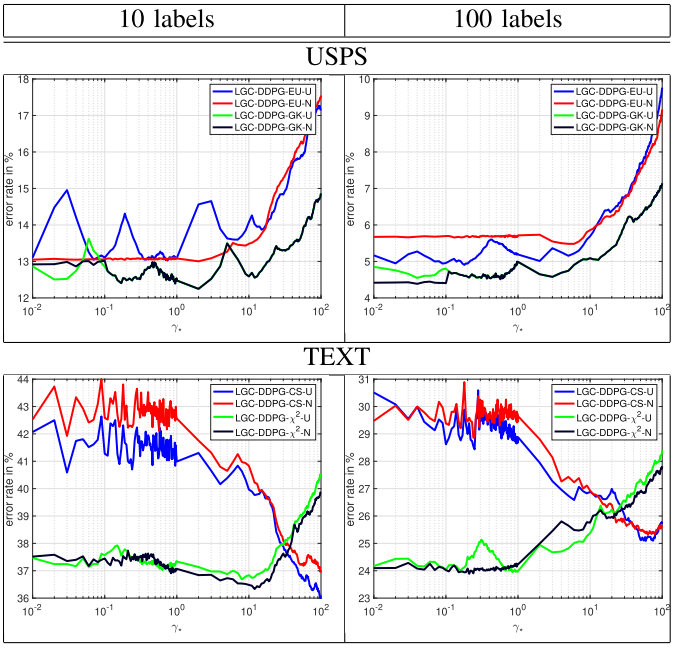

Fig. 5. Sensitivity analysis of our method with respect to parameter λ_* on two datasets with both 10 and 100 labels on four distance functions.

We also conduct the sensitivity analysis of our learned graphs in LGC in terms of error rate with respect to λ_*. The results are shown in Fig. 5. On USPS, the learned graphs with different metric functions behave similarly in LGC, that is, the larger the lambda is, the bigger the error rate becomes, especially $\lambda_* > 10$ for both normalized and un-normalized graph Laplacian matrices. However, LGC with learned graphs using cosine and χ^2 demonstrate completely different behaviors: LGC with cosine distance prefers large λ_*, while LGC with χ^2 performs robustly on a large range of small λ_* but becomes worse for $\lambda_* > 10$. From the graph sparsity point of view, LGC

with χ^2 distance is more robust than that of cosine distance on sparser graphs. Although they behave very differently, both models can achieve competitive results on TEXT with the best baseline method.

All the above observations imply that the graphs learned by our model with different settings can be directly applied to graph-based semisupervised learning models to produce competitive results. Also, it has the advantage of learning graph connectivity and the edge weights simultaneously based on a single hyperparameter after the input distance metric is selected.

VIII. Conclusion

We propose to explore the hidden graph encoded in the dual problem of the density-based distance preservation method for embedding learning as a new approach to construct a sparse similarity graph for various learning problems. Unlike existing methods, our graph learning model aims to learn an undirected graph with nonnegative weights. Also, it can be interpreted as the similarity in analogy to neighborhood graphs from any given dissimilarity metric. Our model takes one single hyperparameter after reparameterization to facilitate the control of the sparsity levels. Besides, our model can be used to refine and reweigh a given graph. The encouraging results obtained in semisupervised learning are showcased. The optimization problem is strongly convex, so the solution is globally optimal, and it facilitates a joint optimization of the graph learning and other learning criteria.

References

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

[2] X. He, D. Cai, and P. Niyogi, "Laplacian score for feature selection," in *Proc. Adv. Neural Inf. Process. Syst.*, 2006, pp. 507–514.

[3] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 16, 2004, pp. 321–328.

[4] U. von Luxburg, "A tutorial on spectral clustering," *Statist. Comput.*, vol. 17, no. 4, pp. 395–416, 2007.

[5] L. Qiao, H. Zhang, M. Kim, S. Teng, L. Zhang, and D. Shen, "Estimating functional brain networks by incorporating a modularity prior," *NeuroImage*, vol. 141, pp. 399–407, Nov. 2016.

[6] T. Jebara, J. Wang, and S.-F. Chang, "Graph construction and b-matching for semi-supervised learning," in *Proc. 26th Annu. Int. Conf. Mach. Learn.*, 2009, pp. 441–448.

[7] Q. Mao, L. Wang, and I. W. Tsang, "A unified probabilistic framework for robust manifold learning and embedding," *Mach. Learn.*, vol. 106, no. 5, pp. 627–650, 2017.

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

[9] F. Wang and C. Zhang, "Label propagation through linear neighborhoods," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 1, pp. 55–67, Jan. 2008.

[10] M. Karasuyama and H. Mamitsuka, "Manifold-based similarity adaptation for label propagation," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 26, 2013, pp. 1547–1555.

[11] L. Wang, Q. Mao, and I. Tsang, "Latent smooth skeleton embedding," in *Proc. AAAI Conf. Artif. Intell.*, vol. 31, no. 1, 2017, pp. 1–7.

[12] L. Wang and R.-C. Li, "Learning low-dimensional latent graph structures: A density estimation approach," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 4, pp. 1098–1112, Apr. 2020.

[13] M. Maier, U. Luxburg, and M. Hein, "Influence of graph construction on graph-based clustering measures," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 21, 2008, pp. 1025–1032.

[14] Z. Xiaojin and G. Zoubin, "Learning from labeled and unlabeled data with label propagation," Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. CMU-CALD-02–107, 2002.

[15] C. A. R. de Sousa, S. O. Rezende, and G. E. Batista, "Influence of graph construction on semi-supervised learning," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases*. Berlin, Germany: Springer, 2013, pp. 160–175.

[16] L. Qiao, L. Zhang, S. Chen, and D. Shen, "Data-driven graph construction and graph learning: A review," *Neurocomputing*, vol. 312, pp. 336–351, Oct. 2018.

[17] C. Cortes and M. Mohri, "On transductive regression," in *Proc. Adv. Neural Inf. Process. Syst.*, 2007, pp. 305–312.

[18] E. Elhamifar and R. Vidal, "Sparse manifold clustering and embedding," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 24, 2011, pp. 55–63.

[19] B. Cheng, J. Yang, S. Yan, Y. Fu, and T. S. Huang, "Learning with ℓ^1-graph for image analysis," *IEEE Trans. Image Process.*, vol. 19, no. 4, pp. 858–866, Dec. 2010.

[20] J. Friedman, T. Hastie, and R. Tibshirani, "Sparse inverse covariance estimation with the graphical lasso," *Biostatistics*, vol. 9, no. 3, pp. 432–441, Jul. 2008.

[21] B. Lake and J. Tenenbaum, "Discovering structure by learning sparse graphs," in *Proc. 32nd Annu. Meeting Cogn. Sci. Soc. (CogSci)*, Portland, OR, USA, Aug. 2010, pp. 778–784.

[22] N. D. Lawrence, "A unifying probabilistic perspective for spectral dimensionality reduction: Insights and new models," *J. Mach. Learn. Res.*, vol. 13, no. 1, pp. 1609–1638, Jan. 2012.

[23] Q. Mao, L. Wang, I. W. Tsang, and Y. Sun, "Principal graph and structure learning based on reversed graph embedding," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 11, pp. 2227–2241, Nov. 2017.

[24] W. Liu, J. He, and S.-F. Chang, "Large graph construction for scalable semi-supervised learning," in *Proc. ICML*, 2010, pp. 679–686.

[25] L. Franceschi, M. Niepert, M. Pontil, and X. He, "Learning discrete structures for graph neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 1972–1982.

[26] R. Li, S. Wang, F. Zhu, and J. Huang, "Adaptive graph convolutional neural networks," in *Proc. AAAI Conf. Artif. Intell.*, vol. 32, no. 1, 2018, pp. 3546–3553.

[27] J. Zhu, N. Chen, and E. P. Xing, "Bayesian inference with posterior regularization and applications to infinite latent SVMs," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1799–1847, 2014.

[28] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, "A limited memory algorithm for bound constrained optimization," *SIAM J. Sci. Comput.*, vol. 16, no. 5, pp. 1190–1208, 1995.

[29] M. Fazel, H. Hindi, and S. P. Boyd, "Log-det heuristic for matrix rank minimization with applications to Hankel and Euclidean distance matrices," in *Proc. IEEE Amer. Control Conf.*, vol. 3, Jun. 2003, pp. 2156–2162.

[30] S. P. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[31] A. J. Smola and R. Kondor, "Kernels and regularization on graphs," in *Learning Theory and Kernel Machines*. Berlin, Germany: Springer, 2003, pp. 144–158.

[32] K. Q. Weinberger, F. Sha, and L. K. Saul, "Learning a kernel matrix for nonlinear dimensionality reduction," in *Proc. 21st Int. Conf. Mach. Learn. (ICML)*, 2004, p. 106.

[33] 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, Dec. 2000.

[34] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Nov. 2008.

[35] Z. Wang, L. Wang, R. Chan, and T. Zeng, "Large-scale semi-supervised learning via graph structure learning over high-dense points," 2019, *arXiv:1912.02233*.

[36] C. Yang, Z. Liu, D. Zhao, M. Sun, and E. Y. Chang, "Network representation learning with rich text information," in *Proc. IJCAI*, 2015, pp. 2111–2117.

[37] O. Chapelle *et al.*, "Semi-supervised learning," *IEEE Trans. Neural Netw.*, vol. 20, no. 3, p. 542, Feb. 2009.