Large-Scale Semi-supervised Learning via Graph Structure Learning over High-dense Points

Li Wang

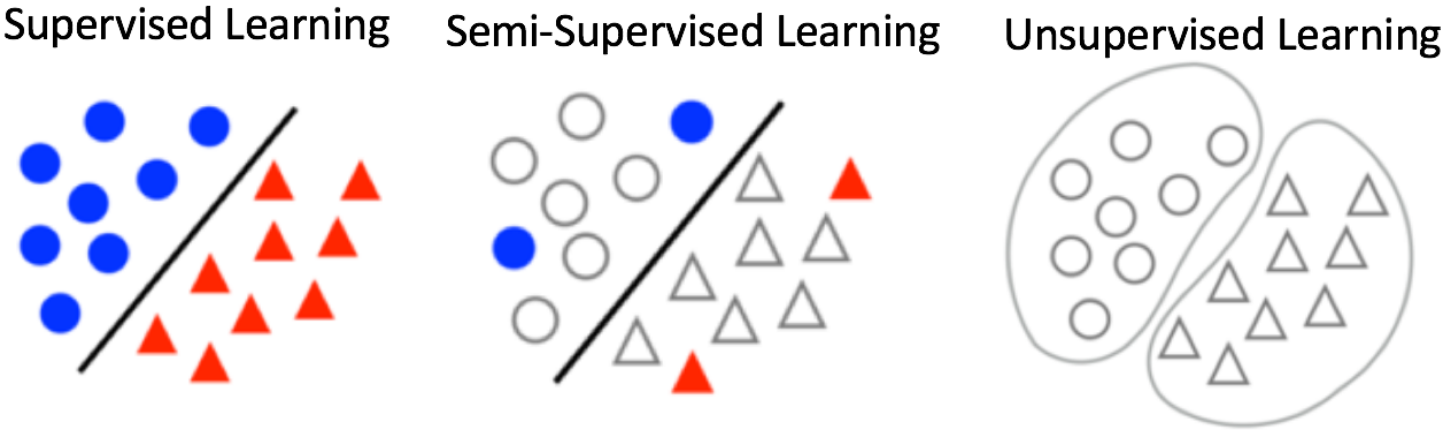
Department of Mathematics, College of Science
Department of Computer Science and Engineering, College of Engineering
University of Texas at Arlington

Joint work with Zitong Wang (CUHK), Raymond Chan (CUHK) and Tieyong Zeng (CUHK)

ICCM 2020 Online Series Conference on Applied Mathematics

August 20, 2020

What is Semi-supervised Learning (SSL)?



Semi-supervised learning (SSL) (Chapelle,2009; Zhu,2005) aims to improve the learning problem in the case that small amounts of labeled data and relatively large amounts of unlabeled data are available. SSL has been widely used in many machine learning applications when annotating training data is time-consuming, costly and error-prone.

Related Work and Motivation

A plenty of SSL algorithms have been proposed in the literature. They are built on various assumptions of the given data, including

- generative models (Druck and McCallum, 2010)
- graph-based methods (Chapelle, Scholkopf, and Zien, 2009; Joachims 2003)
- embedding learning (Weston, Ratle, Mobahi, and Collobert, 2012)
- density-region approaches (Joachims, 1999)

The fundamental assumption of the graph-based methods is that: the data is embedded in a low-dimensional manifold that may be reasonably expressed by a graph, where each vertex is associated with an input data point and the weight of each edge represents the similarity between two vertices, so that nearby vertices are more likely to have the same labels.

Local and Global Consistency (LGC) (1) (Zhou et al. NIPS, 2003)

Given a dataset $X = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{d \times n}$ and a label set $\mathcal{Y} = \{1, \ldots, c\}$, the first l data have labels denoted by $\{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$ where $y_i \in \mathcal{Y}$, and the rest of data are unlabeled.

LGC builds on the following two assumptions:

- **Local assumption:** nearby points are likely to have the same label;
- **Global assumption:** points on the same structure (e.g., clusters or manifold) are likely to have the same label.

Let $F = \begin{bmatrix} F_l \\ F_u \end{bmatrix} \in \mathbb{R}^{n \times c}$ be the classification matrix of X where the label of the ith data point can be obtained by

$$y_i = \arg \max_{j \in \{1, \cdots, c\}} F_{i,j}, \forall i = 1, \ldots, n. \tag{1}$$

Accordingly, the class labels of X can be encoded as $Y \in [0, 1]^{n \times c}$ where $Y_{i,j} = 1$ if $\mathbf{x}_i$ is labeled with $y_i = j$, and $Y_{i,j} = 0$ otherwise.

$Y_{i,j} = 0, \forall j = 1, \ldots, c$, there is no bias to any specific label for unlabeled data.

Local and Global Consistency (LGC) (2) (Zhou et al. NIPS, 2003)

Mathematically, to formulate the local assumption, a non-negative symmetric matrix $W \in \mathbb{R}_+^{n \times n}$ with diagonal elements as zeros is introduced, and the optimal F that satisfies the local assumption is transformed to minimize the following objective function

$$L_W(F) = \frac{1}{2} \sum_{i,j=1}^{n} W_{ij} \left\| \frac{1}{\sqrt{D_{ii}}} F_i - \frac{1}{\sqrt{D_{jj}}} F_j \right\|_{\text{fro}}^2 = \textbf{trace}(F^T (I_n - S) F) \quad (2)$$

where D is a diagonal matrix with $D_{ii} = \sum_{j=1}^{n} W_{ij}, \forall i = 1, \ldots, n$ and $S = D^{-1/2} W D^{-1/2}$. Note that $I_n - S$ is a normalized graph Laplacian matrix over W.

By minimizing $L_W(F)$ with respect to F, it imposes that $\mathbf{x}_i$ and $\mathbf{x}_j$ should have the same label according to (1) since $\frac{1}{\sqrt{D_{ii}}} F_i \approx \frac{1}{\sqrt{D_{jj}}} F_j$ if $W_{i,j}$ is large. Hence, nearby points $\mathbf{x}_i$ and $\mathbf{x}_j$ measured by a large W_{ij} should have the same label that can be enforced by minimizing (2).

Local and Global Consistency (LGC) (3) (Zhou et al. NIPS, 2003)

The global assumption is formulated by a square loss given by

$$L_Y(F) = ||F - Y||^2_{\text{fro}} \tag{3}$$

which means a good classifier should not change too much from the initial label assignment. LGC is proposed to solve

$$\min_F L_W(F) + \mu L_Y(F) \tag{4}$$

where μ is the trade-off between local and global objectives. Problem (4) has the closed form solution

$$F = \mu((1+\mu)I_n - S)^{-1}Y. \tag{5}$$

It is worth noting that the analytic solution (5) is impractical for data with large n since the affinity matrix W needs $O(n^2)$ storage and the computation complexity to calculate the inverse of an n by n matrix is $O(n^3)$.

Anchor Graph Regularization (AGR) (1) (Liu et al. ICML, 2010)

AGR is proposed by constructing W from a stochastic matrix $\widehat{Z} \in \mathbb{R}^{n \times k}$, where k is the number of anchor points and $k \ll n$. Two approaches are used to obtain $\widehat{Z}$.

Let $\{\mathbf{u}_1, \ldots, \mathbf{u}_k\} \subset \mathbb{R}^d$ be a set of anchor points, which are the centroids of the k-means method over X with the number of clusters as k.

- One is from the Nadaraya-Watson kernel regression

$$\widehat{Z}_{i,r} = \frac{K_h(\mathbf{x}_i, \mathbf{u}_r)}{\sum_{r' \in \mathcal{N}_i} K_h(\mathbf{x}_i, \mathbf{u}_{r'})}, \forall r \in \mathcal{N}_i, \tag{6}$$

  where the Gaussian kernel $K_h(\mathbf{x}_i, \mathbf{u}_r) = \exp(-||\mathbf{x}_i - \mathbf{u}_r||/2h^2)$.
- The other is the local anchor embedding (LAE) by solving:

$$\min_{\widehat{Z}} \frac{1}{2} \sum_{i=1}^{n} \left\| \mathbf{x}_i - \sum_{j \in \mathcal{N}_i} \widehat{Z}_{i,j} \mathbf{u}_j \right\|^2 \tag{7}$$

$$\text{s.t.} \sum_{j \in \mathcal{N}_i} \widehat{Z}_{i,j} = 1, \widehat{Z}_{i,j} \geq 0, \forall i, j \in \mathcal{N}_i, \quad \widehat{Z}_{i,j} = 0, \forall i, j \notin \mathcal{N}_i. \tag{8}$$

  This problem can be decomposed into n optimization subproblems.

Anchor Graph Regularization (AGR) (2) (Liu et al. ICML, 2010)

Note that the simplex constraints can promote sparsity of $\widehat{Z}_{i,j}$ with $j \in \mathcal{N}_i$. Hence, the final $\widehat{Z}$ can be as sparse as the $\hat{s}$-nearest neighbor graph. The basic assumption is that any data point $\mathbf{x}_i$ can be represented by a convex combination of its closest $\hat{s}$ anchors, and the coefficients are preserved for the weights in the nonparametric regression.

The affinity matrix W is then constructed from $\widehat{Z}$ by

$$W = \widehat{Z}\Lambda^{-1}\widehat{Z}^T, \tag{9}$$

where $\Lambda \in \mathbb{R}^{k\times k}$ is a diagonal matrix with $\Lambda_{jj} = \sum_{i=1}^{n} \widehat{Z}_{i,j}$. Specifically,

$$W_{i,j} = \sum_{r=1}^{k} \frac{\widehat{Z}_{i,r}\widehat{Z}_{j,r}}{\sum_{i'=1}^{n} \widehat{Z}_{i',r}}, \ \forall i, j. \tag{10}$$

Note that $\sum_j W_{i,j} = \sum_{r=1}^{k} \widehat{Z}_{i,r} = 1, \ \forall i$. Hence, the graph Laplacian matrix of W is $\mathbf{diag}(W\mathbf{1}_n) - W = I_n - W$, so it is normalized.

Anchor Graph Regularization (AGR) (3) (Liu et al. ICML, 2010)

Let

$$p(u_r|\mathbf{x}_i) = \widehat{Z}_{i,r}, \quad p(v_j|u_r) = \frac{\widehat{Z}_{j,r}}{\sum_{i'=1}^{n} \widehat{Z}_{i',r}}. \tag{11}$$

A bipartite graph consists of vertexes $\{\mathbf{x}_i\}_{i=1}^n$ and anchor points $\{u_r\}_{r=1}^k$ is defined as (9). It is clear that

$$p(\mathbf{x}_j|\mathbf{x}_i) = \sum_{r=1}^{k} p(u_r|\mathbf{x}_i)p(\mathbf{x}_j|u_r) = W_{i,j}. \tag{12}$$

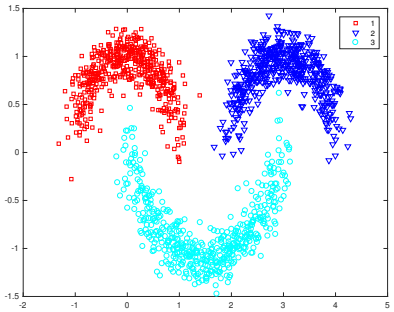
As a result, W is the transition matrix over the bipartite graph from one vertex to the other. (one step of random walk)

Although the anchor graph has nice properties such as the normalized graph Laplacian matrix and the random walk probability interpretation, the heuristic construction based on the k-means method for anchor points and the strong assumption of linear reconstruction of data points from anchor points can degrade the learning performance from labeled and unlabeled data.

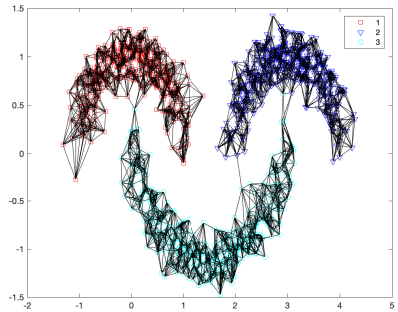
Contributions:

- A subset of representative points are learned from the input data, each of which governs a nearby part of high-dense regions. Moreover, the graph structure such as a spanning tree is learned simultaneously to capture the similarity among these high-dense points.
- A novel graph construction approach is proposed to take the advantage of both the assignment of each input data to its high-dense points and the graph over these high-dense points. We prove that given the set of anchor points, the graph construction in AGR is a special case of our proposed approach.
- We demonstrate that our constructed graph can be efficiently incorporated with two variants of graph-based SSL methods including the approach used in local and global consistency (LGC) and its variant of learning a linear prediction function in AGR. We show that both SSL methods have linear computation complexity with the number of input data. We show that our methods outperform baselines significantly for datasets with extremely small number of labels.

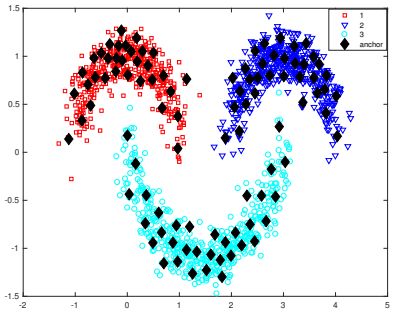
Intuition



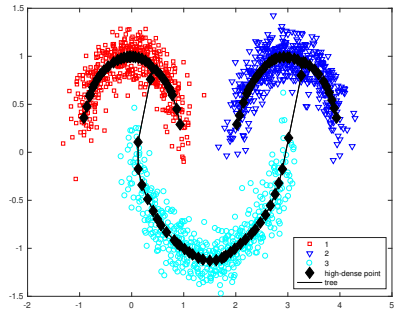
(a) three-moon data



(b) 10-NN graph (LGC)



(c) anchor points (AGR)



(d) G over high-dense points (Our Method)

Figure: 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) the 10-NN graph used in LGC. (c) the anchor points obtained by the k-means method with 100 centroids. (d) the optimized high-dense points and the learned tree structure.

Our Method (1): High-dense Points Learning

Given data $\{\mathbf{x}_i\}_{i=1}^n$, we seek a set of points that can represent the high density regions of the data. For the ease of reference, we name them as the high-dense points denoted as $\{\mathbf{c}_s\}_{s=1}^k$. To model the density of data, we employ kernel density estimation (KDE) 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. Applying KDE to estimate $\mathbf{x}_i$ over high-dense points leads to the following density function,

$$p(\mathbf{x}_i|\{\mathbf{c}_s\}_{s=1}^k) = \frac{(2\pi)^{-\frac{d}{2}}}{k\sigma^d} \sum_{s=1}^{k} \exp(-\frac{1}{2\sigma^2}||\mathbf{x}_i - \mathbf{c}_s||^2), \tag{13}$$

To obtain the optimal set $\{\mathbf{c}_s\}_{s=1}^k$, we can do the maximum likelihood estimation by solving the following maximum optimization:

$$\{\mathbf{c}_s^*\}_{s=1}^k := \arg \max_{\{\mathbf{c}_s\}_{s=1}^k} \log \prod_{i=1}^{n} p(\mathbf{x}_i|\{\mathbf{c}_s\}_{s=1}^k). \tag{14}$$

Our Method (1): High-dense Points Learning

We can further simplify its objective function as

$$f(C) = \sum_{i=1}^{n} \log \sum_{s=1}^{k} \exp(-\frac{1}{2\sigma^2}||\mathbf{x}_i - \mathbf{c}_s||^2). \tag{15}$$

We have the following equation, $\forall s$:

$$\sum_{i=1}^{n} Z_{i,s}(\mathbf{x}_i - \mathbf{c}_s^*) = 0 \Rightarrow \mathbf{c}_s^* = \sum_{i=1}^{n} \frac{Z_{i,s}}{\sum_{i=1}^{n} Z_{i,s}} \mathbf{x}_i, \tag{16}$$

where

$$Z_{i,s} = \frac{\exp(-\frac{1}{2\sigma^2}||\mathbf{x}_i - \mathbf{c}_s^*||^2)}{\sum_{s=1}^{k} \exp(-\frac{1}{2\sigma^2}||\mathbf{x}_i - \mathbf{c}_s^*||^2)}, \forall i, s. \tag{17}$$

Hence, problem (14) can be solved by fixed point iteration in terms of equation (16).

It is worth noting that the high-dense points are different from centroids obtained by the k-means method. Maximizing (14) with respect to $\mathbf{c}_s$ is an iteration step towards the high density $p(X|\mathbf{c}_s)$ over the input data.

Our Method (2): Learning Connectivity Over High-dense Points

Let $\mathcal{T} = (\mathcal{V}, \mathcal{E})$ be a spanning tree with $\mathcal{V}$ as the vertexes and $\mathcal{E}$ as the set of edges. Given the set of high-dense points $\{\mathbf{c}_s\}_{s=1}^k$, we assign each point $\mathbf{c}_s$ to a vertex of the tree $\mathcal{T}$, i.e., $\mathcal{V} = \{\mathbf{c}_s\}_{s=1}^k$, the number of vertexes in the tree $\mathcal{T}$ is k. Let $G \in \{0,1\}^{k \times k}$ be the connectivity matrix where $G_{i,j} = 1$ means $\mathbf{c}_i$ and $\mathbf{c}_j$ are connected, and $G_{i,j} = 0$ otherwise. $\mathcal{T}$ is an undirected graph, so G is symmetric.

By combining the connectivity learning with the high-dense points learning, we propose a joint optimization problem as:

$$\max_{C, G \in \mathcal{T}} f(C) - \frac{\lambda_1}{4} \sum_{r=1}^{k} \sum_{s=1}^{k} G_{r,s} ||\mathbf{c}_r - \mathbf{c}_s||^2 \tag{18}$$

where λ_1 is a parameter to balance the two objectives.

We consider to solve C and G alternatingly.

Why tree structure? Simplest connected graph. The transition between any two points x_i and x_j can be connected by their corresponding high-dense points $\mathbf{c}_r$ and $\mathbf{c}_s$.

Our Method (2): Learning Connectivity Over High-dense Points

Suppose G is given. We obtain the first order optimality condition over C

$$\left[\sum_{i=1}^{n} Z_{i,1}(\mathbf{x}_i - \mathbf{c}_1), \ldots, \sum_{i=1}^{n} Z_{i,k}(\mathbf{x}_i - \mathbf{c}_k)\right] - \lambda_1 CL = 0 \tag{19}$$

where $L = \mathbf{diag}(G\mathbf{1}_k) - G$ is the graph Laplacian matrix over G. We have the closed-form solution for optimization problem (18) with variable C:

$$C = XZ(\Xi + \lambda_1 L)^{-1}. \tag{20}$$

Given C, problem (18) with respect to variable G can be efficiently solved by the Kruskal's algorithm for finding a minimum-cost spanning tree. Hence, problem (18) can be solved by alternating the Kruskal's algorithm for G given C and the fixed point iteration method for updating C given G until convergence.

We aim to recover the graph over the input data using high-dense points C, assignment matrix Z, and their connectivity G.

Our Method (3): Graph Construction

After obtaining C, Z and G, we propose to construct the affinity matrix $W \in \mathbb{R}^{n \times n}$ by the following equation

$$\begin{bmatrix} W & A_1 \\ A_2 & A_3 \end{bmatrix} = P^2 (I_{n+k} - \alpha P)^{-1}, \tag{21}$$

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

$$\begin{aligned} P &= \mathbf{diag}\left(\begin{bmatrix} \mathbf{0}_{n\times n} & Z \\ Z^T & \eta G \end{bmatrix} \mathbf{1}_{n+k} \right)^{-1} \begin{bmatrix} \mathbf{0}_{n\times n} & Z \\ Z^T & \eta G \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{0}_{n\times n} & \mathbf{diag}(Z\mathbf{1}_k)^{-1} Z \\ \mathbf{diag}(Z^T\mathbf{1}_n + \eta G \mathbf{1}_k)^{-1} Z^T & \eta \mathbf{diag}(Z^T\mathbf{1}_n + \eta G \mathbf{1}_k)^{-1} G \end{bmatrix} \\ &= \begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix} = \begin{bmatrix} \mathbf{0}_{n\times n} & Z \\ P_{21} & P_{22} \end{bmatrix}. \end{aligned} \tag{22}$$

$\mathbf{0}_{n\times n}$ is the n by n zero matrix, A_1, A_2 and A_3 are dummy variables for representing W in a compact form. $P_{11} = \mathbf{0}_{n\times n}$, $P_{12} = Z$ since $Z\mathbf{1}_k = \mathbf{1}_n$. η is a positive parameter to balance the scale difference between Z and G. Z is a positive matrix with $Z_{i,s} > 0,\ \forall i, s$ as defined in (17), and G is a 0-1 matrix. The matrix inverse defined in (22) always exists.

Properties of Matrix P and W

Proposition

Let $P \in \mathbb{R}^{(n+k)\times(n+k)}$, and suppose $(I_{n+k} - \alpha P)^{-1}$ exists. Then $P^2(I_{n+k} - \alpha P)^{-1} = (I_{n+k} - \alpha P)^{-1}P^2$.

Proposition

The eigenvalues of matrix P are real, and lie in $[-1, 1]$.

Proposition

Suppose $\alpha \in (0, 1)$, W defined in (21) is symmetric and nonnegative.

Matrix W is symmetric, but the whole matrix $P^2(I - \alpha P)^{-1}$ defined in (21) is not necessarily symmetric.

Proposition

Suppose $\widehat{Z}$ defined in (7) is equal to Z defined in (17). If $\eta = 0$ or $G = 0$ and $\alpha = 0$, W defined in (21) is the same as anchor graph (9).

Our Method (3): Graph Construction

For convenience of discussion, we denote

$$Q = \begin{bmatrix} \mathbf{0}_{n\times n} & Z \\ Z^T & \eta G \end{bmatrix}, \quad E = \mathbf{diag}(Z^T \mathbf{1}_n + \eta G \mathbf{1}_k) \tag{23}$$

and

$$\Gamma = \mathbf{diag}(Q\mathbf{1}_{n+k}) = \begin{bmatrix} I_n & 0 \\ 0 & E \end{bmatrix}, \tag{24}$$

then $P = \Gamma^{-1}Q$ and $P\mathbf{1}_{n+k} = \mathbf{1}_{n+k}$, which satisfies the probability property over each row.

Our Method (3): Graph Construction

$$(I_{n+k} - \alpha P)^{-1} = \begin{bmatrix} I_n & -\alpha Z \\ -\alpha E^{-1} Z^T & I_k - \alpha\eta E^{-1} G \end{bmatrix}^{-1} = \begin{bmatrix} L_{11} & L_{12} \\ L_{21} & L_{22} \end{bmatrix}, \tag{25}$$

where

$$\begin{aligned} L_{11} &= I_n + \alpha^2 Z(I_k - \alpha\eta E^{-1} G - \alpha^2 E^{-1} Z^T Z)^{-1} E^{-1} Z^T, \\ L_{12} &= \alpha Z(I_k - \alpha\eta E^{-1} G - \alpha^2 E^{-1} Z^T Z)^{-1}, \\ L_{21} &= \alpha(I_k - \alpha\eta E^{-1} G - \alpha^2 E^{-1} Z^T Z)^{-1} E^{-1} Z^T, \\ L_{22} &= (I_k - \alpha\eta E^{-1} G - \alpha^2 E^{-1} Z^T Z)^{-1}. \end{aligned}$$

Then

$$\begin{aligned} W &= Z E^{-1} Z^T L_{11} + \eta Z E^{-1} G L_{21}, \\ A_1 &= Z E^{-1} Z^T L_{12} + \eta Z E^{-1} G L_{22}, \\ A_2 &= \eta E^{-1} G E^{-1} Z^T L_{11} + \left(E^{-1} Z^T Z + \eta^2 (E^{-1} G)^2\right) L_{21}, \\ A_3 &= \eta E^{-1} G E^{-1} Z^T L_{12} + \left(E^{-1} Z^T Z + \eta^2 (E^{-1} G)^2\right) L_{22}. \end{aligned} \tag{26}$$

Substituting L_{11} and L_{21} into (26) and simplifying, we get:

$$W = Z(I_k - \alpha\eta E^{-1} G - \alpha^2 E^{-1} Z^T Z)^{-1} E^{-1} Z^T. \tag{27}$$

Our Method (3): Graph Construction

Since $\widetilde{P} = \alpha\eta E^{-1}G + \alpha^2 E^{-1}Z^T Z$ has spectrum between $(-1, 1)$, we have

$$(I_k - \alpha\eta E^{-1}G - \alpha^2 E^{-1}Z^T Z)^{-1} = \sum_{t=0}^{\infty} \widetilde{P}^t. \tag{28}$$

If we only keep the first two terms, i.e., $t = 0$ and $t = 1$, then we have an approximation of W as:

$$\widetilde{W} = ZE^{-1}Z^T + \alpha\eta ZE^{-1}GE^{-1}Z^T + \alpha^2 ZE^{-1}Z^T ZE^{-1}Z^T \tag{29}$$

Rather than storing $n \times n$ 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 get graph matrices (27) and (29). Our graph matrix constructions are very efficient for large-scale datasets.

Our Method (4): SSL finding labels for unlabel data

- LGC-based approach, by solving the following optimization:

$$\min_{F_u \in \mathbb{R}^{(n-l)\times c}} \textbf{trace}(F^T \mathbf{L}(W) F) + \frac{\lambda_2}{2} \|F - Y\|_{\text{fro}}^2 \tag{30}$$

where

$$F = \begin{bmatrix} F_l \\ F_u \end{bmatrix}, \text{and} \quad Y = \begin{bmatrix} Y_l \\ Y_u \end{bmatrix}, \text{and} \quad \mathbf{L}(W) = \begin{bmatrix} L_1(W) & L_2(W) \\ L_2^T(W) & L_3(W) \end{bmatrix}. \tag{31}$$

Taking derivative over F_u and setting gradient over F_u to 0, we get

$$(2L_3(W) + \lambda_2 I_{n-l}) F_u = \lambda_2 Y_u - 2L_2^T(W) F_l. \tag{32}$$

Let $F_u = [F_u^1, \cdots, F_u^c]$, and $\lambda_2 Y_u - 2L_2^T(W) F_l = [b^1, \cdots, b^c]$. Problem (32) can be decomposed into c linear equations that can be solved by conjugate gradient (CG) method in parallel:

$$(2L_3(W) + \lambda_2 I_{n-l}) F_u^i = b^i, \quad \forall i = 1, \cdots, c. \tag{33}$$

For very large number of classes, our method is very efficient.

Our Method (4): SSL finding labels for unlabel data

We would like to study the condition number of the coefficient matrix $2L_3(W) + \lambda_2 I_{n-l}$ in (33). First, the eigenvalues of matrix $L_3(W)$ are shown in the following proposition:

Proposition

Suppose $\alpha \in (0, 1)$. Let W and $\widetilde{W}$ be the graph matrices defined in (27) and (29) respectively. Let $L_3(W)$ be the sub block matrix of $\mathbf{L}(W)$ with graph input matrix W or $\widetilde{W}$.

- *For W, we have the row sums of matrix W are in $[0, \frac{1}{1-\alpha}]$. The eigenvalues of $L_3(W)$ are real and lie in $[0, \frac{2}{1-\alpha}]$.*
- *For $\widetilde{W}$, we have the row sums of matrix $\widetilde{W}$ are in $[0, 1+\alpha]$. The eigenvalues of $L_3(\widetilde{W})$ are real and lie in $[0, 2(1+\alpha)]$.*

If parameters λ_2 and α are properly chosen, the coefficient matrices $2L_3(W) + \lambda_2 I_{n-l}$ and $2L_3(\widetilde{W}) + \lambda_2 I_{n-l}$ will not be ill-conditioned. The CG method will solve the linear equations very efficiently.

Our Method (4): SSL finding labels for unlabel data

- AGR-based approach: Assume $F = ZA$ where $A \in \mathbb{R}^{k \times c}$, i.e., we represent the predicted label F as a linear function of Z. And let $Z = \begin{bmatrix} Z_l \\ Z_u \end{bmatrix}$. We have

$$\begin{aligned} & \textbf{trace}(F^T \mathbf{L}(W) F) + \frac{\lambda_2}{2} \|F - Y\|_{\text{fro}}^2 \\ = & \textbf{trace}(A^T Z^T \mathbf{L}(W) Z A) + \frac{\lambda_2}{2} \|ZA - Y\|_{\text{fro}}^2 \end{aligned} \tag{34}$$

If we minimize above function over variable A, we have

$$A^* = \lambda_2 (2 Z^T \mathbf{L}(W) Z + \lambda_2 Z^T Z)^{-1} (Z^T Y) \tag{35}$$

Here the inverse is defined for a $k \times k$ matrix.

Algorithm

Algorithm 1: High-dense graph learning (HiDeGL)

Input: $k, \sigma, \lambda_1, \lambda_2, \alpha \in (0,1)$, and η

Output: F, Z, C, G

Data: $X, F_l = Y_l$

Initialization: uniform for Y_u, k-means for C, Z by (19)

while *not converge* **do**

  Solve G using minimal spanning tree algorithm;

  $L = \mathbf{diag}(G\mathbf{1}_k) - G,\ \Xi = \mathbf{diag}(Z^T\mathbf{1}_n)$;

  $C \leftarrow XZ(\Xi + \lambda_1 L)^{-1}$;

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

Construct graph W using either (37) or (39)

Update F_u by solving (45) using CG method or (55) with the closed form solution A^* in (56).

$y_i = \arg\max_{j \in \{1,\cdots,c\}} \{(F_u)_{i,j}\}, \forall i = l+1, \ldots, n.$

Our Method (5): Algorithm Complexity

The complexity of Algorithm 1 is determined by two subproblems.

- First, finding the high-dense points and the tree structure learning take 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 matrix multiplication to get C takes $O(nkd + dk^2)$ numeric operations. 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 $O(nk + k^2)$ if graph W is constructed by (27) and $O(nk + k^2 + k^3)$ if graph $\widetilde{W}$ is constructed by (29). The complexity of computing (35) needs $O(k^3 + nk(k + c))$.

Hence, the complexity of Algorithm 1 is linear with n, no matter which inference method, either LGC-based or AGR-based.

Experimental Settings

Table: Datasets used in the experiments

Data Set	n	c	d
three-moon	1,500	3	100
USPS-2	1,500	2	241
USPS	9,298	10	256
Pendigits	10,992	10	16
Letter	15,000	26	16
MNIST	70,000	10	784
EMNIST-Digits	280,000	10	784

Computational Results for Three-moon data

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

method	l=3	l=10	l=25
LGC	94.19 $\pm$ 6.69	98.96 $\pm$ 0.49	99.02 $\pm$ 0.30
TVRF(1)	90.49 $\pm$ 4.80	97.48 $\pm$ 1.15	99.53 $\pm$ 0.03
TVRF(2)	99.52 $\pm$ 0.07	99.47 $\pm$ 0.09	99.46 $\pm$ 0.11
AGR(Gauss)	99.36 $\pm$ 0.32	99.46 $\pm$ 0.20	99.51 $\pm$ 0.25
AGR(LAE)	97.74 $\pm$ 1.41	98.68 $\pm$ 0.31	98.66 $\pm$ 0.39
HiDeGL(L-approx)	99.85 $\pm$ 0.06	**99.86 $\pm$ 0.07**	**99.88 $\pm$ 0.06**
HiDeGL(L-accurate)	99.85 $\pm$ 0.05	99.85 $\pm$ 0.06	**99.88 $\pm$ 0.05**
HiDeGL(A-approx)	**99.87 $\pm$ 0.05**	**99.86 $\pm$ 0.07**	**99.88 $\pm$ 0.05**
HiDeGL(A-accurate)	99.85 $\pm$ 0.09	**99.86 $\pm$ 0.06**	99.87 $\pm$ 0.05

Computational Results for MINST and USPS data

Table: Average accuracies with standard deviations of compared methods over 10 randomly drawn labeled data.

Method	$l = 10$	$l = 50$	$l = 100$	$l = 150$
		MNIST		
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
HiDeGL(L-approx)	75.69 ± 4.69	**85.51 ± 1.94**	87.70 ± 0.66	89.48 ± 0.63
HiDeGL(L-accurate)	**75.73 ± 4.69**	**85.51 ± 1.95**	87.72 ± 0.65	**89.50 ± 0.64**
HiDeGL(A-approx)	72.95 ± 3.83	85.22 ± 1.19	87.85 ± 1.08	89.08 ± 0.72
HiDeGL(A-accurate)	72.95 ± 3.83	85.21 ± 1.19	**87.86 ± 1.07**	89.08 ± 0.73
		USPS		
LGC	83.17 ± 5.24	93.90 ± 0.73	94.90 ± 0.30	95.04 ± 0.39
TVRF(1)	63.36 ± 6.90	81.87 ± 1.78	84.73 ± 0.95	85.58 ± 0.69
TVRF(2)	70.05 ± 7.52	81.65 ± 1.59	88.96 ± 0.19	88.90 ± 0.21
AGR(Gauss)	63.89 ± 10.30	93.35 ± 2.52	94.38 ± 0.48	94.77 ± 0.30
AGR(LAE)	63.51 ± 10.23	93.05 ± 2.39	94.31 ± 0.57	94.52 ± 0.36
HiDeGL(L-approx)	89.53 ± 5.46	94.96 ± 0.92	95.41 ± 0.26	95.55 ± 0.46
HiDeGL(L-accurate)	89.81 ± 4.80	94.96 ± 0.94	95.41 ± 0.26	95.53 ± 0.48
HiDeGL(A-approx)	**91.42 ± 3.81**	95.37 ± 0.36	**95.50 ± 0.23**	**95.60 ± 0.25**
HiDeGL(A-accurate)	91.36 ± 3.86	**95.38 ± 0.36**	95.49 ± 0.22	95.59 ± 0.25

Computational Results for Pendigits and Letter data

Table: Average accuracies with standard deviations of compared methods over 10 randomly drawn labeled data.

Method	$l = 10$	$l = 50$	$l = 100$	$l = 150$
		Pendigits		
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
HiDeGL(L-approx)	85.26 ± 4.09	93.36 ± 1.80	95.54 ± 1.00	**96.44 ± 1.06**
HiDeGL(L-accurate)	**85.72 ± 4.08**	93.24 ± 1.77	**95.56 ± 0.91**	96.36 ± 1.13
HiDeGL(A-approx)	83.01 ± 7.35	**93.67 ± 2.00**	95.44 ± 1.72	96.13 ± 0.86
HiDeGL(A-accurate)	82.89 ± 6.97	**93.67 ± 2.00**	95.44 ± 1.74	96.14 ± 0.87
Method	$l = 26$	$l = 52$	$l = 104$	$l = 156$
		Letter		
LGC	31.12 ± 4.08	39.21 ± 2.54	52.46 ± 2.19	58.35 ± 2.14
TVRF(1)	19.49 ± 2.34	26.06 ± 2.85	33.23 ± 10.47	37.94 ± 11.99
TVRF(2)	22.73 ± 2.40	33.33 ± 4.08	45.79 ± 2.75	51.33 ± 1.47
AGR(Gauss)	25.44 ± 2.98	36.33 ± 2.31	49.18 ± 2.51	56.42 ± 1.99
AGR(LAE)	25.64 ± 2.68	36.42 ± 2.29	49.28 ± 2.55	56.43 ± 2.03
HiDeGL(L-approx)	31.89 ± 4.53	41.34 ± 2.73	53.32 ± 2.24	58.41 ± 1.29
HiDeGL(L-accurate)	32.02 ± 4.54	41.44 ± 2.81	53.17 ± 2.42	58.44 ± 1.35
HiDeGL(A-approx)	33.58 ± 4.43	42.02 ± 2.83	**54.30 ± 1.97**	**58.97 ± 1.71**
HiDeGL(A-accurate)	**33.59 ± 4.42**	**42.03 ± 2.80**	54.21 ± 1.34	58.57 ± 1.70

Computational Results for EMINST data

Table: Average accuracies with standard deviations and CPU time of compared methods over 10 randomly drawn labeled data.

Method	$l = 10$	$l = 50$	$l = 100$	$l = 150$
Accuracy ($k = 500$)				
ARG(Gauss)	47.25 ± 5.39	76.64 ± 1.22	80.41 ± 1.06	82.72 ± 0.98
ARG(LAE)	47.63 ± 5.07	78.06 ± 1.34	81.24 ± 0.85	83.32 ± 0.80
HiDeGL(L-approx)	59.72 ± 8.34	80.63 ± 2.30	**84.18 ± 1.56**	**85.37 ± 0.97**
HiDeGL(L-accurate)	60.74 ± 8.18	80.73 ± 1.74	**84.18 ± 1.56**	**85.37 ± 0.97**
HiDeGL(A-approx)	67.58 ± 7.80	**80.79 ± 1.88**	84.04 ± 1.55	85.24 ± 0.95
HiDeGL(A-accurate)	**67.69 ± 7.74**	**80.79 ± 1.88**	84.04 ± 1.55	85.24 ± 0.95
CPU Time (in seconds)				
AGR(Gauss)	6.4 ±0.44	6.75 ± 0.48	6.53 ± 0.58	6.41 ± 0.33
AGR(LAE)	3430.1 ± 70.3	3409.9 ± 94.2	3368.5 ± 122.3	3391.9 ± 36.4
HiDeGL(L-approx)	242.7 ± 70.4	240.7 ± 43.9	229.7 ± 37.6	257.7 ± 44.0
HiDeGL(L-accurate)	241.7 ± 70.4	239.8 ± 43.8	228.7 ± 37.6	256.7 ± 43.9
HiDeGL(A-approx)	244.0 ± 70.5	243.8 ± 41.6	231.9 ± 36.3	259.1 ± 44.4
HiDeGL(A-accurate)	240.6 ± 70.4	240.4 ± 41.6	228.7 ± 36.4	255.4 ± 44.5

Parameter Sensitivity Analysis

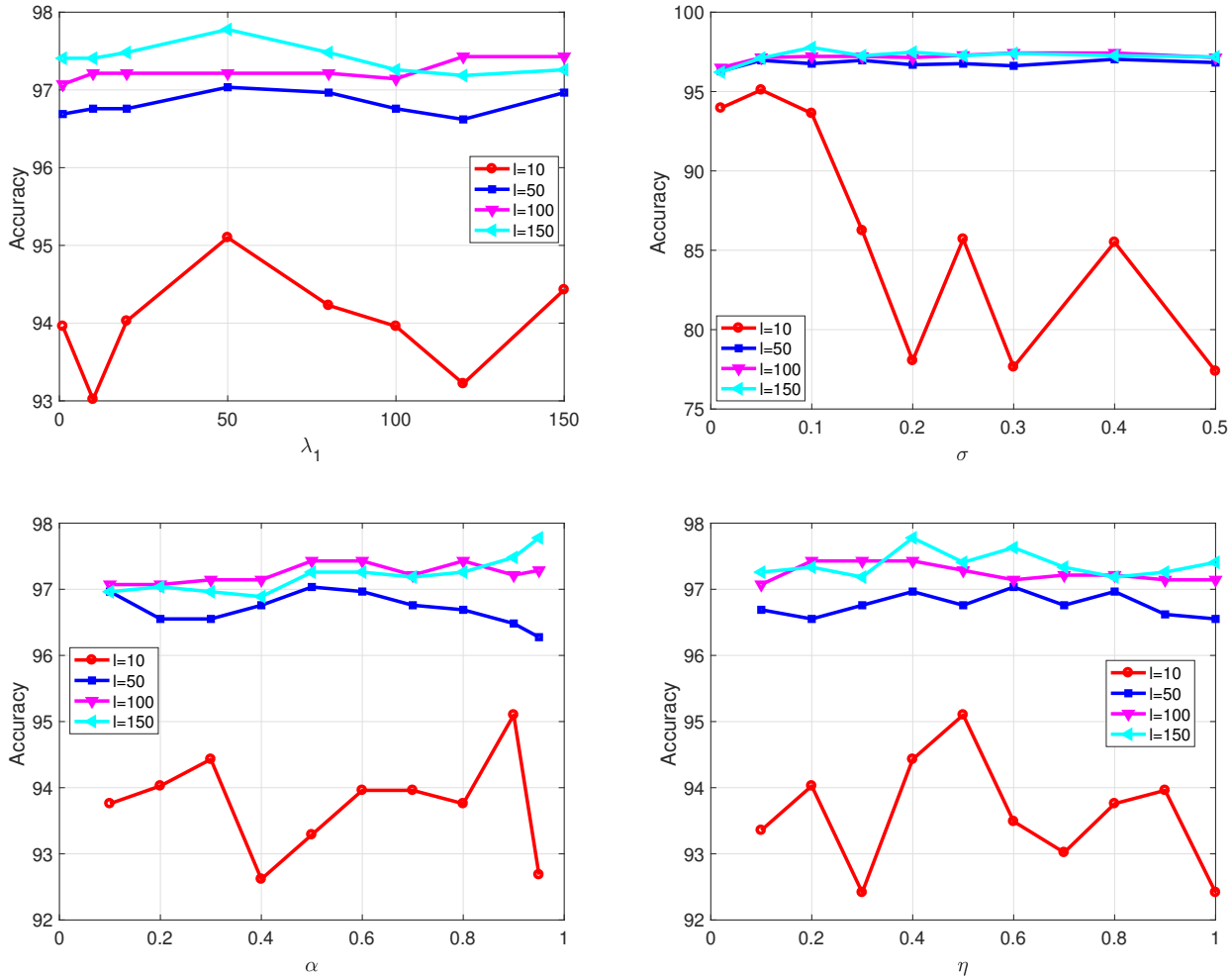


Figure: Parameter sensitivity analysis of HiDeGL(L-accurate) on USPS-2 by varying the corresponding parameters $\lambda_1, \sigma, \alpha, \eta$ respectively with $k = 500$ and $\lambda_2 \in \{10^{-3}, 10^{-2}\}$ in terms of the labeled set $l \in \{10, 50, 100, 150\}$.

Related Reference

http://www.uta.edu/faculty/wangl3/

[1] Large-Scale Semi-supervised Learning via Graph Structure Learning over High-dense Points. Submitted, 2019. https://arxiv.org/abs/1912.02233

(works for datasets with sample size $>$ 1 million)

[2] Probabilistic Semi-supervised Learning via Sparse Graph Structure Learning. IEEE Transactions on Neural Networks and Learning Systems (TNNLS). To appear, 2020.

(Full graph learning for small to medium size data)

Thank you!