Learning Safe Graph Construction from Multiple Graphs

De-Ming Liang[1,2] and Yu-Feng Li[1,2]✉

[1] National Key Laboratory for Novel Software Technology, Nanjing University,
[2] Collaborative Innovation Center of Novel Software Technology and Industrialization
Nanjing 210023, China
{liangdm,liyf}@lamda.nju.edu.cn

Abstract. Graph-based method is one important paradigm of semi-supervised learning (SSL). Its learning performance typically relies on excellent graph construction which, however, remains challenging for general cases. What is more serious, constructing graph improperly may even deteriorate performance, which means its performance is worse than that of its supervised counterpart with only labeled data. For this reason, we consider learning a safe graph construction for graph-based SSL in this work such that its performance will not significantly perform worse than its supervised counterpart. Our basic idea is that, given a data distribution, there often exist some dense areas which are robust to graph construction. We then propose to combine trustable subgraphs in these areas from a set of candidate graphs to derive a safe graph, which remains to be a convex problem. Experimental results on a number of datasets show that our proposal is able to effectively avoid performance degeneration compared with many graph-based SSL methods.

Keywords: safe, graph construction, semi-supervised learning

1 Introduction

Weakly supervised learning [32] is a core area in machine learning, among which semi-supervised learning [8,31] is the representative problem. It aims to improve learning performance via the usage of unlabeled data. During the past decades, extensive SSL studies have been presented, among which one popular paradigm is known as Graph-based SSL (GSSL) [5,28,29]. It is built on smooth assumption [8], i.e., similar instances share similar labels. Technically, it constructs a graph to encode the similarities between labeled and unlabeled data, and then learns a label assignment for unlabeled data in the goal that the inconsistency with respect to the constructed graph is minimized. GSSL is an SSL extension of classic supervised nearest neighbor method [11] and now is found useful for many diverse applications [20].

Nowadays it is widely accepted that the key for the success of GSSL is to construct an excellent graph for given training data, rather than designing various learning or optimization algorithms [3,10,25,31]. For this reason, many efforts have been devoted to graph construction during the decades, e.g., [7,10,24]. Generally, excellent graph construction remains challenging or an open problem, especially when domain knowledge is scarce or insufficient to afford a reliable graph construction.

Beyond constructing excellent graphs, one another or more serious issue is that constructing graph improperly may even deteriorate performance, which means its performance is worse than that of its supervised counterpart (supervised nearest neighbor method) with only labeled data [3,10,28,31]. These phenomena clearly conflicts with the original intention of GSSL. They will also encumber the deployment of GSSL in reality, because the GSSL users typically expect that employing GSSL methods should not be worse than direct supervised nearest neighbor methods. Therefore, it is highly desirable to derive safe graph constructions, which would not be outperformed by its supervised counterpart.

In order to tend this goal, in this work, we present a new GSSL method named SAGRAPH (Safe GRAPH). The basic intuition for our proposal is that, given a data distribution, there often exist some dense areas which are robust or insensitive to graph construction. We refer to these areas as *safe* areas. For the cases where domain knowledge is scarce or insufficient to construct an excellent graph, one may be more reliable to construct a graph from the data in safe areas, so as to avoid the risk caused by an improper graph construction.

Based on this intuition, SAGRAPH proposes to exploit a set of candidate graphs and combines their trustable subgraphs in safe areas to derive a safe graph. To locate the safe areas, SAGRAPH optimizes the worst nearest neighbor error on each training instance according to a set of candidate graphs, and then treats the unlabeled data with the smallest nearest neighbor error (which implies that the prediction on these unlabeled data is not sensitive to graph construction) as the data in safe areas. The final optimization remains to be a convex problem. Experimental results on a number of datasets demonstrate that our proposal clearly improves the safeness of GSSL compared to many state-of-the-art methods.

In the following, we first introduce related work and then present our proposal. After that, we give the experimental justification and finally we conclude this work.

2 Related Work

This work is related to two branches of studies. One is GSSL and the other is safe SSL. In the aspect of GSSL, considerable attention has been paid since it was proposed, which can be separated into two categories. The first one works on various optimization algorithms, e.g. [4,5,10,11,28,29] and the second works on graph construction, e.g. [7,10,25]. There are also approaches, e.g., [1,30] proposed to optimize graph construction as well as label assignment simultaneously. It is notable that, as the deepening of research, graph construction is realized to be more important than the concrete optimization algorithms [3,10,25,31]. Nevertheless, generally, excellent graph construction remains challenging for GSSL. Particularly, the research on explicitly constructing safe graph, to our best knowledge, has not been thoroughly studied yet.

In the aspect of safe SSL, this line of research is raised in very recent. [13,14] is one early work to build safe semi-supervised SVMs. They optimize the worst-case performance gain given a set of candidate low-density separators, showing that the proposed S4VM (Safe Semi-Supervised SVM) is probably safe given that low-density assumption [8] holds. Later, a modified cluster assumption is proposed by Wang et al. [26] to

safely utilize the unlabeled data. Krijthe and Loog [12] present to build a safe inductive semi-supervised classifier, which learns a projection of a supervised least square classifier from all possible semi-supervised least square ones. Apart from least square loss, a safe method for complex performance measures such as Top-k precision, F_β score or AUC is studied in [16]. Recently, Balsubramani and Freund [2] propose to learn a robust and high accurate prediction given that the ground-truth label assignment is restricted to one specific candidate set. Li et al. [15] study the quality of graph via a large margin principle and empirically achieve promising performance, while safe graph construction remains an open problem for GSSL. Besides, Wei et al. [27] study safe multi-label learning with weakly labeled data. Niu et al., [22] give a theoretical study about when positive unlabeled learning outperforms positive negative learning. Li et al. [17] cast the safe semi-supervised regression problem as a geometric projection issue with an efficient algorithm. Most recently, a general formulation for safe weakly supervised learning is proposed [9]. In this work, we consider a new scenario of safe SSL, i.e., safe graph construction that has not been studied.

3 The Proposed Method

In this section, we first briefly introduce a background for GSSL, and then present our idea and problem formulation, finally we derive its connection to safe graph construction and the learning algorithm.

3.1 Brief Background of GSSL

In SSL, we have l labeled instances $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{l}$ and u unlabeled instances $\{\mathbf{x}_j\}_{j=l+1}^{l+u}$ ($l \ll u$). $\mathbf{y} = [y^1, \ldots, y^c] \in \{0,1\}^c$ is the output label vector for input instance $\mathbf{x}$. c denotes the total number of classes, where each instance belongs to one class, i.e., $\sum_{h=1}^{c} y^h = 1$.

For GSSL, a graph $G = (\mathcal{V}, \mathcal{E}, \mathbf{W})$ is constructed for both the labeled and unlabeled data. Specifically, $\mathcal{V}$ is a set of $l+u$ nodes each corresponds to one instance. $\mathcal{E}$ is a set of undirected edges between node pairs. $\mathbf{W} \in \mathbb{R}^{(l+u)\times(l+u)}$ is a nonnegative and symmetric adjacency weighted matrix associating with $\mathcal{E}$ in G, i.e., the weight w_{ij} on the edge $e_{ij} \in \mathcal{E}$ reflects the similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$. The larger the value w_{ij} is, the more similar $\mathbf{x}_i$ and $\mathbf{x}_j$ are. The goal of GSSL is to learn a label assignment $\{\tilde{\mathbf{y}}_j\}_{j=1}^{l+u}$ for training data such that the label inconsistency w.r.t. graph G is minimized. It is cast as the following optimization.

$$\begin{aligned} \min_{\{\tilde{\mathbf{y}}_j\}_{j=1}^{l+u}} & \sum\nolimits_{e_{ij}\in\mathcal{E}} w_{ij}\|\tilde{\mathbf{y}}_i - \tilde{\mathbf{y}}_j\|^2 \\ \text{s.t.} \quad & \tilde{\mathbf{y}}_j \in [0,1]^c, \sum_{h=1}^{c} \tilde{y}_j^h = 1, \forall j = l+1, \ldots, l+u. \\ & \tilde{\mathbf{y}}_i = \mathbf{y}_i, \forall i = 1, \ldots, l. \end{aligned} \tag{1}$$

It is worth noting that as stated in [11], the objective of GSSL (i.e., Eq.(1)) is a tight convex relaxation of supervised nearest neighbor error on training data. In other

words, GSSL is no more than an SSL extension of classic supervised nearest neighbor algorithms for unlabeled data.

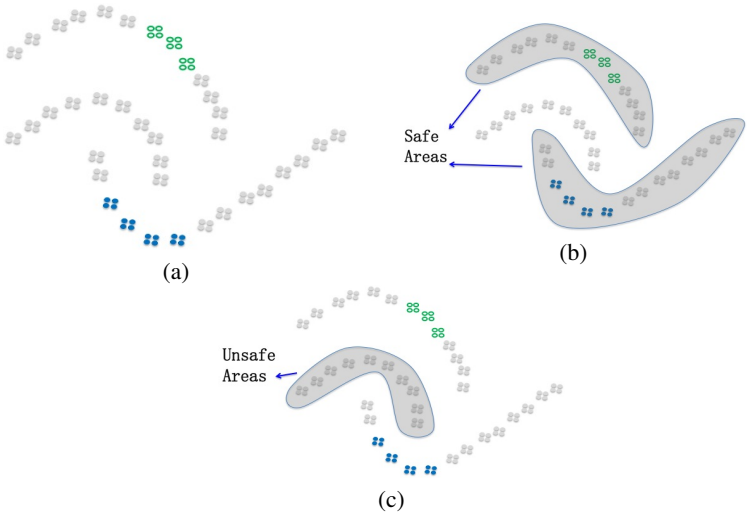

Fig. 1. Illustration for safe areas. (a) Labeled data (empty and filled circles) and unlabeled data (gray points). Given a data distribution, there exist some *safe* areas (b) that are robust to graph construction, and some unsafe areas (c) that are highly sensitive to graph construction.

3.2 Problem Formulation

Conventional GSSL methods typically aim to derive a good performance through a good graph construction. However, as mentioned previously, an inappropriate graph construction will cause GSSL to degenerate performance. To alleviate such a challenging problem, in this work we propose to learn a safe graph construction and present SAGRAPH. Unlike many GSSL methods which are developed on a certain graph, SAGRAPH considers to use a set of candidate graphs and exploits their trustable subgraphs to avoid performance decrease caused by improper graphs.

Figure 1 illustrates the intuition of SAGRAPH. Given several labeled data and a large amount of unlabeled data, there often exist some dense areas of data distribution which are robust or insensitive to graph construction. We refer to these areas as *safe* areas and the other areas as unsafe ones. Without sufficient domain knowledge to construct an excellent graph, one should only exploit the data as well as their subgraphs in safe areas to help improve the performance, and do not use the high risky data in unsafe areas.

The key is to locate safe areas. Remind that, according to the properties of safe areas, the data within safe areas should have small nearest neighbor errors with respect to multiple graphs. This motivates us the formulation of safe graph construction. Specifically, let $\{G_t = (\mathcal{V}, \mathcal{E}_t, \mathbf{W}_t)\}_{t=1}^{T}$ denote a set of candidate graphs, where T is the

number of graphs. Let $er(\tilde{\mathbf{y}}_j)^{G_t}$ denote the nearest neighbor error of training data $\mathbf{x}_j$ on graph G_t, where it is defined as follows [11],

$$er(\tilde{\mathbf{y}}_j)^{G_t} = \delta(p_j^{G_t} \neq \arg\max_{h\in\{1,\ldots,c\}} \tilde{y}_j^h)$$

δ is an indicator function. $p_j^{G_t} = \arg\max_{h\in\{1,\ldots,c\}} \bar{p}_j^h$, where $\bar{\mathbf{p}}_j = [p_j^1, \ldots, p_j^c]$ is the prediction of $\mathbf{x}_j$ via classic nearest neighbor algorithm, i.e.,

$$\bar{\mathbf{p}}_j = \frac{\sum_{k:e_{jk}\in\mathcal{E}_t} \tilde{\mathbf{y}}_k w_{j,k}}{\sum_{k:e_{jk}\in\mathcal{E}_t} w_{jk}}$$

We then aim to locate data in safe areas. Specifically, given a label assignment $\tilde{\mathbf{Y}} = [\tilde{\mathbf{y}}_1, \tilde{\mathbf{y}}_2, \ldots, \tilde{\mathbf{y}}_{l+u}]$ for training data, the worst or equivalently maximal nearest neighbor error of training data $\mathbf{x}_j$ is,

$$\max_{t=1,\ldots,T}\{er(\tilde{\mathbf{y}}_j)^{G_t}\}.$$

We then have our formulation for SAGRAPH as follows,

$$\begin{aligned}
\min_{\tilde{\mathbf{Y}}} & \sum\nolimits_{j=l+1}^{l+u} \max_{t=1,\ldots,T}\{er(\tilde{\mathbf{y}}_j)^{G_t}\} \\
\text{s.t. } & \tilde{\mathbf{y}}_i = \mathbf{y}_i, \forall i = 1,\ldots,l, \\
& \tilde{\mathbf{y}}_j \in [0,1]^c, \forall j = l+1,\ldots,l+u. \\
& \sum\nolimits_{h=1}^{c} \tilde{y}_i^h = 1, \forall i = 1,\ldots,l+u. \\
& \sum\nolimits_{j=l+1}^{l+u} \tilde{\mathbf{y}}_j/u = \sum\nolimits_{i=1}^{l} \tilde{\mathbf{y}}_i/l.
\end{aligned} \tag{2}$$

The last equality is balance constraint [11] in order to avoid trivial solutions which predict all training data to one class. However, the optimization in Eq.(2) is non-trivial, as the form of $er(\tilde{\mathbf{y}}_j)^{G_t}$ is non-continuous and non-convex. Inspired by the way in many GSSL studies [11], a tight convex upper bound of $er(\tilde{\mathbf{y}}_j)^{G_t}$ is optimized alternatively, and we have the following convex form for SAGRAPH,

$$\min_{\tilde{\mathbf{Y}}\in\Omega} \sum_{j=l+1}^{l+u} \max_{t=1,\ldots,T}\{\sum\nolimits_{i:e_{ij}\in\mathcal{E}_t} \tilde{w}_{ij}^t\big(\|\tilde{\mathbf{y}}_i - \tilde{\mathbf{y}}_j\|^2\big)\} \tag{3}$$

where Ω refers to the feasible set of $\tilde{\mathbf{Y}}$, i.e.,

$$\begin{aligned}
\Omega = \ & \{\tilde{\mathbf{Y}} | \tilde{\mathbf{y}}_j = \mathbf{y}_j, j = 1,\ldots,l; \\
& \tilde{\mathbf{y}}_j \in [0,1]^c, j = l+1,\ldots,l+u; \\
& \sum\nolimits_{h=1}^{c} \tilde{y}_i^h = 1, i = 1,\ldots,l+u; \\
& \sum\nolimits_{j=l+1}^{l+u} \tilde{\mathbf{y}}_j/u = \sum\nolimits_{i=1}^{l} \tilde{\mathbf{y}}_i/l\}.
\end{aligned}$$

As Eq.(3) shows, SAGRAPH considers the worst nearest neighbor error, rather than direct nearest neighbor error in typical GSSL methods, tending to derive robust predictions.

Algorithm 1 The SAGRAPH Method

Input: Few label instances $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{l}$, a large amount of unlabeled instances $\{\mathbf{x}_j\}_{j=l+1}^{l+u}$, a set of candidate graphs $\{G_t = (\mathcal{V}, \mathcal{E}_t, \mathbf{W}_t)\}_{t=1}^{T}$, a parameter ϵ;
Output: A label assignment for labeled and unlabeled data $\hat{\mathbf{Y}} = [\hat{\mathbf{y}}_1, \ldots, \hat{\mathbf{y}}_{l+u}]$;

1: Address the convex optimization problem in Eq.(5)
2: Denote $\tilde{\mathbf{Y}}^* = [\tilde{\mathbf{y}}_1^*, \ldots, \tilde{\mathbf{y}}_{l+u}^*]$ as the optimal solution;
3: $\tilde{\mathbf{y}}_j^* = \tilde{\mathbf{y}}_j^* - \sum_{i=1}^{l} \tilde{\mathbf{y}}_i / l$, $u_j = \arg\max_{h=1,\ldots,c} \tilde{y}_j^{*,h}$, $\bar{u}_j = \arg\max_{h \neq u_j} \tilde{y}_j^{*,h}$, $\forall j = l+1, \ldots, l+u$;
4: For $\mathbf{x}_j \in \mathcal{B} = \{\mathbf{x}_k | \tilde{y}_k^{*,u_k} - \tilde{y}_k^{*,\bar{u}_k} \geq \epsilon, \forall j = l+1, \ldots, l+u\}$, predict $\mathbf{x}_j$ to class u_j; Otherwise, predict $\mathbf{x}_j$ with a direct supervised counterpart;
5: Return $\hat{\mathbf{Y}} = [\hat{\mathbf{y}}_1, \ldots, \hat{\mathbf{y}}_{l+u}]$;

3.3 Connection to Safe Graph Construction and Learning Algorithm

In this section we first present that Eq.(3) leads to a safe graph construction and then present the algorithm for Eq.(3). Specifically, by introducing additional variables $\boldsymbol{\theta} = [\theta_{l+1}, \ldots, \theta_{l+u}]$ for each unlabeled data, Eq.(3) can be written in an equivalent way as Eq.(4),

$$\begin{aligned} &\min_{\tilde{\mathbf{Y}} \in \Omega, \boldsymbol{\theta}} \sum\nolimits_{j=l+1}^{l+u} \theta_j \\ &\text{s.t} \quad \theta_j \geq \sum\nolimits_{i:e_{ij} \in \mathcal{E}_t} \tilde{w}_{ij}^t \big(\|\tilde{\mathbf{y}}_i - \tilde{\mathbf{y}}_j\|^2\big), \forall t = 1, \ldots, T. \end{aligned} \tag{4}$$

By further introducing dual variables $\boldsymbol{\alpha} = [\alpha_{1,1}, \ldots, \alpha_{1,T}, \alpha_{2,1}, \ldots, \alpha_{u,T}]$ each corresponds to one constraint in Eq.(4), and setting the derivatives with respect to $\boldsymbol{\theta}$ to zero, Eq.(4) can be further rewritten as Eq.(5),

$$\min_{\tilde{\mathbf{Y}} \in \Omega} f(\tilde{\mathbf{Y}}) \tag{5}$$

where

$$f(\tilde{\mathbf{Y}}) \triangleq \max_{\boldsymbol{\alpha} \in \mathcal{M}} \sum_{j=l+1}^{l+u} \sum_{t=1}^{T} \alpha_{j-l,t} \sum_{i:e_{ij} \in \mathcal{E}_t} \tilde{w}_{ij}^t \big(\|\tilde{\mathbf{y}}_i - \tilde{\mathbf{y}}_j\|^2\big)$$

$$\mathcal{M} = \{\boldsymbol{\alpha} | \boldsymbol{\alpha} \geq \mathbf{0}; \sum\nolimits_{t=1}^{T} \alpha_{j-l,t} = 1, \forall j = l+1, \ldots, l+u\}.$$

We now show that Eq.(5) can be regarded as conventional GSSL methods on a "safe" graph. Specifically, let $\boldsymbol{\alpha}^*$ denote the optimal solution to Eq.(5). We define a new graph $G^* = \{\mathcal{V}, \mathcal{E}^*, \mathbf{W}^*\}$ as follows,

- $e_{ij} \in \mathcal{E}^*$ if and only if $\exists t, \alpha_{j-l,t}^* > 0$ & $e_{ij} \in \mathcal{E}_t$
- $w_{ij}^* = \sum\limits_{t=1}^{T} \alpha_{j-l,t}^* \tilde{w}_{ij}^t$ if and only if $e_{ij} \in \mathcal{E}^*$

Table 1. Statistics of data sets.

Data	# Dim.	# Inst.	# Clas.	Data	# Dim.	# Inst.	# Clas.	Data	# Dim.	# Inst.	# Clas.
austra	15	690	2	coil	241	1,500	2	house	16	232	2
bci	117	400	2	digit1	241	1,500	2	house-votes	16	435	2
usps	256	7,291	10	dna	180	2,000	3	ionosphere	33	351	2
breastw	9	683	2	glass	9	214	6	liverDisorders	6	345	2
clean1	166	476	2	heart	9	270	2	isolet	51	600	2
iris	4	150	3	wine	13	178	3	vehicle	18	846	4
wdbc	14	569	2	text	11,960	1,500	2	-	-	-	-

Refer to the definition of $f(\tilde{\mathbf{Y}})$ and according to the property for optimal solution [21], we can rewrite Eq.(5) in a equivalent way as Eq.(6),

$$\min_{\tilde{\mathbf{Y}} \in \Omega} \sum\nolimits_{e_{ij} \in \mathcal{E}^*} w_{ij}^* \|\tilde{\mathbf{y}}_i - \tilde{\mathbf{y}}_j\|^2. \tag{6}$$

By comparing Eq.(6) with Eq.(1), it is not hard to find that Eq.(6) (or equivalently Eq.(5)) is no more than a GSSL method based on a "safe" graph $G^* = \{\mathcal{V}, \mathcal{E}^*, \mathbf{W}^*\}$.

As for solving Eq.(5), it is easy to find that Eq.(5) is a convex optimization, and can be addressed via state-of-the-art efficient optimization technique. Further note that the objective of Eq.5 is a convex yet non-smooth, and thus gradient-based method is not applicable. In this paper we adopt the projected subgradient optimal method [21] for the solving of Eq.(5), which was shown as the fastest first-order method with optimal convergence rate. After solving Eq.(5), the nearest neighbor errors within the safe areas are small to various graphs and in contrast, the nearest neighbor errors within unsafe areas are diverse. Therefore, the unlabeled data with high confident label assignment are realized as the data in safe areas, while the rest ones are the data in unsafe areas. Algorithm 1 summarizes the pseudo codes of SAGRAPH.

3.4 Efficient SGD Optimization

To optimize Eq.(5), it is a saddle-point convex-concave optimization problem. Direct subgradient approaches are generally slow due to a poor convergence rate [21]. We adopt efficient stochastic gradient descent (SGD) proposed in [18] to solve Eq.(5). Specifically, the key for SGD is to derive an unbiased estimator of full gradient. Let

$$g_j(\tilde{\mathbf{Y}}) = \max_{\boldsymbol{\alpha} \in \mathcal{M}} \sum_{t=1}^{T} \alpha_{j-l,t} \sum_{i:e_{ij} \in \mathcal{E}_t} \tilde{w}_{ij}^t \big(\|\tilde{\mathbf{y}}_i - \tilde{\mathbf{y}}_j\|^2\big).$$

Therefore, $g_j(\tilde{\mathbf{Y}})$ is the value of worst-case nearest neighbor error at the $(j-l)$-th unlabeled instance, and $f(\tilde{\mathbf{Y}}) = \sum_{j=l+1}^{l+u} g_j(\tilde{\mathbf{Y}})$ is the total error. In SGD, the full

gradient of $f(\tilde{\mathbf{Y}})$ is approximated by a gradient at a singe instance as it is unbiased to full gradient:

$$\tilde{\mathbf{Y}}_{r+1} := Proj_{\Omega}(\tilde{\mathbf{Y}}_r - \eta_r \nabla g_j(\tilde{\mathbf{Y}}_r)),\ \forall j = l+1, \ldots, l+u \tag{7}$$

$\tilde{\mathbf{Y}}_r$ is the solution in r-th iteration, η_r is the step size and $Proj_{\Omega}(\mathbf{Y})$ refers to the projection of $\mathbf{Y}$ onto Ω. Through SGD, one can update $\tilde{\mathbf{Y}}$ without calculating the full gradient. Further note that the calculation of $\nabla g_j(\tilde{\mathbf{Y}})$ is cheap as it is only related to a small piece of nodes/instances (i.e., the set of nearest neighbors for each unlabeled instance) and can be computed in a very efficient manner. Specifically, let $h_{j,t}(\tilde{\mathbf{Y}}) = \sum_{i:e_{ij}\in\mathcal{E}_t} \tilde{w}_{ij}^t \left(\|\tilde{\mathbf{y}}_i - \tilde{\mathbf{y}}_j\|^2\right)$ and $mt = \arg\max_{t=1,\ldots,T} h_{j,t}(\tilde{\mathbf{Y}}_r)$. According to the rule of subgradient [21], $\nabla g_j(\tilde{\mathbf{Y}}_r) = \nabla h_{j,mt}(\tilde{\mathbf{Y}}_r)$ is cheap to compute. It is not hard to have the following convergence results by adopting the proof from [6].

Theorem 1 *Let B be the norm of $\tilde{\mathbf{Y}}$ (i.e., $\|\tilde{\mathbf{Y}}\|_2 \leq B$), and G be the upper bound of $\nabla g_j(\tilde{\mathbf{Y}}_r)$ (i.e., $\|\nabla g_j(\tilde{\mathbf{Y}}_r)\|_2 \leq G$). By setting $\eta_r = \frac{B/G}{\sqrt{r}}$, one can have*

$$\mathbb{E}[f(\tilde{\mathbf{Y}}_r)] - f(\tilde{\mathbf{Y}}^*) \leq O(\frac{GB}{\sqrt{r}})$$

Further more, it is easy to show that

Proposition 1 *$B = O((l+u)c)$ and G is a small constant.*

In practice the convergence rate is much better than the theoretical bound. The algorithm often obtains a quite good approximate solution by passing the unlabeled instances in less than five times. In other words, linear running time $O(|E|)$ is usually sufficient to obtain a good empirical result, where $|E|$ is the maximal number of edges among candidate graphs.

After solving Eq.(5), the nearest neighbor errors within the safe areas are small to various graphs and in contrast, the nearest neighbor errors within unsafe areas are diverse. Therefore, the unlabeled data with high confident label assignment are realized as the data in safe areas, while the rest are the data in unsafe ones. Algorithm 1 summarizes the pseudo codes of SAGRAPH.

4 Experiments

We evaluate our SAGRAPH method on a broad range of data sets in this section. The size of dataset ranges from 150 to over 7,000, the dimension of instance ranges from 4 to more than 10,000, and the tasks cover both binary and multi-class classification. Table 1 summarizes the detail statistics of data sets.

4.1 Experimental Setup

The proposed method is compared with the following methods including one baseline supervised method and four classic GSSL methods.

- k**-NN**: supervised k nearest neighbor method, which serves as the baseline method. The key of this work is to show whether GSSL methods could always outperform such a baseline method or not.
- **GFHF**: Gaussian Field Harmonic Function [29]. This method formulates GSSL as gaussian field and uses belief propagation technique for inference, and finally a closed-form solution is derived.
- **CMN**: Class Mass Normalization [29]. This method is a variant of GFHF. CMN further enforces that its prediction on unlabeled data fits a balance constraint, avoiding the cases that all predictions are categorized into one class.
- **LGC**: Local and Global Consistency [28]. This method is motived by the spirit of random walk. It adopts label propagation idea and formulates GSSL as a convex problem with a simple iterative algorithm.
- **SGT**: Spectral Graph Transduction [11]. This method revisits GSSL as an extension form of the nearest neighbor algorithm. It then formulates GSSL as regularized normalized cut form and provides an efficient algorithm via the spectral of Laplacian matrix.

We adopt the code shared in the websites[3][4] for the implementations of GFHF, CMN and SGT. LGC is implemented by ourselves. For LGC and SGT, the parameters are set to the ones recommended in the paper; For SAGRAPH, the parameter ϵ is fixed to 0.3 on all cases.

It is notable that for all GSSL methods, graph construction may exist the cases that some connected components do not contain any labeled data. In this case, inferring the label for the unlabeled instances in these connected components is infeasible. In this paper, we assign such unlabeled instances with the predictive results of supervised k-nearest neighbor method. Note that such a strategy have already improved the safeness of classic GSSL methods. For all the experiments, 10 instances are randomly selected as labeled data for binary classification and 20 instances are randomly selected as labeled data for multi-class classification. The rest are employed as unlabeled data. Experiments repeat for 20 times and average accuracies with standard deviations are reported.

4.2 Comparison Results

We first compare with supervised 1 nearest neighbor (1NN) algorithm. Compared GSSL methods employ 1NN graph as the graph construction. For SAGRAPH method, 1NN, 3NN and 5NN graphs are adopted as candidate graphs. Since SGT focuses on binary classification, it does not have results on multi-class data sets.

Table 2 shows the compared results. As can be seen, in terms of *win* counts, i.e., the times where GSSL method significantly outperforms supervised 1NN method, SAGRAPH obtains the highest times compared with classic GSSL methods. More importantly, as Table 2 shows, compared GSSL methods did will significantly decrease performance in some cases, while SAGRAPH does not suffer from such a deficiency on these data sets. CMN does not work well, mainly because the class mass normalization

[3] http://pages.cs.wisc.edu/œjerryzhu/pub/harmonic_function.m
[4] http://sgt.joachims.org./

Table 2. The accuracies (with standard derivations) of supervised 1 nearest neighbor (1NN) method and multiple GSSL methods. The number in bold (resp. underline) means that the performance is significantly better (resp. worse) than supervised 1NN method (paired t-test at 95% confidence level). The results of Win/Tie/Loss are also listed in the last row, and the methods with the fewest losses are highlighted.

Data	1NN	GFHF	CMN	LLGC	SGT	SAGRAPH
aust	63.4 ± 6.5	63.5 ± 6.6	58.8 ± 6.0	63.6 ± 6.5	63.5 ± 6.6	63.4 ± 6.6
bci	51.3 ± 2.8	51.3 ± 2.9	45.2 ± 2.5	51.3 ± 2.8	51.3 ± 2.8	51.2 ± 2.8
brea	93.3 ± 3.9	93.3 ± 4.1	78.3 ± 4.5	93.3 ± 4.1	93.3 ± 4.1	93.0 ± 7.3
clea	58.6 ± 5.1	58.7 ± 5.2	52.1 ± 5.7	58.7 ± 5.2	58.7 ± 5.2	58.4 ± 5.6
coil	58.1 ± 6.3	$\mathbf{58.2 \pm 6.3}$	56.3 ± 6.4	$\mathbf{58.2 \pm 6.3}$	$\mathbf{58.2 \pm 6.3}$	58.2 ± 6.5
digi	77.5 ± 5.7	77.5 ± 5.7	75.2 ± 5.7	77.5 ± 5.7	77.5 ± 5.7	78.5 ± 9.2
hear	71.1 ± 5.6	71.2 ± 5.7	60.0 ± 5.5	71.1 ± 5.7	71.2 ± 5.7	71.6 ± 5.6
hous	89.0 ± 2.3	87.9 ± 3.0	70.2 ± 4.1	87.9 ± 3.0	88.0 ± 2.8	85.0 ± 12.7
houv	86.6 ± 3.1	86.3 ± 3.0	72.7 ± 4.5	86.3 ± 3.0	86.3 ± 2.9	$\mathbf{87.3 \pm 3.0}$
iono	73.3 ± 6.2	75.8 ± 6.7	51.1 ± 6.1	75.7 ± 6.7	75.7 ± 6.7	73.4 ± 6.4
isol	91.3 ± 4.1	91.3 ± 4.0	84.4 ± 3.9	91.3 ± 4.0	91.3 ± 4.0	$\mathbf{94.1 \pm 4.4}$
live	52.9 ± 3.2	52.6 ± 3.2	46.1 ± 3.9	52.6 ± 3.2	52.6 ± 3.3	52.8 ± 3.3
text	59.8 ± 3.5	59.7 ± 3.5	57.2 ± 3.5	59.7 ± 3.5	59.7 ± 3.5	59.8 ± 3.4
wdbc	81.2 ± 5.6	81.3 ± 5.3	65.1 ± 5.2	81.3 ± 5.3	81.3 ± 5.3	81.2 ± 5.6
dna	54.8 ± 3.3	$\mathbf{57.4 \pm 4.2}$	$\mathbf{57.5 \pm 3.4}$	$\mathbf{57.1 \pm 4.5}$	-	54.8 ± 3.6
glas	55.9 ± 4.3	55.9 ± 3.9	55.9 ± 4.0	56.1 ± 3.9	-	56.2 ± 4.2
iris	94.7 ± 2.9	94.3 ± 3.2	94.3 ± 3.2	94.2 ± 3.2	-	94.8 ± 2.3
usps	64.4 ± 4.5	$\mathbf{64.5 \pm 4.5}$	$\mathbf{64.5 \pm 4.5}$	$\mathbf{64.5 \pm 4.5}$	-	$\mathbf{68.2 \pm 5.4}$
vehi	48.1 ± 4.3	48.2 ± 4.3	48.2 ± 4.3	48.2 ± 4.3	-	$\mathbf{48.6 \pm 4.3}$
wine	92.0 ± 1.8	90.5 ± 2.0	90.5 ± 2.0	90.5 ± 2.0	-	92.5 ± 1.5
GSSL methods against 1NN: win/tie/loss		3/15/2	2/3/15	3/15/2	1/12/1	**4/16/0**

is particularly suitable for one or very few connected components, which may be not that cases when using 1NN graph.

Table 2 shows that compared GSSL methods only obtain comparative performance with 1NN method. One reason is that 1NN graph is not so powerful for GSSL methods. We then evaluate our method in comparison to supervised 5 nearest neighbor (1NN) algorithm. Compared GSSL methods all employ 5NN graph as the graph construction. Following the same setup as before, 1NN, 3NN and 5NN graphs are adopted as candidate graphs for SAGRAPH. SGT works on binary classification and does not have results on multi-class data.

The comparison results are shown in Table 3. As can be seen, in the case of 5NN graph, compared GSSL methods obtain more aggressive results, e.g., CMN works much better because 5NN graph leads to much less connected component. Even in this situation, SAGRAPH still achieves the highest times in terms of *win* counts. More worth

Table 3. The accuracies (with standard derivations) of 5NN method and multiple GSSL methods.

Data	5NN	GFHF	CMN	LLGC	SGT	SAGRAPH
aust	62.5 ± 5.2	59.8 ± 4.3	59.6 ± 4.5	59.6 ± 4.3	56.1 ± 6.1	62.8 ± 5.5
bci	50.3 ± 2.8	49.8 ± 2.1	50.0 ± 2.2	49.7 ± 1.9	49.7 ± 2.1	50.3 ± 2.4
brea	90.2 ± 3.4	**96.0 ± 0.9**	**96.0 ± 0.6**	**95.9 ± 0.7**	**96.6 ± 0.2**	**94.4 ± 3.1**
clea	53.4 ± 3.4	**57.6 ± 4.8**	**57.8 ± 4.8**	**57.5 ± 5.1**	55.1 ± 4.7	54.4 ± 4.4
coil	47.6 ± 5.5	**66.5 ± 6.0**	50.3 ± 9.9	**64.9 ± 7.2**	**65.5 ± 7.5**	**53.3 ± 8.2**
digi	69.2 ± 8.3	**88.2 ± 5.3**	**87.4 ± 3.2**	**90.8 ± 3.2**	**95.0 ± 1.8**	**71.8 ± 9.2**
hear	70.2 ± 5.6	67.2 ± 7.3	69.4 ± 6.7	61.3 ± 5.5	57.8 ± 5.8	**71.6 ± 4.4**
hous	89.7 ± 1.8	84.0 ± 8.6	88.3 ± 2.3	78.4 ± 11.4	89.5 ± 1.4	84.2 ± 13.0
houv	85.9 ± 4.9	84.1 ± 6.6	87.7 ± 1.7	82.7 ± 7.5	**88.3 ± 0.4**	84.6 ± 7.4
iono	66.6 ± 2.6	**75.2 ± 9.1**	**77.2 ± 6.6**	**73.2 ± 7.3**	68.3 ± 8.9	**70.1 ± 4.2**
isol	91.7 ± 4.0	**97.6 ± 1.3**	**98.3 ± 0.6**	**97.5 ± 1.3**	**98.3 ± 0.4**	**93.9 ± 4.6**
live	52.0 ± 4.5	51.4 ± 4.7	52.1 ± 3.4	52.4 ± 3.7	51.5 ± 3.3	52.0 ± 4.4
text	57.2 ± 3.4	51.8 ± 2.0	**64.2 ± 5.3**	56.4 ± 6.1	61.0 ± 11.0	**57.4 ± 3.4**
wdbc	77.6 ± 5.1	76.9 ± 8.2	79.8 ± 4.8	70.7 ± 5.1	**94.3 ± 0.5**	**78.2 ± 5.2**
dna	56.4 ± 4.5	52.5 ± 5.5	53.3 ± 0.4	53.3 ± 0.4	-	**56.6 ± 4.6**
glas	49.5 ± 5.3	**54.6 ± 3.8**	**54.5 ± 4.1**	52.0 ± 5.8	-	**52.7 ± 4.9**
iris	92.5 ± 2.6	**94.0 ± 3.5**	**95.0 ± 1.8**	92.6 ± 3.0	-	**93.5 ± 2.6**
usps	46.6 ± 4.5	**67.5 ± 7.4**	**79.7 ± 7.5**	**86.8 ± 4.9**	-	**49.3 ± 5.1**
vehi	38.5 ± 4.1	**50.4 ± 5.6**	**51.9 ± 5.0**	**49.9 ± 5.5**	-	**41.9 ± 5.2**
wine	93.7 ± 1.9	94.2 ± 1.8	94.5 ± 1.0	93.7 ± 3.0	-	93.9 ± 1.8
GSSL methods against 5NN: win/tie/loss		10/6/4	10/7/3	8/7/5	6/6/2	**13/7/0**

mentioning is that, all compared GSSL methods suffer much more serious issue on performance degeneration than that in Table 3. While the proposal SAGRAPH still does not decrease performance significantly in this situation.

We now summarize the results in Tables 2 and 3.

- Direct GSSL methods did decrease performance significantly in considerable cases, no matter for sparse (1NN) or denser (5NN) graphs. Our proposal SAGRAPH does not suffer from this issue in all the cases reported in Tables 2 and 3.
- SAGRAPH achieves the highest times in terms of *win* counts in both Table 2 and Table 3, showing that SAGRAPH owns a good ability in performance gain.
- In the aspect for the comparison between 1NN and 5NN graphs, GSSL methods with 5NN graph obtain more aggressive results (especially for CMN method), however, they also suffer from a more serious issue on performance degeneration.

4.3 Influence on Candidate Graphs

The situation of candidate graphs is one factor of our proposal. We further study the influence on the candidate graphs for SAGRAPH. Specifically, we generate the candi-

Table 4. Influence on the number of candidate graphs for the SAGRAPH method

Win Counts											
GFHF	CMN	LLGC	SGT	Number of candidate graphs in SAGRAPH							
				2	3	4	5	6	7	8	9	10
6	6	6	6	7	7	7	7	7	7	7	7	7

Loss Counts											
GFHF	CMN	LLGC	SGT	Number of candidate graphs in SAGRAPH							
				2	3	4	5	6	7	8	9	10
3	2	1	4	0	0	0	0	0	0	0	0	0

date graphs as follows. For each training instance, the number of nearest neighbors is randomly picked up from 1 to 9 with uniform distribution. In this case, 1NN, 3NN and 5NN graphs are special cases of these candidate graphs. The number of candidate graphs are set from 2 to 10. The experiments are conducted for 20 times and the win/loss counts against supervised 5NN method on binary classifications are shown in Table 4. The win/loss counts of compared GSSL methods are also listed for comparison. Table 4 shows that, SAGRAPH consistently works quite well on the number of candidate graphs, i.e., competitive win counts and much fewer loss counts. One reason is that the proposal only exploits some reliable subgraphs of candidate ones, rather than full graphs.

5 Conclusion and Future Work

In this paper we propose to learn a safe graph for graph-based SSL (GSSL), which could always outperform its supervised counterpart, i.e., classic nearest neighbor method. This is motivated by a crucial issue of GSSL that GSSL with the use of inappropriate graph construction may cause serious performance degeneration which could be even worse than its supervised counterpart. To overcome this issue, in this work we present an SAGRAPH method. The basic idea is that given a data distribution, there often exist some dense areas (or safe areas) which are quite robust or less sensitive to graph construction. One should exploits the data and the subgraphs in safe areas to learn a safe graph. We then consequently formulate the above consideration as a convex optimization and connect it to safe graph construction. Empirical studies on a number of data sets verify that our proposal achieves promising performance on the safeness of GSSL.

In our work achieving safe graph construction requires additional costs, e.g., more running time or smaller performance gain in some cases. One reason in that safe SSL needs to always take the safeness of SSL into account, whereas previous SSL studies do not have to take such kind of consideration and behave to be more aggressive. In future, we will study scalable safe GSSL as well as some other effective safe SSL approaches such as incorporating specific domain knowledge, e.g., known laws of physics [23].

Acknowledgement

The authors want to thank the reviewers for their helpful comments. This research was supported by the National Natural Science Foundation of China (61772262) and the Fundamental Research Funds for the Central Universities (020214380044).

References

1. A. Argyriou, M. Herbster and M. Pontil: Combining Graph Laplacians for Semi-Supervised Learning. Advances in Neural Information Processing Systems, pages 67-74, Cambridge, MA, 2005.
2. A. Balsubramani and Y. Freund: Optimally Combining Classifiers Using Unlabeled Data. Proceedings of International Conference on Learning Theory, pages 211-225, Paris, France, 2015.
3. M. Belkin and P. Niyogi: Towards a theoretical foundation for laplacian-based manifold methods. Journal of Computer and System Sciences, 74(8):1289-1308, 2008.
4. M. Belkin, P. Niyogi, and V. Sindhwani: Manifold regularization: a geometric framework for learning from labeled and unlabeled examples. Journal of Machine Learning Research, 7:2399-2434, 2006.
5. A. Blum and S. Chawla: Learning from labeled and unlabeled data using graph mincuts. In Proceedings of the 8th International Conference on Machine Learning, pages 19-26, Williamstown, MA, 2001.
6. L. Bottou, FE. Curtis and J. Nocedal: Optimization methods for large-scale machine learning. arXiv preprint arXiv:1606.04838, 2016
7. M. A . CarreiraPerpinan and R. S. Zemel: Proximity graphs for clustering and manifold learning. Advances in Neural Information Processing Systems, pages 225-232, Cambridge, MA, 2005.
8. O. Chapelle, B. Scholkopf, and A. Zien, editors: Semi-Supervised Learning. MIT Press, 2006.
9. L.-Z. Guo, Y.-F. Li: A general formulation for safely exploiting weakly supervised data. In Proceedings of the 32nd AAAI Conference on Artificial Intelligence, New Orleans, LA, 2018.
10. T. Jebara, J. Wang, and S. F. Chang: Graph construction and b-matching for semi-supervised learning. In Proceedings of the 26th International Conference on Machine Learning, pages 441-448, Montreal, Canada, 2009.
11. T. Joachims: Transductive learning via spectral graph partitioning. In Proceedings of the 20th International Conference on Machine Learning, pages 290-297, Washington, DC, 2003.
12. J. H. Krijthe and M. Loog: Implicitly constrained semi-supervised least squares classification. In Advances in 14th International Symposium on Intelligent Data Analysis, pages 158-169, Saint Etienne, France, 2015.
13. Y.-F. Li and Z.-H. Zhou: Towards making unlabeled data never hurt. In Proceedings of the 28th International Conference on Machine learning, pages 1081-1088, Bellevue, WA, 2011.
14. Y.-F. Li and Z.-H. Zhou: Towards making unlabeled data never hurt. IEEE Transactions on Pattern Analysis and Machine Intelligence, 37(1):175-188, 2015.
15. Y.-F. Li, S.-B. Wang and Z.-H. Zhou: Graph Quality Judgement: A Large Margin Expedition. In: Proceedings of the 25th International Joint Confernece on Artificial Intelligence, pages 1725-1731, New York, NY, 2016.
16. Y.-F. Li, J. Kwok and Z.-H. Zhou: Towards safe semi-supervised learning for multivariate performance measures. In: Proceedings of the 30th AAAI conference on Artificial Intelligence, Phoenix, AZ, 2016, pp. 1816-1822.

17. Y.-F. Li, H.-W. Zha and Z.-H. Zhou: Learning safe prediction for semi-supervised regression. In: Proceedings of the 31st AAAI conference on Artificial Intelligence, San Francisco, CA, 2017, pp.2217-2223.
18. D.-M. Liang and Y.-F. Li: Lightweight label propagation for large-scale network data. In: Proceedings of the 27th International Joint Conference on Artificial Intelligence, Stockholm, Sweden, 2018.
19. L.-Z. Guo and Y.-F. Li: A general formulation for safely exploiting weakly supervised data. In: Proceedings of the 32nd AAAI conference on Artificial Intelligence, New Orleans, LA, 2018.
20. W. Liu, J. Wang, and S.F. Chang: Robust and scalable graph-based semisupervised learning. Proceedings of the IEEE, 100(9):2624-2638, 2012.
21. Y. Nesterov: Introductory Lectures on Convex Optimization. A Basic Course. Springer, 2003.
22. Niu, G., du Plessis, M. C., Sakai, T., Ma, Y. and Sugiyama, M: Theoretical Comparisons of Positive-Unlabeled Learning against Positive-Negative Learning. Advances in Neural Information Processing Systems, pages 1199-1207, Barcelona, Spain, 2016.
23. R. Stewart and S. Ermon: Label-free supervision of neural networks with physics and domain knowledge. In Proceedings of 31th AAAI Conference on Artificial Intelligence, San Francisco, CA, 2017.
24. F.Wang and C. Zhang: Label propagation through linear neighborhoods. In Proceedings of the 23rd International Conference on Machine Learning, pages 985-992, Pittsburgh, PA, 2006.
25. F. Wang and C. Zhang: Label propagation through linear neighborhoods. IEEE Transactions on Knowledge and Data Engineering, 20(1):55-67, 2008.
26. Y.-Y. Wang, S.-C. Chen, Z.-H. Zhou: New Semi-Supervised Classification Method Based on Modified Cluster Assumption: IEEE Transactions on Neural Network and Learning System, 23(5):689-702, 2012
27. T. Wei, L.-Z. Guo, Y.-F. Li, W. G.: Learning safe multi-label prediction for weakly labeled data. Machine Learning. 107(4): 703-725, 2018.
28. D. Zhou, O. Bousquet, T. Navin Lal, J. Weston, and B. Scholkopf: Learning with local and global consistency. Advances in Neural Information Processing Systems, pages 595-602. MIT Press, Cambridge, MA, 2004.
29. X. Zhu, Z. Ghahramani, and J. Lafferty: Semi-supervised learning using Gaussian fields and harmonic functions. In Proceedings of the 20th International Conference on Machine learning, pages 912-919, Washington, DC, 2003.
30. X. Zhu, J. Kandola, Z. Ghahramani, and J. Lafferty: Nonparametric transforms of graph kernels for semi-supervised learning. Advances in Neural Information Processing Systems, pages 1641-1648. MIT Press, Cambridge, MA, 2005.
31. X. Zhu: Semi-supervised learning literature survey. Technical report, University of Wisconsin-Madison, 2007.
32. Z.-H. Zhou: A brief introduction to weakly supervised learning. National Science Review, 5(1):44-53, 2018.