

Exploring Latent Sparse Graph for Large-Scale Semi-supervised Learning

Zitong Wang¹, Li Wang^{2,3}, Raymond Chan⁴, and Tieyong Zeng⁵

¹ Department of Industrial Engineering and Operations Research, Columbia University in the City of New York, USA. zw2690@columbia.edu

² Department of Computer Science and Engineering, University of Texas at Arlington, Arlington, TX 76019-0408, USA. li.wang@uta.edu

³ Department of Mathematics, University of Texas at Arlington, Arlington, TX 76019-0408, USA.

⁴ Department of Mathematics, City University of Hong Kong, Hong Kong. rchan.sci@cityu.edu.hk

⁵ Department of Mathematics, The Chinese University of Hong Kong, Hong Kong. zeng@math.cuhk.edu.hk

Abstract. We focus on developing a novel scalable graph-based semi-supervised learning (SSL) method for input data consisting of a small amount of labeled data and a large amount of unlabeled data. Due to the lack of labeled data and the availability of large-scale unlabeled data, existing SSL methods usually either encounter suboptimal performance because of an improper graph constructed from input data or are impractical due to the high-computational complexity of solving large-scale optimization problems. In this paper, we propose to address both problems by constructing a novel graph of input data for graph-based SSL methods. A density-based approach is proposed to learn a latent graph from input data. Based on the latent graph, a novel graph construction approach is proposed to construct the graph of input data by an efficient formula. With this formula, two transductive graph-based SSL methods are devised with the computational complexity linear in the number of input data points. Extensive experiments on synthetic data and real datasets demonstrate that the proposed methods not only are scalable for large-scale data, but also achieve good classification performance, especially for an extremely small number of labeled data.

Keywords: Graph structure learning · graph-based semi-supervised learning · large-scale learning.

1 Introduction

Semi-supervised learning (SSL) is an important learning paradigm for the situations where a large amount of data are easily obtained, but only a few labeled data points are available due to the laborious or expensive annotation process [4]. A variety of SSL methods have been proposed over the past decades. Among them, graph-based SSL methods attract wide attention due to their superior

performance including manifold regularization [1] and label propagation [34]. However, these methods usually suffer from the high computational complexity of computing the kernel matrix or graph Laplacian matrix and the optimization problem with a large number of optimized variables. Moreover, the quality of the input graph becomes critically important for graph-based SSL methods.

Many graph structure learning methods often provide a reliable similarity matrix (or graph) to characterize the underlying structure of the input data, but their performance can be significantly affected by graphs constructed from the input data with varying density, such as LLE-type graphs [26, 18, 32, 29] and K -NN graphs. Moreover, they are not scalable for large-scale data by learning a full similarity matrix. Graph neural networks are also exploited to infer graph structures from input data [10, 17], but it is not easy to control the sparsity of the graph weights. Some graph construction methods can recover a full graph from a small set of variables such as anchor points of the bipartite graph in [18]. However, these methods usually neglect the importance of the similarities among the small set of variables, e.g., the similarities of anchor points are not explored.

In this paper, we aim to design a novel graph construction approach by taking into account the latent sparse graph learning for high scalability of graph-based SSL. The proposed graph construction approach learns the latent sparse graph and the assignment probabilities to construct the graph of the input data in an efficient form so that graph-based SSL methods can be scalable for large-scale data without the need of explicitly computing the graph of the input data. The main contributions of this paper are summarized as follows:

- A density-based model is proposed to simultaneously learn a sparse latent graph and assignment probabilities from the input data. We further uncover the connection of our density-based model to reversed graph embedding [19] from the perspective of density estimation.
- A novel graph construction approach is proposed to take advantage of both the latent graph and the assignment probabilities learned by the proposed density-based model. We show the spectral properties of our constructed graph via the convergence property of a matrix series. We prove that the graph construction approach used in [18] is a special case of our approach.
- We demonstrate that the graph constructed by our approach can be efficiently integrated into two variants of graph-based SSL methods. We show that both methods have linear computation complexity in the number of data points.
- Extensive experiments on synthetic data and various real data sets are conducted. Results show that our methods not only achieve competitive performance to baselines but also are more efficient for large-scale data.

2 Related Work

2.1 Graph Construction and Graph Learning

Graphs can be constructed via heuristic approach or learned from input data. Manually crafted graphs are often used. Examples include: a dense matrix from

a prefixed kernel function [34], a sparse matrix from a neighborhood graph [1], a pre-constructed graph using labeled data and side information [33], or a transformed graph from an initial one using graph filtering [17]. The dense matrix is computationally impractical for large-scale data due to high storage requirement, and the neighborhood graph is less robust for data with varying density regions [8]. Graph structure learning has also shown great successes in SSL. Learning sparse graphs from input data based on locally linear embedding (LLE) [22] has been widely studied to improve graph-based SSL methods. Linear neighborhood propagation [26] learns a sparse graph via LLE, which is then used in label propagation for SSL. For large-scale data, an anchor graph is constructed by local anchor embedding (LAE) [18]. The joint learning of an LLE-type graph and SSL model has also been explored [32, 29]. The graphs obtained by the above methods highly rely on LLE, so they may not work well in cases where the LLE assumption fails [5]. In addition, various other strategies are also studied. The coefficients from the low-rank representation [35] or matrix completion based on the nuclear norm [25] are used to construct a graph for SSL. Metric learning is used to learn the weights of a graph with the fixed connectivities [27]. Graph neural networks [10, 17] are also used to infer graph structures from input data. These methods update weights of graphs instead of learning a sparse representation, so it is not easy to control the sparsity of the graph weights.

2.2 Scalability Consideration for Large-scale Data

Various methods have been proposed to solve the scalability issue by concentrating on either learning an efficient representation of a graph or developing scalable optimization methods. Graph construction approaches have been proposed to reduce the computation cost of graph-based SSL. The Nystrom method is used to approximate the graph adjacency matrix in [30, 23], but the approximated graph Laplacian matrix is not guaranteed to be positive semi-definite. Numerical approximations to the eigenvectors of the normalized graph Laplacian are used to easily propagate labels through huge collections of images [9]. As pointed out by the authors of [9], the approximations are accurate only when the solution of the label propagation algorithm [34] is a linear combination of the single-coordinate eigenfunctions. This condition can be strong in general. Anchor graph regularization (AGR) [18] constructs the graph of the input data based on a small set of anchor points. Another approach is to design fast optimization algorithms for solving graph-based SSL problems. The primal problem of the Laplacian SVM was solved by the preconditioned conjugate descent [20] for fast approximation solutions. Distributed approaches have been explored by decomposing a large-scale problem into smaller ones [3].

3 Latent Sparse Graph Learning for SSL

We propose a new graph construction approach built on a density-based method and latent graph learning to construct a reliable graph of input data for large-scale graph-based SSL.

3.1 High-density Points Learning

Given input data $\{\mathbf{x}_i\}_{i=1}^n$ with $\mathbf{x}_i \in \mathbb{R}^d$, we seek a small number of latent points called high-density points denoted by $\{\mathbf{c}_s\}_{s=1}^k$ that can best represent the high-density regions of the input data. Our goal here is to formulate a novel objective function for learning these latent points and their relationships with input data.

To model the density of the input data, we employ kernel density estimation (KDE) [6] on $\{\mathbf{c}_s\}_{s=1}^k$ to approximate the true distribution of data by assuming that the observed data $\{\mathbf{x}_i\}_{i=1}^n$ is sampled from the true distribution. The basic idea of KDE involves smoothing each point \mathbf{c}_s by a kernel function. A typical choice of the kernel function is Gaussian. The density function of $\{\mathbf{x}_i\}_{i=1}^n$ becomes

$$p(\mathbf{x}_i | \{\mathbf{c}_s\}_{s=1}^k) = \frac{(2\pi)^{-\frac{d}{2}}}{k\sigma^d} \sum_{s=1}^k \exp\left(-\frac{1}{2\sigma^2} \|\mathbf{x}_i - \mathbf{c}_s\|^2\right), \quad (1)$$

where σ is the bandwidth of the Gaussian kernel function. To obtain the optimal latent points $\{\mathbf{c}_s\}_{s=1}^k$, we can do the maximum log-likelihood estimation:

$$\max_{\{\mathbf{c}_s\}_{s=1}^k} f(C) := \sum_{i=1}^n \log \sum_{s=1}^k \exp\left(-\frac{1}{2\sigma^2} \|\mathbf{x}_i - \mathbf{c}_s\|^2\right), \quad (2)$$

where terms independent of $C = [\mathbf{c}_1, \dots, \mathbf{c}_k] \in \mathbb{R}^{d \times k}$ are ignored. Maximizing (2) is equivalent to finding k peaks of the density function (1). Each peak governs some local high-density region of the density function comparing with other peaks. Let $\{\mathbf{c}_s^*\}_{s=1}^k$ be the optimal solution of problem (2) and denote $C^* = [\mathbf{c}_1^*, \dots, \mathbf{c}_k^*]$. The first order optimality condition of problem (2) is

$$\frac{\partial f(C^*)}{\partial \mathbf{c}_s} = \sum_{i=1}^n Z_{i,s}(\mathbf{x}_i - \mathbf{c}_s^*) = 0 \Rightarrow \mathbf{c}_s^* = \sum_{i=1}^n \frac{Z_{i,s}}{\sum_{i=1}^n Z_{i,s}} \mathbf{x}_i, \forall s = 1, \dots, k. \quad (3)$$

where the assignment probability of \mathbf{x}_i to high-density point \mathbf{c}_k is

$$Z_{i,s} = \exp\left(-\frac{1}{2\sigma^2} \|\mathbf{x}_i - \mathbf{c}_s^*\|^2\right) / \sum_{s=1}^k \exp\left(-\frac{1}{2\sigma^2} \|\mathbf{x}_i - \mathbf{c}_s^*\|^2\right), \forall i, s. \quad (4)$$

We notice that our high-density points learning approach has close relations to probabilistic c-means (PCM) [15] and Gaussian mixture model (GMM) [2]. The key difference is that our unconstrained smooth objective function (2) can facilitate the joint optimization with other objectives as shown in subsection 3.2.

3.2 Joint Learning of High-density Points and Latent Graph

We formulate a joint optimization problem for simultaneously learning a latent graph over high-density points and the probabilities of assigning each input data point to these high-density points. The latent graph consists of the high-density

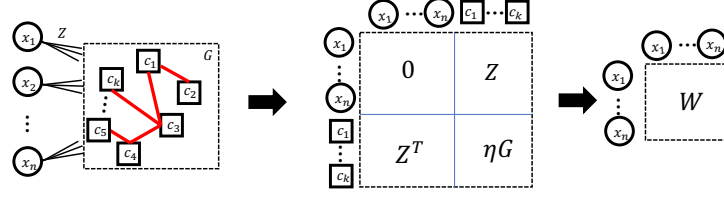


Fig. 1: The construction process of graph similarity matrix W based on Z and G . A larger graph by stacking the input points and the high-density points as vertexes is first constructed, and then W is derived based on the random walk on the larger graph.

points as vertexes and the similarities among these high-density points as edge weights. The latent graph learning aims to find optimal vertexes and edge weights.

We particularly concentrate on spanning trees since they are naturally connected and sparse. Let \mathcal{T} be the set of all spanning trees over $\{\mathbf{c}_i\}_{i=1}^k$, and define by $G \in \{0, 1\}^{k \times k}$ the adjacency matrix for edge weights, where $G_{i,j} = 1$ means \mathbf{c}_i and \mathbf{c}_j are connected, and $G_{i,j} = 0$ otherwise. As each vertex has its associated high-density point as the node feature vector, the dissimilarity of two vertexes can be simply defined as the Euclidean distance between two high-density points.

Our goal is to find an adjacency matrix G with minimum total cost from all feasible spanning trees. By combining the latent graph learning with the high-density points learning, we propose a joint optimization problem as

$$\max_{C, G \in \mathcal{T}} f(C) - \frac{\lambda_1}{4} \sum_{r=1}^k \sum_{s=1}^k G_{r,s} \|\mathbf{c}_r - \mathbf{c}_s\|^2, \quad (5)$$

where λ_1 is a parameter to balance the two objectives.

Suppose G is given. Similarly to (3), we have the following optimality condition

$$\left[\sum_{i=1}^n Z_{i,1}(\mathbf{x}_i - \mathbf{c}_1), \dots, \sum_{i=1}^n Z_{i,k}(\mathbf{x}_i - \mathbf{c}_k) \right] - \lambda_1 CL = 0, \quad (6)$$

where $L = \mathbf{diag}(G\mathbf{1}_k) - G$ is the graph Laplacian matrix over G . Accordingly, we have optimal Z in (4) and the closed-form solution

$$C = XZ(\mathbf{diag}(Z^T \mathbf{1}_n) + \lambda_1 L)^{-1}. \quad (7)$$

Given C , problem (5) with respect to G can be efficiently solved by Kruskal's algorithm [16]. Hence, the alternating method can be used to solve (5).

3.3 A Novel Graph Construction Approach

By solving (5) in Section 3.2, we can obtain C , Z and G . It is worth noting that G characterizing the relationships among high-density points is unique comparing

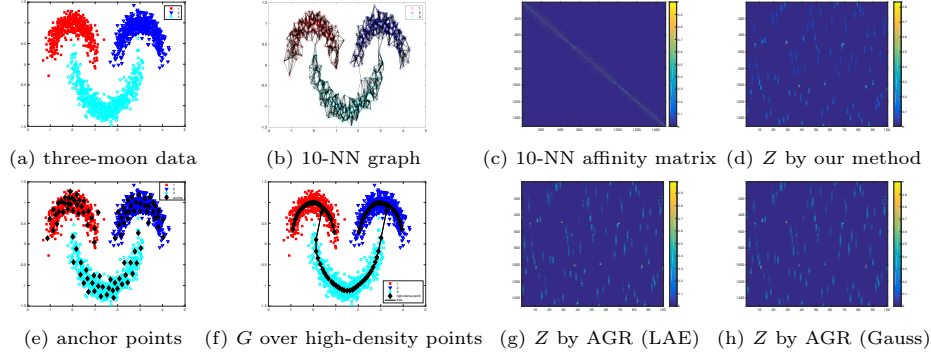


Fig. 2: The graph construction by three methods (LGC, AGR and our proposed method) on three-moon data. (a) the three-moon data points in 2-D space using the first two features. (b)-(c) the 10-NN graph and its affinity matrix used in LGC. (d) the Z matrix obtained by our proposed method. (e) the anchor points obtained by the k -means method with 100 centroids. (f) the optimized high-density points and the learned tree structure. (g)-(h) the Z matrices obtained by LAE and the Nadaraya-Watson kernel regression with Gaussian kernel function in AGR, respectively.

with existing methods such as AGR. Below, we will show how G can be leveraged to build a better graph of input data.

We propose to construct an $n \times n$ affinity matrix W by taking advantage of both Z and G through the proposed process illustrated in Fig. 1. During the graph construction process, we first build a larger graph matrix of $(n+k) \times (n+k)$ with similarities formed by Z and G , and then derive the similarity graph matrix of $n \times n$ based on random walks in order to satisfy certain criterion for SSL. Motivated by the stationary Markov random walks, we propose to construct the affinity matrix $W \in \mathbb{R}^{n \times n}$ by the following equation

$$\begin{bmatrix} W & A_1 \\ A_2 & A_3 \end{bmatrix} = P^2(I_{n+k} - \alpha P)^{-1}, \quad (8)$$

where $\alpha \in (0, 1)$, and

$$P = \text{diag} \left(\begin{bmatrix} \mathbf{0}_{n \times n} & Z \\ Z^T & \eta G \end{bmatrix} \mathbf{1}_{n+k} \right)^{-1} \begin{bmatrix} \mathbf{0}_{n \times n} & Z \\ Z^T & \eta G \end{bmatrix} = \begin{bmatrix} \mathbf{0}_{n \times n} & Z \\ P_{21} & P_{22} \end{bmatrix}. \quad (9)$$

$\mathbf{0}_{n \times n}$ is the $n \times n$ zero matrix, and A_1, A_2, A_3, P_{21} , and P_{22} are sub-blocks in the partition. Here, η is a positive parameter to balance the scale difference between Z and G , and Z is a positive matrix with $Z_{i,s} > 0, \forall i, s$ as defined in (4), and G is a 0-1 matrix. The matrix inverse in (9) always exists. Later, we will show that P is a stochastic matrix and possesses the stationary property.

Fig. 2 demonstrates three key differences of our graph construction approach from LGC and AGR on the synthetic three-moon data: 1) the graph matrix W

over all input data is implicitly represented by both Z and G ; 2) the high-density points characterize the high-density regions of the input data, much better than the simple centroids obtained by the k -means method; 3) the tree structure can effectively model the relationships among these high-density points, while AGR does not have this property. Details of this experiment on synthetic data can be found in subsection 4.4.

Below, we conduct the theoretical analysis to justify the proposed graph construction approach. The proofs of propositions are given in the supplementary material. For convenience of analysis, we denote

$$Q = \begin{bmatrix} \mathbf{0}_{n \times n} & Z \\ Z^T & \eta G \end{bmatrix}, E = \mathbf{diag}(Z^T \mathbf{1}_n + \eta G \mathbf{1}_k), \Gamma = \mathbf{diag}(Q \mathbf{1}_{n+k}) = \begin{bmatrix} I_n & 0 \\ 0 & E \end{bmatrix}. \quad (10)$$

Then $P = \Gamma^{-1}Q$ and $P \mathbf{1}_{n+k} = \mathbf{1}_{n+k}$, satisfying the probability property over each row. We denote $M \geq 0$ if all elements in matrix M are nonnegative.

Firstly, we show in Proposition 1 that the matrix W is symmetric and nonnegative. This result is important since W will be used as the weighted graph in Section 3.4 to compute a graph Laplacian for graph-based SSL.

Proposition 1 *For $\alpha \in (0, 1)$, W defined in (8) is symmetric and nonnegative.*

Secondly, we would like to show that the anchor graph defined in AGR is a special case of our proposed formulation (8).

Proposition 2 *Suppose anchors in AGR are equal to Z defined in (4). If either $\eta = 0$ or $G = 0$, and $\alpha = 0$, then W defined in (8) is the same as anchor graph.*

Thirdly, we demonstrate that the matrix W in (8) can be written in an explicit formula as shown in Proposition 3.

Proposition 3 *W in (8) has an explicit formula:*

$$W = Z(I_k - \alpha\eta E^{-1}G - \alpha^2 E^{-1}Z^T Z)^{-1} E^{-1} Z^T. \quad (11)$$

Fourthly, let us consider the affinity W defined in (11). Since $\tilde{P} = \alpha\eta E^{-1}G + \alpha^2 E^{-1}Z^T Z$ has spectrum in $(-1, 1)$, we have

$$(I_k - \alpha\eta E^{-1}G - \alpha^2 E^{-1}Z^T Z)^{-1} = \sum_{t=0}^{\infty} \tilde{P}^t. \quad (12)$$

The series (12) motivate us to take the second-order approximation to the exact W in (11) for cheaper computation. If we only keep the first two terms, i.e., $t = 0$ and $t = 1$, then we have an approximation of W in (11) as

$$\widetilde{W} = ZE^{-1}Z^T + \alpha\eta ZE^{-1}GE^{-1}Z^T + \alpha^2 ZE^{-1}Z^T ZE^{-1}Z^T. \quad (13)$$

Obviously, matrix \widetilde{W} is symmetric and nonnegative.

Finally, we can verify that much less storage requirement is needed to represent the full graph. Rather than storing the $n \times n$ graph matrices W and \widetilde{W} , we only need to store the $n \times k$ matrix Z , $k \times k$ diagonal matrix E and $k \times k$ matrix $Z^T Z$. We can easily use $Z, E, Z^T Z$ to compute (11) and (13). Hence, our proposed graph construction methods are very efficient for large n but small k .

3.4 Graph-based SSL

We apply W derived in Section 3.3 as the learned graph of input data to two types of SSL methods: LGC-based approach and AGR-based approach. We will show that both approaches can become much more computationally efficient by using the proposed formulas (11) and (13) for large-scale data sets.

Let $F = [F_l; F_u] \in \mathbb{R}^{n \times c}$ be the label matrix of one hot representation $Y = [Y_l; Y_u]$ of class labels that maps n sample data points in $X = [\mathbf{x}_1, \dots, \mathbf{x}_n] \in \mathbb{R}^{d \times n}$ to c labels, where F_l is the submatrix corresponding to the l samples with known labels and F_u corresponds to $n - l$ unlabeled samples. We would like to infer F_u by label propagation. Specifically, denote by \mathbf{L} the graph Laplacian operator of W , i.e., $\mathbf{L}(W) = \text{diag}(W\mathbf{1}_n) - W$, where W is either (11) or (13).

LGC-based approach. By following the objective function of LGC [34], given a graph matrix W , we obtain F_u by solving the following optimization problem:

$$\min_{F_u \in \mathbb{R}^{(n-l) \times c}} \text{trace}(F^T \mathbf{L}(W) F) + \frac{\lambda_2}{2} \|F_u - Y_u\|_{\text{fro}}^2, \quad (14)$$

where $\lambda_2 > 0$ is a regularization parameter. The optimal F_u is obtained by solving c linear systems of equations:

$$(2L_3(W) + \lambda_2 I_{n-l}) F_u^i = b^i, \quad \forall i = 1, \dots, c. \quad (15)$$

where $F_u = [F_u^1, \dots, F_u^c]$, $\lambda_2 Y_u - 2L_2^T(W) F_l = [b^1, \dots, b^c]$, and $L_2(W) \in \mathbb{R}^{l \times (n-l)}$, $L_3(W) \in \mathbb{R}^{(n-l) \times (n-l)}$ are sub-blocks of $\mathbf{L}(W)$. Taking the special form of W , (15) can be solved efficiently by conjugate gradient method [21].

AGR-based approach. Given a graph matrix W , we also study the label propagation model used in AGR [18] by learning a linear decision function, denoted by $F = ZA$, where $A \in \mathbb{R}^{k \times c}$ represents the linear coefficients for c classes. Let $Z = [Z_l; Z_u]$. We solve the following optimization problem

$$\min_A \text{trace}(A^T Z^T \mathbf{L}(W) Z A) + \frac{\lambda_2}{2} \|ZA - Y\|_{\text{fro}}^2. \quad (16)$$

The optimal A^* has a closed-form expression

$$A^* = \lambda_2 (2Z^T \mathbf{L}(W) Z + \lambda_2 Z^T Z)^{-1} (Z^T Y). \quad (17)$$

Note that $Z^T \mathbf{L}(W) Z$ can be computed very efficiently. As in (17), the inverse is on a $k \times k$ matrix, so we simply calculate the matrix inversion since k is small.

Algorithm 1: High-density graph learning (HiDeGL)

Input: $X, F_l = Y_l, k, \sigma, \lambda_1, \lambda_2, \alpha \in (0, 1), \eta$
Output: F, Z, C, G

- 1 Initialization: $Y_u = 0$, k -means for C, Z by (4)
- 2 **while** *not convergent* **do**
- 3 Solve G using the minimal spanning tree algorithm;
- 4 $L = \text{diag}(G\mathbf{1}_k) - G, \Xi = \text{diag}(Z^T \mathbf{1}_n)$;
- 5 $C \leftarrow XZ(\Xi + \lambda_1 L)^{-1}$;
- 6 $Z_{i,s} \leftarrow \frac{\exp(-\|\mathbf{x}_i - \mathbf{c}_s\|^2/\sigma)}{\sum_{s=1}^k \exp(-\|\mathbf{x}_i - \mathbf{c}_s\|^2/\sigma)}, \forall i = 1, \dots, n, s = 1, \dots, k.$
- 7 Construct the graph W using either (11) or (13)
- 8 Update F_u by solving (15) using CG or (16) with A^* in (17).
- 9 $y_i = \arg \max_{j \in \{1, \dots, c\}} \{(F_u)_{i,j}\}, \forall i = l + 1, \dots, n.$

3.5 Optimization Algorithm and Complexity Analysis

The proposed method is summarized in Algorithm 1 with two graph construction approaches and two inference approaches for the unlabeled data. The complexity of Algorithm 1 is determined by two individual subproblems. First, the complexity of finding the high-density points and the tree structure has the complexities of the following three components: 1) the complexity of Kruskal’s algorithm requires $O(k^2 d)$ for computing the fully connected graph and $O(k^2 \log k)$ for finding the spanning tree G ; 2) computing the soft-assignment matrix Z requires $O(nkd)$; 3) computing the inverse of a k by k matrix $(\Xi + \lambda_1 L)^{-1}$ requires $O(k^3)$ and doing the matrix multiplication to get C takes $O(nkd + dk^2)$ flops. Therefore, the total complexity of each iteration is $O(k^3 + nkd + dk^2)$. The second subproblem is the inference of the unlabeled data. The computation complexity of each CG iteration requires $O(nk + k^2)$ for (11) and $O(nk + k^2 + k^3)$ for (13). The complexity of computing (17) needs $O(k^3 + nk(k + c))$. Hence, the complexity of Algorithm 1 is linear with n , no matter which inference method is used.

4 Experiments

4.1 Data sets

One simulated three-moon data set is generated as follows: 500 points in two-dimensional space are first randomly generated on a lower half circle centered at (1.5,0.4) with radius 1.5; and then, another 500 points in two-dimensional space are randomly generated on two upper half unit circles centered at (0,0) and (3,0), respectively; finally, the 1500 points in total are expanded to have dimension 100 by filling up the bottom 98 entries of each point with noise following normal distribution with mean 0 and standard deviation 0.14 to each of the 100 dimensions. USPS-2 is popularly used as benchmark for evaluating the

Table 1: Average accuracies with standard deviations of nine methods over 10 randomly drawn labeled data with varied sizes on three-moon data. Best results are in bold.

method	$l=3$	$l=10$	$l=25$	$l=50$	$l=100$
LGC	94.19±6.69	98.96±0.49	99.02±0.30	99.23±0.13	99.40±0.12
TVRF(1)	90.49±4.80	97.48±1.15	99.53±0.03	99.52±0.05	99.56±0.06
TVRF(2)	99.52±0.07	99.47±0.09	99.46±0.11	99.53±0.03	99.56±0.06
AGR(Gauss)	99.36±0.32	99.46±0.20	99.51±0.25	99.65±0.08	99.61±0.17
AGR(LAE)	97.74±1.41	98.68±0.31	98.66±0.39	98.83±0.29	98.76±0.30
GCN	94.58±4.58	96.54±3.10	98.60±0.15	98.67±0.22	98.74±0.19
IGCN(RNM)	98.28±0.32	98.99±0.09	99.13±0.07	99.15±0.08	99.19±0.13
IGCN(AR)	98.30±0.29	99.01±0.09	99.17±0.08	99.17±0.10	99.22±0.13
GLP(RNM)	97.74±0.84	98.19±0.36	98.58±0.22	98.68±0.13	98.61±0.09
GLP(AR)	95.04±4.51	97.49±1.35	98.28±0.26	98.19±0.27	98.22±0.20
KernelLP	89.63±2.67	92.90±4.22	97.33±1.67	97.89±1.19	98.27±0.86
SSLRR	87.28±4.00	88.47±4.39	96.81±0.29	96.81±0.29	97.04±0.25
HiDeGL(L-approx)	99.85±0.06	99.86±0.07	99.88±0.06	99.88±0.06	99.90±0.05
HiDeGL(L-accurate)	99.85±0.05	99.85±0.06	99.88±0.05	99.88±0.05	99.90±0.05
HiDeGL(A-approx)	99.87±0.05	99.86±0.07	99.88±0.05	99.88±0.06	99.89±0.06
HiDeGL(A-accurate)	99.85±0.09	99.86±0.06	99.87±0.05	99.88±0.05	99.90±0.05

performance of SSL methods. COIL20 is used to show the performance of multi-class classification with a large number of classes. To demonstrate the capability of HiDeGL for medium-size data, we conduct experiments on Pendigits and MNIST. EMNIST-Digits [7] and Extended MNIST [13] are used for large-scale evaluation. The statistics of real datasets are respectively shown in Tables 2-4.

4.2 Compared methods

For ease of references, we name our four proposed methods including the LGC-based approach with (11) and (13) as HiDeGL(L-accurate) and HiDeGL(L-approx), respectively, and the AGR-based approach with (11) and (13) as HiDeGL(A-accurate) and HiDeGL(A-approx), respectively. The comparing methods include LGC [34], AGR with Gaussian kernel regression (AGR-Gauss) and AGR with LAE (AGR-LAE) [18], K -NN classifier (K -NN), spectral graph transduction (SGT) [12], Laplacian regularized least squares (LapRLS) [1], \mathcal{P}_{SQ} solved using SQ-Loss-1 [24], measure propagation (MP) [24]. TVRF with one edge (TVRF(1)) or two edges (TVRF(2)) [28], GCN [14], GLP and IGCN [17], KernelLP [31] and SSLRR [35]. Some methods cannot work for medium-large-size data sets such as KernelLP and SSLRR due to their high computational complexity, so their results on data sets with $n > 9000$ will not be reported.

4.3 Experimental setting

Our experiments follow the work in [24]. For most graph-based SSL methods, there are some hyperparameters to tune. As the labeled data is very limited in our exper-

Table 2: Average accuracies with standard deviations of compared methods over 10 randomly drawn labeled data with varied sizes on two datasets. Best results are in bold.

Method	$l = 10$	$l = 50$	$l = 100$	$l = 150$
USPS-2 ($n = 1500, c = 2, d = 24$)				
k-NN	80.0	90.7	93.6	94.9
SGT	86.2	94.0	96.0	97.0
LapRLS	83.9	93.7	95.4	95.9
SQ-Loss-I	81.4	93.6	95.2	95.2
MP	88.1	93.9	96.2	96.8
LGC	85.21 \pm 5.54	92.94 \pm 3.36	95.94 \pm 0.63	96.73 \pm 0.28
TVRF(1)	82.00 \pm 7.47	88.11 \pm 2.85	92.47 \pm 3.04	94.25 \pm 1.80
TVRF(2)	73.66 \pm 8.15	87.45 \pm 4.19	92.86 \pm 1.67	94.67 \pm 1.05
AGR(Gauss)	75.01 \pm 6.55	88.88 \pm 2.65	91.92 \pm 1.86	93.04 \pm 1.04
AGR(LAE)	74.02 \pm 8.60	88.01 \pm 2.15	91.44 \pm 1.39	92.33 \pm 1.01
GCN	69.52 \pm 10.97	88.01 \pm 4.31	92.74 \pm 2.32	94.59 \pm 1.70
IGCN(RNM)	68.05 \pm 10.06	88.96 \pm 4.28	93.21 \pm 1.77	94.35 \pm 1.76
IGCN_AR	68.26 \pm 9.61	88.17 \pm 3.99	91.62 \pm 1.68	94.22 \pm 1.38
GLP(RNM)	71.78 \pm 9.78	87.10 \pm 5.98	91.36 \pm 3.09	93.61 \pm 2.67
GLP(AR)	69.65 \pm 10.01	86.35 \pm 5.98	90.45 \pm 1.66	93.32 \pm 1.82
KernelLP	72.22 \pm 6.81	88.77 \pm 2.50	92.66 \pm 1.49	93.97 \pm 1.09
SSLRR	64.36 \pm 4.49	67.78 \pm 2.25	68.59 \pm 3.82	68.77 \pm 3.58
HiDeGL(L-approx)	90.01 \pm 3.94	95.88 \pm 0.50	96.23 \pm 0.43	96.77 \pm 0.39
HiDeGL(L-accurate)	89.41 \pm 1.64	95.88 \pm 0.50	96.36 \pm 0.71	96.95 \pm 0.25
HiDeGL(A-approx)	91.93 \pm 3.69	95.30 \pm 0.79	95.68 \pm 0.81	96.16 \pm 0.53
HiDeGL(A-accurate)	91.94 \pm 3.68	95.30 \pm 0.79	95.68 \pm 0.81	96.16 \pm 0.53
Method	$l = 40$	$l = 80$	$l = 100$	$l = 160$
COIL20 ($n = 1440, c = 20, d = 1024$)				
LGC	87.39 \pm 1.43	90.88 \pm 1.53	93.43 \pm 1.22	95.66 \pm 1.13
TVRF(1)	89.31 \pm 2.13	92.65 \pm 0.92	94.24 \pm 1.47	95.20 \pm 1.06
TVRF(2)	87.19 \pm 2.23	90.32 \pm 2.33	92.42 \pm 1.44	95.04 \pm 0.74
AGR(Gauss)	84.16 \pm 3.55	93.81 \pm 2.20	94.09 \pm 1.84	95.70 \pm 1.51
AGR(LAE)	89.55 \pm 3.22	97.19 \pm 1.67	96.91 \pm 1.73	98.24 \pm 0.78
GCN	72.42 \pm 2.16	79.11 \pm 2.04	82.14 \pm 1.35	87.37 \pm 1.41
IGCN(RNM)	74.44 \pm 2.65	80.93 \pm 1.97	83.12 \pm 1.55	88.18 \pm 0.79
IGCN(AR)	75.75 \pm 1.73	81.14 \pm 2.27	84.09 \pm 1.43	88.88 \pm 0.88
GLP(RNM)	73.81 \pm 2.19	80.26 \pm 1.94	82.77 \pm 1.56	87.68 \pm 1.23
GLP(AR)	76.18 \pm 1.80	80.96 \pm 1.89	83.24 \pm 1.18	88.06 \pm 1.14
KernelLP	71.76 \pm 2.70	80.60 \pm 1.60	83.19 \pm 1.10	87.82 \pm 1.47
SSLRR	62.96 \pm 1.61	64.78 \pm 2.25	65.66 \pm 2.54	68.21 \pm 2.90
HiDeGL(L-approx)	92.95 \pm 1.55	96.23 \pm 0.88	96.37 \pm 1.38	97.16 \pm 1.70
HiDeGL(L-accurate)	91.20 \pm 1.65	95.45 \pm 1.30	96.37 \pm 1.41	97.45 \pm 0.77
HiDeGL(A-approx)	96.75 \pm 1.51	97.88 \pm 0.44	98.16 \pm 0.94	98.58 \pm 0.73
HiDeGL(A-accurate)	96.74 \pm 1.43	98.04 \pm 0.98	98.09 \pm 0.74	98.66 \pm 0.52

imental setting, the commonly used cross-validation approach is not applicable [4].

Table 3: Average accuracies with standard deviations of compared methods over 10 randomly drawn labeled data with varied sizes on two datasets. Best results are in bold.

Method	$l = 10$	$l = 50$	$l = 100$	$l = 150$
MNIST ($n = 70000, c = 10, d = 784$)				
LGC	66.66 ± 5.52	83.76 ± 2.33	87.84 ± 1.11	89.41 ± 0.88
TVRF(1)	53.44 ± 6.73	74.35 ± 1.64	78.50 ± 1.70	81.27 ± 1.38
TVRF(2)	61.73 ± 6.12	78.05 ± 2.58	84.70 ± 1.20	86.19 ± 0.95
AGR (Gauss)	51.97 ± 4.15	76.05 ± 4.37	79.26 ± 0.68	80.32 ± 1.41
AGR (LAE)	52.29 ± 3.92	76.97 ± 4.37	80.33 ± 0.93	81.30 ± 1.45
GCN	31.97 ± 6.63	57.59 ± 3.44	64.97 ± 2.21	69.09 ± 1.77
IGCN(RNM)	42.93 ± 5.53	64.67 ± 4.82	76.64 ± 2.39	81.06 ± 2.71
IGCN(AR)	42.26 ± 6.69	68.73 ± 2.81	79.66 ± 1.35	83.60 ± 1.90
GLP(RNM)	44.59 ± 4.49	69.30 ± 4.11	79.21 ± 4.26	83.04 ± 4.04
GLP(AR)	46.60 ± 5.14	70.79 ± 4.35	79.59 ± 5.13	83.27 ± 4.71
HiDeGL(L-approx)	83.38 ± 4.37	88.23 ± 1.87	90.36 \pm 1.33	91.51 ± 0.78
HiDeGL(L-accurate)	83.38 ± 4.37	88.23 ± 1.87	90.36 \pm 1.33	91.51 ± 0.78
HiDeGL(A-approx)	83.59 \pm 4.19	88.22 ± 2.00	90.14 ± 1.15	91.28 \pm 0.84
HiDeGL(A-accurate)	83.59 \pm 4.19	90.73 \pm 1.46	90.14 ± 1.15	91.28 \pm 0.84
Pendigits ($n = 10992, c = 10, d = 16$)				
LGC	80.97 ± 7.41	93.21 ± 1.99	94.44 ± 1.39	95.89 ± 1.02
TVRF(1)	43.57 ± 4.20	59.52 ± 2.11	66.23 ± 2.57	74.69 ± 1.76
TVRF(2)	52.50 ± 4.05	83.39 ± 2.86	89.54 ± 2.80	92.99 ± 1.62
AGR(Gauss)	52.56 ± 6.85	91.73 ± 1.95	95.01 ± 1.03	96.43 ± 0.85
AGR(LAE)	52.52 ± 6.67	91.60 ± 1.88	94.59 ± 1.24	96.18 ± 1.21
GCN	64.87 ± 5.56	83.90 ± 2.01	90.10 ± 1.66	92.72 ± 1.18
IGCN(RNM)	66.74 ± 4.33	83.19 ± 2.01	90.74 ± 1.18	94.00 ± 1.44
IGCN(AR)	71.90 ± 5.83	85.48 ± 2.41	91.41 ± 0.98	94.16 ± 1.34
GLP(RNM)	67.73 ± 5.80	84.46 ± 2.38	89.46 ± 1.45	92.25 ± 0.98
GLP(AR)	67.99 ± 3.63	85.74 ± 2.22	89.58 ± 1.70	92.33 ± 1.14
HiDeGL(L-approx)	85.26 ± 4.09	93.36 ± 1.80	95.54 ± 1.00	96.44 \pm 1.06
HiDeGL(L-accurate)	85.72 \pm 4.08	93.24 ± 1.77	95.56 \pm 0.91	96.36 ± 1.13
HiDeGL(A-approx)	85.37 ± 4.61	93.67 \pm 2.00	95.44 ± 1.72	96.13 ± 0.86
HiDeGL(A-accurate)	85.37 ± 4.61	93.67 \pm 2.00	95.44 ± 1.74	96.14 ± 0.87

To alleviate the difficulty of tuning hyperparameters, we choose to tune all hyperparameters in terms of the mean accuracies over the 10 random experiments for fair comparisons. For the two benchmark datasets, we directly take the results from [24] under the same setting of the number of labeled data $l \in \{10, 50, 100, 150\}$. In the experiments, we tune the parameters $k \in \{200, 500, 750, 1500\}$, $\sigma \in [0.01, 0.5]$, $\lambda_1 \in \{0.1, 1, 10, 100\}$, $\lambda_2 \in \{0.001, 0.01, 0.05\}$, $\eta \in \{0.01, 0.1, 1\}$ and $\alpha \in [0.1, 0.9]$. All these parameters are tuned based on the mean accuracies over the 10 random experiments. The mean accuracies and their standard deviations are reported.

4.4 Experiments on synthetic data

In Fig. 2, we demonstrate the neighborhood graph structure with neighbor size equal to 10 and its affinity matrix used in LGC, anchors in AGR with two approaches (Gauss and LAE) for obtaining Z , and our proposed graph construction approach by optimizing high-density points and a tree structure. In comparing Fig. 2(e) with Fig. 2(f), we highlight the key differences between anchor points and high-density points: 1) high-density points locate in the high-density regions, so they are different from cluster centroids by the k -means method; 2) the additional tree structure shown in Fig. 2(f) is the unique feature compared to the existing methods. We notice that the matrices Z obtained by AGR and HiDeGL are quite similar (see Fig. 2 (d), (g) and (h)). Hence, both methods are able to capture the relations between input data and latent points (anchor points in AGR and high-density points in HiDeGL).

Table 1 shows the average accuracies with standard deviations over 10 randomly drawn labeled data obtained by the compared methods in terms of the varying number of labeled data. From Table 1, we have the following observations: 1) our proposed HiDeGL outperforms other methods over all varying number of labels; 2) With small numbers of labels such as $l \in \{3, 10\}$, HiDeGL performs significantly better than others; 3) the four variants of HiDeGL with two graph construction approaches and two inference approaches for unlabeled data achieve almost similar accuracies. These observations imply that our proposed methods are effective for SSL, especially with very small amount of labeled data.

4.5 Experiments on real data of varied sizes

We evaluate four variants of our HiDeGL on varying sizes of datasets by comparing with baseline methods in terms of varying number of labeled data. The experimental setting same on synthetic data is applied. The average accuracies and their standard deviations are shown in Table 2, Table 3, and Table 4, for varied data sizes. Over all sizes of tested datasets, HiDeGL gives the best accuracy than other methods when the number of labeled data points is small. On benchmark datasets, SGT is the best for $l = 150$ on USPS-2. For medium-size data, the similar results can be observed for a small number of labeled data. With a large number of labeled data, HiDeGL also shows better performance than others. For two large-scale data sets, EMNIST-Digits and Extended MNIST, HiDeGL significantly outperforms AGR over all testing cases. These observations imply that our constructed graphs are effective for SSL.

We further show in Table 4 the CPU time of HiDeGL compared with AGR on EMNIST-Digits and Extended MNIST as the number of labeled data points varies and $k = 500$. It is clear that 1) AGR(Gauss) is the fastest method, while its performance in accuracy is the worst; 2) AGR(LAE) is the slowest since solving LAE for each point is time consuming; 3) HiDeGL with all four variants shows the similar CPU time but 10 times faster than AGR(LAE), and also demonstrates the best performance over all varying numbers of labeled data points.

Table 4: Average accuracies with standard deviations and CPU time of compared methods over 10 randomly drawn labeled data on EMNIST-digits and Extended MNIST in terms of varying number of labeled data points. Best results are in bold.

Method	$l = 10$	$l = 50$	$l = 100$	$l = 150$
Accuracy on EMNIST-digits ($k = 500, n = 280000, c = 10, d = 784$)				
AGR(Gauss)	77.34±4.92	86.46±1.44	88.89±1.17	90.07±0.79
AGR(LAE)	77.93±5.22	87.17±1.94	89.43±1.22	90.60±0.85
HiDeGL(L-approx)	79.55±6.38	89.34±1.34	91.46±0.90	91.85±0.92
HiDeGL(L-accurate)	79.63±6.40	89.36±1.36	91.46±0.90	91.85±0.92
HiDeGL(A-approx)	79.84±6.45	89.55±1.52	91.46±0.89	91.83±0.98
HiDeGL(A-accurate)	79.86±6.52	89.55±1.52	91.46±0.89	91.83±0.98
CPU Time on EMNIST-digits (in seconds)				
AGR(Gauss)	6.59±0.71	6.45±0.28	6.55±0.42	6.56±0.24
AGR(LAE)	8886±18.5	8634±14.1	8641±7.9	8607±14.4
HiDeGL(L-approx)	414.6± 2.5	414.4± 3.9	410.5± 2.7	415.1± 3.2
HiDeGL(L-accurate)	635.5± 14.8	647.7±11.5	629.7± 3.0	648.7±6.9
HiDeGL(A-approx)	403.8± 2.1	403.6± 1.8	403.7± 1.4	403.0± 1.0
HiDeGL(A-accurate)	402.9± 1.9	401.4± 1.4	401.6± 1.5	402.5± 2.3
Accuracy on Extended MNIST ($k = 500, n = 630000, c = 10, d = 784$)				
AGR (Gauss)	64.59±7.36	76.79±1.26	79.88±0.80	82.11±0.78
AGR (LAE)	66.27±7.27	78.72±1.44	80.97±0.75	83.13±0.65
HiDeGL (L-approx)	68.10±7.81	80.97±1.42	82.55±1.24	83.99±1.08
HiDeGL (L-accurate)	68.13±7.80	80.96±1.48	82.55±1.24	84.00±1.08
HiDeGL (A-approx)	68.14±8.33	79.40±1.45	81.41±1.02	83.25±1.06
HiDeGL (A-accurate)	68.14±8.33	79.40±1.45	81.42±1.02	83.25±1.06
CPU time on Extended MNIST (in seconds)				
AGR (Gauss)	13.81±1.50	13.94±1.63	14.49±1.84	14.20±1.62
AGR (LAE)	12120±3281	12134±3331	12242±3307	12183± 3413
HiDeGL (L-approx)	1074±11.6	1078±9.1	1086±9.3	1083±17.1
HiDeGL (L-accurate)	1123±4.7	1146±11.9	1169±7.4	1097±65.7
HiDeGL (A-approx)	1049±15.4	1050± 10.8	1055±11.8	1059± 7.5
HiDeGL (A-accurate)	1047±10.0	1055± 6.1	1052± 7.8	1053± 8.3

4.6 Parameter sensitivity analysis

We conduct the parameter sensitivity analysis of HiDeGL(L-accurate) as an illustrating example in terms of different amount of labeled data. Specifically, we report the best accuracy for the parameter over results obtained by tuning the others using $k = 500$. Fig. 3 shows the accuracies of HiDeGL(L-accurate) by varying parameters. First, we notice that our method is quite robust with respect to λ_1 and η . Second, σ and α can have large impact on the classification performance. It is clear to see that the accuracy changes as σ varies more smoothly with a peak in $[0.05, 0.1]$. Third, the classification accuracies improve as l increases. However, parameters are robust to l due to similar trends.

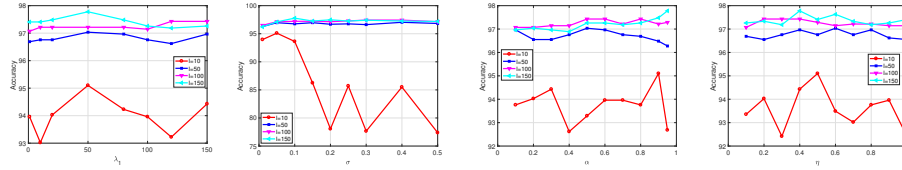


Fig. 3: Parameter sensitivity analysis of HiDeGL(L-accurate) on USPS-2 by varying the corresponding parameters $\lambda_1, \sigma, \alpha, \eta$ respectively with $k = 500$ and $\lambda_2 \in \{10^{-3}, 10^{-2}\}$ in terms of the number of labeled data points $l \in \{10, 50, 100, 150\}$.

5 Conclusion

We proposed a novel graph construction approach for graph-based SSL methods by learning a set of high-density points, the assignment of each input data point to these high-density points, and the relationships over these high-density points as represented by a spanning tree. Our theoretical results showed various useful properties about the constructed graphs, and that AGR is a special case of our approach. Our experimental results showed that our methods not only achieved competitive performance to baseline methods but also were more efficient for large-scale data. More importantly, we found that our methods outperformed all baseline methods on the datasets with extremely small amount of labeled data.

Acknowledgements L. Wang was supported in part by NSF DMS-2009689. R. Chan was supported in part by HKRGC GRF Grants CUHK14301718, CityU11301120, and CRF Grant C1013-21GF. T. Zeng was supported in part by the National Key R&D Program of China under Grant 2021YFE0203700.

References

1. Belkin, M., Niyogi, P., Sindhwani, V.: Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *JMLR* **7**, 2399–2434 (2006)
2. Bishop, C.M.: *Pattern Recognition and Machine Learning*. Springer (2006)
3. Chang, X., Lin, S.B., Zhou, D.X.: Distributed semi-supervised learning with kernel ridge regression. *JMLR* **18**(1), 1493–1514 (2017)
4. Chapelle, O., Schölkopf, B., Zien, A. (eds.): *Semi-Supervised Learning*. MIT Press, Cambridge, MA (2006)
5. Chen, J., Liu, Y.: Locally linear embedding: a survey. *Artificial Intelligence Review* **36**(1), 29–48 (2011)
6. Chen, Y.C.: A tutorial on kernel density estimation and recent advances. *Biostatistics & Epidemiology* **1**(1), 161–187 (2017)
7. Cohen, G., Afshar, S., Tapson, J., van Schaik, A.: Emnist: an extension of mnist to handwritten letters. *arXiv preprint arXiv:1702.05373* (2017)
8. Elhamifar, E., Vidal, R.: Sparse manifold clustering and embedding. In: *NIPS*. pp. 55–63 (2011)

9. Fergus, R., Weiss, Y., Torralba, A.: Semi-supervised learning in gigantic image collections. In: NIPS. pp. 522–530 (2009)
10. Franceschi, L., Niepert, M., Pontil, M., He, X.: Learning discrete structures for graph neural networks. ICML (2019)
11. Geršgorin, S.: Über die abgrenzung der eigenwerte einer matrix. *Izv. Akad. Nauk SSSR Ser. Mat* **1**(7), 749–755 (1931)
12. Joachims, T.: Transductive learning via spectral graph partitioning. In: ICML. pp. 290–297 (2003)
13. Karlen, M., Weston, J., Erkan, A., Collobert, R.: Large scale manifold transduction. In: ICML. pp. 448–455 (2008)
14. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. ICLR (2017)
15. Krishnapuram, R., Keller, J.M.: The possibilistic c-means algorithm: insights and recommendations. *IEEE Transactions on Fuzzy Systems* **4**(3), 385–393 (1996)
16. Kruskal, J.B.: On the shortest spanning subtree of a graph and the traveling salesman problem. *Proc. Am. Math. Soc* **7**(1), 48–50 (1956)
17. Li, Q., Wu, X.M., Liu, H., Zhang, X., Guan, Z.: Label efficient semi-supervised learning via graph filtering. In: CVPR. pp. 9582–9591 (2019)
18. Liu, W., He, J., Chang, S.F.: Large graph construction for scalable semi-supervised learning. In: ICML. pp. 679–686 (2010)
19. Mao, Q., Wang, L., Tsang, I.W.: Principal graph and structure learning based on reversed graph embedding. *IEEE TPAMI* **39**(11), 2227–2241 (2016)
20. Melacci, S., Belkin, M.: Laplacian support vector machines trained in the primal. *JMLR* **12**(Mar), 1149–1184 (2011)
21. Nocedal, J., Wright, S.: Numerical optimization. Springer series in operations research and financial engineering, Springer, New York, NY, 2. ed. edn. (2006)
22. Roweis, S.T., Saul, L.K.: Nonlinear dimensionality reduction by locally linear embedding. *Science* **290**(5500), 2323–2326 (2000)
23. Sivananthan, S., et al.: Manifold regularization based on nyström type subsampling. *Applied and Computational Harmonic Analysis* (2018)
24. Subramanya, A., Bilmes, J.: Semi-supervised learning with measure propagation. *JMLR* **12**(Nov), 3311–3370 (2011)
25. Taherkhani, F., Kazemi, H., Nasrabadi, N.M.: Matrix completion for graph-based deep semi-supervised learning. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. vol. 33, pp. 5058–5065 (2019)
26. Wang, F., Zhang, C.: Label propagation through linear neighborhoods. *IEEE TKDE* **20**(1), 55–67 (2007)
27. Wang, M., Li, H., Tao, D., Lu, K., Wu, X.: Multimodal graph-based reranking for web image search. *IEEE TIP* **21**(11), 4649–4661 (2012)
28. Yin, K., Tai, X.C.: An effective region force for some variational models for learning and clustering. *Journal of Scientific Computing* **74**(1), 175–196 (2018)
29. Zhang, H., Zhang, Z., Zhao, M., Ye, Q., Zhang, M., Wang, M.: Robust triple-matrix-recovery-based auto-weighted label propagation for classification. *arXiv preprint arXiv:1911.08678* (2019)
30. Zhang, K., Kwok, J.T., Parvin, B.: Prototype vector machine for large scale semi-supervised learning. In: ICML. pp. 1233–1240. ACM (2009)
31. Zhang, Z., Jia, L., Zhao, M., Liu, G., Wang, M., Yan, S.: Kernel-induced label propagation by mapping for semi-supervised classification. *IEEE TBD* **5**(2), 148–165 (2019)

32. Zhang, Z., Zhang, Y., Liu, G., Tang, J., Yan, S., Wang, M.: Joint label prediction based semi-supervised adaptive concept factorization for robust data representation. *IEEE TKDE* (2019)
33. Zhang, Z., Zhao, M., Chow, T.W.: Marginal semi-supervised sub-manifold projections with informative constraints for dimensionality reduction and recognition. *Neural Networks* **36**, 97–111 (2012)
34. Zhou, D., Bousquet, O., Lal, T.N., Weston, J., Schölkopf, B.: Learning with local and global consistency. In: *NIPS*. pp. 321–328 (2003)
35. Zhuang, L., Zhou, Z., Gao, S., Yin, J., Lin, Z., Ma, Y.: Label information guided graph construction for semi-supervised learning. *IEEE TIP* **26**(9), 4182–4192 (2017)