Learning from Labeled and Unlabeled Vertices in Networks

Wei Ye[†], Linfei Zhou[†], Dominik Mautz[†], Claudia Plant[§], Christian Böhm[†]
[†]Ludwig-Maximilians-Universität München, Munich, Germany
[§]University of Vienna, Vienna, Austria
{ye,zhou,mautz,boehm}@dbs.ifi.lmu.de
claudia.plant@univie.ac.at

ABSTRACT

Networks such as social networks, citation networks, protein-protein interaction networks, etc., are prevalent in real world. However, only very few vertices have labels compared to large amounts of unlabeled vertices. For example, in social networks, not every user provides his/her profile information such as the personal interests which are relevant for targeted advertising. Can we leverage the limited user information and friendship network wisely to infer the labels of unlabeled users?

In this paper, we propose a semi-supervised learning framework called weighted-vote Geometric Neighbor classifier (wvGN) to infer the likely labels of unlabeled vertices in sparsely labeled networks. wvGN exploits random walks to explore not only local but also global neighborhood information of a vertex. Then the label of the vertex is determined by the accumulated local and global neighborhood information. Specifically, wvGN optimizes a proposed objective function by a search strategy which is based on the gradient and coordinate descent methods. The search strategy iteratively conducts a coarse search and a fine search to escape from local optima. Extensive experiments on various synthetic and real-world data verify the effectiveness of wvGN compared to state-of-the-art approaches.

KEYWORDS

Semi-supervised learning; Geometric neighborhood; Random walk; Support vector machines; Regularization

1 INTRODUCTION

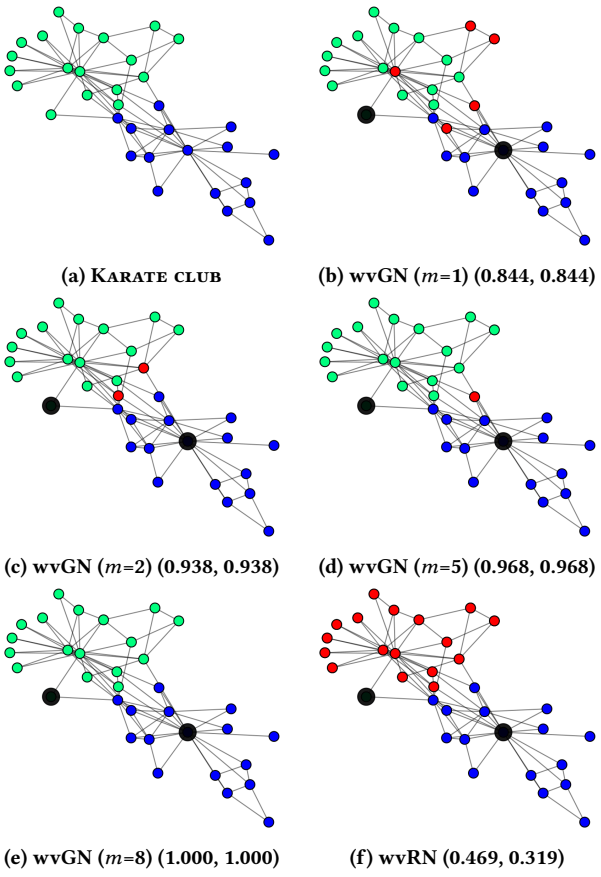
The problem of learning from labeled and unlabeled data, literarily attributed to semi-supervised learning, has aroused considerable interests in recent years. Generally, labeled data is scarce while unlabeled data is abundant. Labeling data can be very tedious, time-consuming and expensive because it needs the efforts of skilled human annotators. How to effectively make use of unlabeled data to improve learning performance is of great practical significance. Recently, various semi-supervised learning methods have been proposed, such as TSVM [12, 27], LapSVM [1] and LGC [31]. Both LapSVM and LGC are based on the assumption of local and global label consistency in networks. Although LapSVM and LGC can be directly applied on sparsely labeled networks, their performance is not satisfying when their assumptions are not met.

Differing from conventional data which is assumed to be i.i.d (independent and identically distributed), networked data, extracted from social media, bibliographic databases and etc., is interdependent. Vertices connected to each other are likely to have the same labels according to the principle of homophily [20]. Relational learning [9, 18] has been proposed to capture the correlations between connected vertices. It makes a first-order Markov assumption to infer labels, i.e., the label of a vertex is determined by those of its direct neighbors in the network. In the prediction process, collective inference is used to find an equilibrium state such that the inconsistency between neighboring vertices is minimized. Relational learning methods do not perform well when unlabeled vertices have too few labeled neighbors to support learning and/or inference [8]. The question is how to infer the labels of unlabeled vertices when they have too few labeled neighbors? Our idea is to use the random walk to capture both short and long distance relationships in networks. To be specific, if a vertex has too few labeled neighbors, we can exploit its one- to m-hop (See Section 3) neighbors to support the learning process.

To motivate the problem, we use a real network called Karate Club [30] shown in Figure 1(a). The network records 78 pairwise interactions (links) between 34 members of a karate club who interacted outside the club. The network is partitioned into two communities. The blue community contains 16 members and the green community contains 18 members. For the test, the labels of the two vertices in black are given and the labels of the other vertices are unknown to all the methods. Figure 1(b) shows the result of our method wvGN only using the geometric one-hop neighborhood information (the definition is given in Section 3.1). Five members are misclassified, which leads to a Micro-F1 score (the averaged F1 score over classes, please refer to [7, 21, 29] for more details) of 0.844; Figure 1(c) gives the result of wvGN using the geometric one- and two-hop neighborhood information. Two members are misclassified, which leads to a Micro-F1 score of 0.938; Figure 1(d) shows the result of wvGN using the geometric one- to five-hop neighborhood information. One member is misclassified, which leads to a Micro-F1 score of 0.968. When using the geometric one-to eight-hop neighborhood information, wvGN correctly classifies all the members (Figure 1(e)). With the accumulation of both local and global neighborhood information, the learning performance of wvGN is strengthened. Figure 1(f) demonstrates the result of wvRN (weighted-vote Relational Neighbor classifier) [17] (the results of LapSVM and LGC are the same and not shown here to reduce the

KDD'17, August 13–17, 2017, Halifax, NS, Canada.
 ISBN 978-1-4503-4887-4/17/08…$15.00
DOI: http://dx.doi.org/10.1145/3097983.3098142

clutter). We can see that wvRN fails in such an occasion when the network is sparsely labeled.



(a) Karate club (b) wvGN (m=1) (0.844, 0.844)

(c) wvGN (m=2) (0.938, 0.938) (d) wvGN (m=5) (0.968, 0.968)

(e) wvGN (m=8) (1.000, 1.000) (f) wvRN (0.469, 0.319)

Figure 1: Classification results on Karate Club. The two labeled vertices are in black and the misclassified vertices are in red. m is the number of hops of a random walker. Subcaption: Method (Micro-F1, Macro-F1).

To achieve better classification results in sparsely labeled networks, our method wvGN exploits not only local but also global neighborhood information of vertices. Specifically, the label of a vertex is jointly determined by its geometric one-hop to m-hop ($m = 1, 2, \ldots$) neighbors in the network. The geometric m-hop neighbors of a vertex are those vertices which are m-hops away by a random walker. Our main contributions are as follows:

- We use random walks with arbitrary m hops to accumulate local and global neighborhood information for better classification. m is implicitly determined by the power iteration method.
- We formulate the semi-supervised learning in networks as an optimization problem and propose an objective function based on the L2-loss SVM.
- We propose a search strategy called gradient and coordinate descent (GCD) to optimize the objective function. GCD is a combination of the gradient descent (GD) and coordinate descent (CD) methods. GD is used for a coarse search and CD is used for a fine search. GCD does not easily get trapped in local optima.
- We show the effectiveness of our method wvGN by carrying out exhaustive comparative studies with state-of-the-art methods from various related domains.

2 NOTATIONS AND PROBLEM FORMULATION

We use lower-case Roman letters (e.g. a, b) to denote scalars. We denote vectors (row) by boldface lower case letters (e.g. $\mathbf{x}$). Matrices are denoted by boldface upper case letters (e.g. $\mathbf{X}$). We denote entries in a matrix by non-bold lower case letters, such as $x_{i,j}$. Row i of matrix $\mathbf{X}$ is denoted by the vector $\mathbf{x}_i$. We use $[x_1, \cdots, x_n]$ to denote a vector created by stacking n scalars; similarly, we use $\mathbf{X} = [\mathbf{x}_1; \ldots; \mathbf{x}_n]$ to denote creating a matrix by stacking the vector $\mathbf{x}_i$ along the rows. A set is denoted by calligraphic capital letters (e.g. $\mathcal{S}$). A network is denoted by $\mathrm{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, \ldots, v_n\}$ is a set of vertices and $\mathcal{E}$ is a set of edges. The affinity matrix of vertices is denoted by $\mathbf{A} \in \mathbb{R}^{n \times n}$ with $a_{i,j} = a_{j,i}, a_{i,i} = 0$. The degree matrix $\mathbf{D}$ is a diagonal matrix associated with $\mathbf{A}$ with $d_{i,i} = \sum_j a_{j,i}$. The random walk transition matrix $\mathbf{P}$ is defined as $\mathbf{D}^{-1}\mathbf{A}$. For a matrix $\mathbf{X} \in \mathbb{R}^{n \times n}$, $diag(\mathbf{X}) \in \mathbb{R}^{n \times 1}$ is a vector created by extracting the diagonal of $\mathbf{X}$. We consider the following semi-supervised learning problem in networks:

Semi-Supervised Learning in Networks. *Given a network $\mathrm{G} = (\mathcal{V}, \mathcal{E})$ with vertices $\{v_1, \ldots, v_l\}$ labeled as $\mathbf{y}_l = [y_1, \ldots, y_l], l \ll n, y_i (1 \leqslant i \leqslant l) \in \{+1, -1\}$ and vertices $\{v_{l+1}, \ldots, v_n\}$ unlabeled. The goal is to learn a classifier to infer the labels $\hat{\mathbf{y}}_u = [\hat{y}_{l+1}, \ldots, \hat{y}_n]$ of the unlabeled vertices.*

3 WEIGHTED-VOTE GEOMETRIC NEIGHBOR CLASSIFIER

3.1 The Model

To classify the vertices in a network, in this work, we first transform vertices from network space to vector space. For a network, we define its geometric one-hop neighborhood as follows:

Definition 3.1. **Geometric One-hop Neighborhood.** The geometric one-hop neighbors of a vertex v_i is defined as a set $\mathcal{N}_i^1$ which contains those vertex v_j which can be reached by a random walker from v_i in one step. The geometric one-hop neighborhood is defined as a set $\mathcal{N}^1 = \bigcup_{i=1}^{l} \mathcal{N}_i^1$.

We denote a vertex v_q in the geometric one-hop neighborhood $\mathcal{N}^1$ by $\mathbf{p}_q$ which is the q^{th} row of the transition matrix $\mathbf{P}$. The class indicator score of the vertex v_q is defined as follows:

$$f(\mathbf{p}_q) = \mathbf{p}_q \cdot \mathbf{w}^\intercal + b \tag{1}$$

where the weight vector $\mathbf{w}$ and bias term b are the parameters to be learned.

Then the label of the vertex v_q is inferred by the following formula:

$$y_q = sign(f(\mathbf{p}_q)) = sign(\mathbf{p}_q \cdot \mathbf{w}^\intercal + b) \tag{2}$$

If we want to seek the separating hyperplane with the largest margin for the positive and negative samples, Equation (1) becomes the problem of linear support vector machines (SVM). However,

Equation (1) does not take the neighborhood relationship into consideration to classify vertices in networks, which would lead to the deterioration of the classifier. The theory of homophily [20] tells us that vertices connected to each other tend to have the same labels. For example, friends connected in a social network are likely to have similar interests; papers connected in a citation network are likely to have similar topics. Thus to classify the vertex v_q, we need to consider its neighborhood information. A simple relational neighbor classifier (RN) [17] estimates the class-membership probability of the vertex v_q by $P(\text{class}|v_q) = \frac{1}{Z}\sum_{\text{label}(v_i)=\text{class}|i\in\mathcal{N}_q^1} a_{i,q}$, where $Z = \sum_{i\in\mathcal{N}_q^1} a_{i,q}$. However, in some cases, it is still hard to determine the label of a vertex according to the theory of homophily. For example in Figure 2, the vertex v_5 is connected to three vertices in the green class and also connected to three vertices in the yellow class. Since $P(\text{yellow}|v_5) = P(\text{green}|v_5) = 0.5$, what is the label of the vertex v_5?

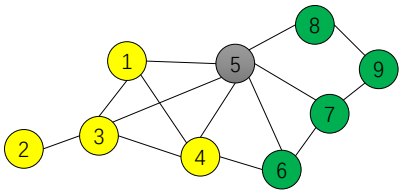


Figure 2: An example network with the vertex v_5 unlabeled.

In this work, we use a weighted-vote strategy to integrate the neighborhood information of the vertex v_q with Equation (1) as follows:

$$\begin{aligned} f(\mathbf{p}_q) &= \mathbf{p}_q \cdot \mathbf{w}^\top + b > 0 \\ y_q &= 1 \\ \text{if } & \frac{\sum_{i\in\mathcal{N}_q^1} a_{i,q} f(\mathbf{p}_i)}{\sum_{i\in\mathcal{N}_q^1} a_{i,q}} \geq +1 \end{aligned} \tag{3}$$

Similarly, if the weighted indicator score is less than or equal to -1, we have:

$$\begin{aligned} f(\mathbf{p}_q) &= \mathbf{p}_q \cdot \mathbf{w}^\top + b < 0 \\ y_q &= -1 \\ \text{if } & \frac{\sum_{i\in\mathcal{N}_q^1} a_{i,q} f(\mathbf{p}_i)}{\sum_{i\in\mathcal{N}_q^1} a_{i,q}} \leq -1 \end{aligned} \tag{4}$$

Inequalities (3) and (4) can be combined into one set of inequalities:

$$\begin{aligned} & y_q \cdot \frac{\sum_{i\in\mathcal{N}_q^1} a_{i,q}(\mathbf{p}_i \cdot \mathbf{w}^\top + b)}{\sum_{i\in\mathcal{N}_q^1} a_{i,q}} \geq 1 \\ \Rightarrow & y_q \cdot \left(\frac{\sum_{i\in\mathcal{N}_q^1} a_{i,q} \cdot \mathbf{p}_i}{d_{q,q}} \cdot \mathbf{w}^\top + b \right) \geq 1 \\ \Rightarrow & y_q \cdot \left(\frac{\mathbf{a}_q \cdot \mathbf{P}}{d_{q,q}} \cdot \mathbf{w}^\top + b \right) \geq 1 \\ \Rightarrow & y_q \cdot \left(\mathbf{p}_q \cdot \mathbf{P} \cdot \mathbf{w}^\top + b \right) \geq 1 \end{aligned} \tag{5}$$

The above inequality is just for the classification of the vertex v_q using its geometric one-hop neighborhood information. However, in real networks, the vertex may be sparsely labeled. For example in Figure 3, how to infer the class label of the vertex v_5 when its directly connected neighbors are unlabeled? The relational neighbor classifier (RN) [17] iteratively classifies vertices using its previously inferred labels. However, if there is an error inference, the error will be amplified in the subsequent inference procedure.

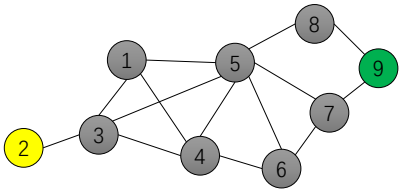


Figure 3: An example network with sparsely labeled vertices.

Differing from RN which only uses the class labels of known related instances without doing learning, our method wvGN learns geometic neighborhood information from data. The geometric m-hop ($m \geq 2$) neighborhood is defined as follows:

Definition 3.2. **Geometric m-hop Neighborhood.** The geometric m-hop neighbors of a vertex v_i is defined as a set $\mathcal{N}_i^m$ which contains those vertex v_j which can be reached by a random walker from v_i in m steps. The geometric m-hop neighborhood is defined as a set $\mathcal{N}^m = \bigcup_{i=1}^{l} \mathcal{N}_i^m$.

In the geometric m-hop neighborhood $\mathcal{N}_i^m$, v_q is denoted by $\mathbf{p}_q^m$ which is the q-th row of the transition matrix $\mathbf{P}^m$. If we replace $\mathbf{p}_q$ in the Inequality (5) with $\mathbf{p}_q^m$, we have:

$$\begin{aligned} & y_q \cdot \frac{\sum_{i\in\mathcal{N}_q^m} a_{i,q}(\mathbf{p}_i^m \cdot \mathbf{w}^\top + b)}{\sum_{i\in\mathcal{N}_q^m} a_{i,q}} \geq 1 \\ \Rightarrow & y_q \cdot \left(\frac{\sum_{i\in\mathcal{N}_q^m} a_{i,q} \cdot \mathbf{p}_i^m}{d_{q,q}} \cdot \mathbf{w}^\top + b \right) \geq 1 \\ \Rightarrow & y_q \cdot \left(\frac{\mathbf{a}_q \cdot \mathbf{P}^m}{d_{q,q}} \cdot \mathbf{w}^\top + b \right) \geq 1 \\ \Rightarrow & y_q \cdot \left(\mathbf{p}_q \cdot \mathbf{P}^m \cdot \mathbf{w}^\top + b \right) \geq 1 \\ \Rightarrow & y_q \cdot \left(\mathbf{p}_q^m \cdot \mathbf{P} \cdot \mathbf{w}^\top + b \right) \geq 1 \\ \Rightarrow & y_q \cdot \left(\mathbf{p}_q^{m+1} \cdot \mathbf{w}^\top + b \right) \geq 1 \end{aligned} \tag{6}$$

where $\mathbf{p}_q^{m+1}$ is the representation of v_q in the geometric $(m+1)$-hop neighborhood.

Now the label of the vertex v_5 is still determined by its directly connected neighbors in the geometric m-hop neighborhood where each neighbor has integrated information from both labeled vertices v_2 and v_9. We can see from Equations (5) and (6) that in geometric neighborhood $\mathcal{N}^1 \cup \cdots \mathcal{N}^m$, a vertex is always transformed to its next-hop neighborhood by the transition matrix $\mathbf{P}$. Similar to SVM [3, 6], we introduce a positive slack variable ξ_q in the constraints and the Inequality (6) becomes:

$$\begin{aligned} & y_q \cdot \left(\mathbf{p}_q^{m+1} \cdot \mathbf{w}^\top + b \right) \geq 1 - \xi_q \\ & \xi_q \geq 0,\ 1 \leq q \leq l,\ m \geq 1 \end{aligned} \tag{7}$$

Compared with the large amount of unlabeled data, the amount of labeled data is limited in the semi-supervised learning scenario. In addition, as we said before, it is hard to infer the label of a vertex if it has two few labeled neighbors. To alleviate these situations, we accumulate the information from every geometric neighborhood and represent the vertex v_q by $\mathbf{x}_q = \mathbf{p}_q^2 + \cdots + \mathbf{p}_q^{m+1} = \mathbf{p}_q \cdot (\mathbf{P} + \cdots + \mathbf{P}^m)$. The unconstrained SVM problem with L2 loss is formulated as follows:

$$\min_{\mathbf{w},b} \quad \frac{1}{2}\mathbf{w}\cdot\mathbf{w}^\intercal + \frac{\alpha}{l}\sum_{q=1}^{l}\max(1 - y_q \cdot f(\mathbf{x}_q), 0)^2 \tag{8}$$

Note that $\mathbf{x}_q$ can be considered as a random walker taking $m+1 (m \geq 1)$ hops from the vertex v_q and accumulating the neighborhood information at each hop. With higher values of m, $\mathbf{x}_q$ becomes more global. Meanwhile, $\mathbf{x}_q$ becomes identical, because the transition probability tends to converge towards the stationary distribution which is not useful in classification. To avoid the situation where all vertices have identical representation, we introduce a dampening factor at each hop such that higher hops have higher penalty and thus decay faster. Inspired from the heat kernel [5], the dampening factor for the m^{th} hop is defined as $\rho^m/m!$, where ρ is a non-negative parameter (considered as the temperature in the heat kernel).

In this work, we penalize $\mathbf{w}_i$ by $d_{i,i}^{-\frac{1}{2}}$. For mathematical convenience, we extend each instance $\mathbf{x}_q$ as $\left[\mathbf{x}_q, 1\right]$ and $\mathbf{w}$ as $[\mathbf{w}, b]$. The objective function (8) now becomes:

$$\min_{\mathbf{w}} \quad F(\mathbf{w}) = \frac{\lambda}{2}(\mathbf{w}\odot\mathbf{d})(\mathbf{w}\odot\mathbf{d})^\intercal + \frac{\alpha}{l}\sum_{q=1}^{l}\max(1 - y_q\mathbf{x}_q\mathbf{w}^\intercal, 0)^2 \tag{9}$$

where $\odot$ means the Hadamard product, λ and α are regularization parameters, $\mathbf{x}_q = \left[\mathbf{p}_q \cdot \left(\frac{\rho^1}{1!}\mathbf{P} + \cdots + \frac{\rho^m}{m!}\mathbf{P}^m\right), 1\right]$, $\mathbf{X} = [\mathbf{x}_1; \ldots; \mathbf{x}_n]$ and $\mathbf{d} = \left[diag(\mathbf{D}^{-\frac{1}{2}})^\intercal, 1\right]$.

3.2 Optimization

Stochastic gradient descent (SGD) algorithm is very successful in training large-scale SVM [24] on sparse data. However, we find it gets easily trapped in local optima because the representation of each vertex in vector space is not sparse. So do the gradient descent (GD) and coordinate descent (CD) methods (See Section 4.1) because they are dependent on the starting point. In this work, we use a combination of GD and CD to optimize our objective function given in (9). Specifically, we first conduct a coarse search by GD owing to its fast convergence and then conduct a fine search by CD. The gradient of (9) with respect to $\mathbf{w}$ is:

$$F'(\mathbf{w}) = \lambda\mathbf{w}\odot\mathbf{d} - \frac{2\alpha}{l}\sum_{j\in\mathbb{I}(\mathbf{w})} y_j\mathbf{x}_j b_j(\mathbf{w}) \tag{10}$$

where $b_j(\mathbf{w}) = 1 - y_j\mathbf{x}_j\mathbf{w}^\intercal$ and $\mathbb{I}(\mathbf{w}) = \left\{j | b_j(\mathbf{w}) > 0\right\}$.

We iteratively update $\mathbf{w}$ as follows:

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \eta_t F'(\mathbf{w}^t) \tag{11}$$

where η_t is the learning rate at the t^{th} iteration and is chosen from $\left\{1, \beta, \beta^2, \ldots\right\}$ by a line search.

We can see that the learning rate η_t for each element in $\mathbf{w}$ is the same at each iteration, which leads to GD's easily getting trapped in local optima. To let GD escape from local optima, we conduct a fine search by CD. CD finds an appropriate learning rate for each element. The CD method has been successfully applied for solving large-scale L2-loss SVM [4]. The method starts from an initial vector $\mathbf{w}^0$ (the final output of GD in this work) and iteratively generates a sequence $\left\{\mathbf{w}^t\right\}$ $(t = 0, 1, 2, \ldots)$. At each iteration, $\mathbf{w}^{t+1}$ is produced by sequentially updating each entry of $\mathbf{w}^t$ with other entries fixed. The process produces a sequence of vectors $\mathbf{w}^{t,i}$ $(i = 1, \ldots, n+1)$, such that $\mathbf{w}^{t,0} = \mathbf{w}^t$, $\mathbf{w}^{t,n+1} = \mathbf{w}^{t+1}$ and

$$\mathbf{w}^{t,i} = \left[w_1^{t+1}, \ldots, w_i^{t+1}, w_{i+1}^t, \ldots, w_{n+1}^t\right]$$

Updating $\mathbf{w}^{t,i}$ to $\mathbf{w}^{t,i+1}$ becomes the following one-variable sub-problem:

$$\begin{aligned}
&\min_z \quad F_i(w_1^{t+1}, \ldots, w_i^{t+1}, w_{i+1}^t + z, w_{i+2}^t \ldots, w_{n+1}^t) \\
\equiv &\min_z \quad F_i(\mathbf{w}^{t,i} + z\mathbf{e}_i) \\
= &\min_z \quad \frac{\lambda}{2}\left((\mathbf{w}^{t,i} + z\mathbf{e}_i)\odot\mathbf{d}\right)\left((\mathbf{w}^{t,i} + z\mathbf{e}_i)\odot\mathbf{d}\right)^\intercal \\
&\quad + \frac{\alpha}{l}\sum_{j\in\mathbb{I}(\mathbf{w}^{t,i}+z\mathbf{e}_i))}(b_j(\mathbf{w}^{t,i} + z\mathbf{e}_i))^2
\end{aligned} \tag{12}$$

where $\mathbf{e}_i \in \mathbb{R}^{1\times(n+1)}$ is a vector with the i^{th} entry 1 and all other entries 0.

The first derivative of (12) with respect to z is:

$$F_i'(z) = \lambda\left(w_i^{t,i} + z\right)\cdot\mathbf{d}_i - \frac{2\alpha}{l}\sum_{j\in\mathbb{I}(\mathbf{w}^{t,i}+z\mathbf{e}_i))}(y_j x_{j,i}(b_j(\mathbf{w}^{t,i} + z\mathbf{e}_i)) \tag{13}$$

As pointed out in [4], $F_i(z)$ is not twice differentiable at some j, where $b_j(\mathbf{w}^{t,i} + z\mathbf{e}_i) = 0$. Following [4, 19], we define the generalized second derivative of (12) with respect to z as:

$$F_i''(z) = \lambda\mathbf{d}_i + \frac{2\alpha}{l}\sum_{j\in\mathbb{I}(\mathbf{w}^{t,i}+z\mathbf{e}_i))} x_{j,i}^2 \tag{14}$$

The Newton direction at a given z is $\frac{F_i'(z)}{F_i''(z)}$. We start from $z = 0$ and apply a line search $z = z - \eta_i\frac{F_i'(z)}{F_i''(z)}$ until $F_i(z - \eta_i\frac{F_i'(z)}{F_i''(z)}) < F_i(z)$, where η_i is the learning rate for the i^{th} element and is chosen from $\left\{1, \beta, \beta^2, \ldots\right\}$.

3.3 Implementation Details and Analysis

For each vertex, the random walk takes arbitrary m hops. In the following, we use the power iteration method to **implicitly** decide the value of m for each vertex. Note that different vertices may have different numbers of hops and **we do not explicitly compute $\mathbf{P}^m$** owing to its high time complexity. If we denote the m-th term $\mathbf{p}_q \cdot \frac{\rho^m}{m!}\mathbf{P}^m$ in $\mathbf{x}_q$ by $\mathbf{v}_q^m$, then $\mathbf{v}_q^m = \frac{\rho}{m}\mathbf{v}_q^{m-1}\cdot\mathbf{P}$. We define the velocity at $m-1$ to be the vector $\boldsymbol{\delta}_q^{m-1} = \mathbf{v}_q^m - \mathbf{v}_q^{m-1}$, the acceleration at $m-1$ to be the vector $\boldsymbol{\epsilon}_q^{m-1} = \boldsymbol{\delta}_q^m - \boldsymbol{\delta}_q^{m-1}$ and stop the iteration when $\|\boldsymbol{\epsilon}_q^{m-1}\|_{max}$ is below a threshold $\hat{\epsilon}$. We set $\rho = 5$ according to [13], $\lambda = 2^{-6}$ according to [21], and $\alpha = 1, \beta = \frac{1}{2}$ according to [4]. The pseudo-code for our binary classifier wvGN is given in Algorithm 1.

In Algorithm 1, steps 4–14 are the details of using the power iteration method to compute the representation $\mathbf{x}_q$ of the vertex v_q. Note that this part can be parallelized since the representation computation for each vertex is independent. Step 16 uses our GCD to optimize (9). As described in Procedure (lines 19–25), our GCD method iteratively conducts GD and CD to escape from local optima. Given the randomly generated initial $\mathbf{w}$, we first apply a coarse search by GD. If the change of the objective (9) is less than a threshold (10^{-4}), we apply a fine search by CD; otherwise, we continue the coarse search by GD. We repeat the process until the maximum iteration is reached. In the optimization process, we find CD costs a lot of time. Inspired by the mini-batch SGD [24], we use CD to update only a number $k = 10\%$ of the elements randomly selected in $\mathbf{w}$. We find such a mini-batch CD method accelerates the search but does not deteriorate the results (see Section 4.1).

Algorithm 1: wvGN

Input: Affinity matrix $\mathbf{A}$ for a network and labels $\mathbf{y}_l = [y_1, \ldots, y_l]$ for the labeled vertices $\{v_1, \ldots, v_l\}$
Output: estimated $\hat{\mathbf{y}}_u = [\hat{y}_{l+1}, \ldots, \hat{y}_n]$ for the unlabeled vertices $\{v_{l+1}, \ldots, v_n\}$

1 $\rho \leftarrow 5, t \leftarrow 50, \hat{\epsilon} \leftarrow 10^{-4}, \lambda \leftarrow 2^{-6}, \alpha \leftarrow 1, \beta \leftarrow \frac{1}{2}$;
2 compute the degree matrix $\mathbf{D}$;
3 compute the transition matrix $\mathbf{P} \leftarrow \mathbf{D}^{-1}\mathbf{A}$;
/* power iteration */
4 **for** $j \leftarrow 1$ **to** n **do**
5 $\quad \mathbf{v}_j^1 \leftarrow \mathbf{p}_j \frac{\rho^1}{1!} \cdot \mathbf{P}$;
6 $\quad m \leftarrow 1$;
7 $\quad \mathbf{x}_j \leftarrow \mathbf{v}_j^1$;
8 $\quad$ **repeat**
9 $\quad\quad \mathbf{v}_j^{m+1} \leftarrow \frac{\rho}{m+1}\mathbf{v}_j^m \cdot \mathbf{P}$;
10 $\quad\quad \mathbf{x}_j \leftarrow \mathbf{x}_j + \mathbf{v}_j^{m+1}$;
11 $\quad\quad \boldsymbol{\delta}^m \leftarrow |\mathbf{v}_j^{m+1} - \mathbf{v}_j^m|$;
12 $\quad\quad m \leftarrow m + 1$;
13 $\quad$ **until** $\|\boldsymbol{\delta}_j^{m+1} - \boldsymbol{\delta}_j^m\|_{\max} \le \hat{\epsilon}$ **or** $m \ge t$;
14 $\quad \mathbf{x}_j \leftarrow [\mathbf{x}_j, 1]$;
15 $\mathbf{X} \leftarrow [\mathbf{x}_1; \ldots; \mathbf{x}_n], \mathbf{d} \leftarrow \left[diag(\mathbf{D}^{-\frac{1}{2}})^\intercal, 1\right]$;
16 $\mathbf{w} \leftarrow \texttt{GCD}(\mathbf{X}_l, \mathbf{y}_l, \mathbf{d}, \lambda, \alpha, \beta)$; /* $\mathbf{X}_l$ is the labeled data (training data). */
17 $\hat{\mathbf{y}}_u \leftarrow sign(\mathbf{X}_u \cdot \mathbf{w}^\intercal)$; /* $\mathbf{X}_u$ is the unlabeled data (test data). */
18 return $\hat{\mathbf{y}}_u$;
19 **Procedure** $\texttt{GCD}(\mathbf{X}_l, \mathbf{y}_l, \mathbf{d}, \lambda, \alpha, \beta)$
20 $\quad iter \leftarrow 25$, $\mathbf{w} \leftarrow$rand (1,n), $\mathbf{w} \leftarrow \frac{\mathbf{w}}{\|\mathbf{w}\|_2}$;
21 $\quad$ **for** $i \leftarrow 1$ **to** $iter$ **do**
22 $\quad\quad \mathbf{w} \leftarrow \texttt{GD}(\mathbf{w}, \mathbf{X}_l, \mathbf{y}_l, \mathbf{d}, \lambda, \alpha, \beta)$;
23 $\quad\quad$ **if** $\Delta F(\mathbf{w}) < 10^{-4}$ **then**
24 $\quad\quad\quad \mathbf{w} \leftarrow \texttt{miniBatchCD}(\mathbf{w}, \mathbf{X}_l, \mathbf{y}_l, \mathbf{d}, \lambda, \alpha, \beta)$;
25 $\quad$ return $\mathbf{w}$;

Complexity analysis. Assume that a network has n vertices and r edges. In Algorithm 1, lines 1–3 costs $O(r)$ time. Lines 4–14 adopts the power iteration method to approximately compute the representation for each vertex. Since the time complexity of the power iteration method is $O(r)$ [16], we need $O(n \cdot r)$ to finish steps 4–14. Procedure (lines 19–25) gives the pseudo-code for our GCD method. Line 22 uses GD to update $\mathbf{w}$. The time complexity of GD is bounded by the complexity of computing the gradient (Equation (10)), i.e., $O(l \cdot (n+1))$, where $l \ll n$ is the number of the labeled vertices. Line 23 computes $\Delta F(\mathbf{w})$, whose time complexity is $O(l \cdot (n+1))$. Line 24 updates $\mathbf{w}$ by the mini-batch CD, whose time complexity is determined by the first derivative (Equation (13)). It costs $O(l \cdot (n+1))$ to update one element in $\mathbf{w}$. Thus the time complexity to update k elements in $\mathbf{w}$ is $O(k \cdot l \cdot (n+1))$. The total time complexity of GCD is bounded by $O(k \cdot l \cdot n)$. Finally, the time complexity of our method wvGN is $O(n \cdot (r + k \cdot l))$.

4 EXPERIMENTAL EVALUATION

Our experiments evaluate the classification performance of wvGN and its competitors on synthetic and real-world data. We compare wvGN with state-of-the-art methods from related research fields. To be specific, the baseline methods are as follows:

- Semi-supervised learning. TSVM [12, 27], LapSVM [1] and LGC [31]. We use the rows of the transition matrix $\mathbf{P}$ to represent vertices and input them to TSVM and LapSVM.
- Relational learning. wvRN [17], SocDim [25] and SCRN [29].
- Random walk based network learning. Deepwalk [22], node2vec [10] and SNBC [21].
- Graph diffusion based learning. Heat kernel diffusion [5]. We use the power iteration method to approximately compute the heat kernel.

For more descriptions on those competitors, please refer to our related work and their original papers. For the datasets that have more than two classes, we use the one-vs-rest [2] method to train wvGN for each class, which leads to c (the number of classes) decision values for each vertex. However, the decision values generated by each binary wvGN cannot be compared directly. According to [21], we use Platt's Scaling [23] to transform these decision values to probability scores which are based on the same scale and can be compared directly. We assign the most probable class label to each vertex. To validate results, we use two popular evaluation measures from [7, 21, 29]: Micro-F1 score and Macro-F1 score. The higher the values of these evaluation measures, the better the classification.

We randomly sample a number of vertices with labels from each class as training data and use the rest of vertices as test data. Following the settings in [10, 22, 25], we repeat this process ten times and report the average and standard deviation of Micro-F1 score and Macro-F1 score for each method. The default parameters are adopted for the baseline methods according to their original papers. All experiments are run on the same machine with an Intel Core Quad i7-3770 with 3.4 GHz and 32 GB RAM. All networks shown in this work are plotted by the python toolbox *NetworkX*. The code of wvGN and all the synthetic and real-world data used in this work are available at the website[1].

[1] https://github.com/yeweiysh/wvGN

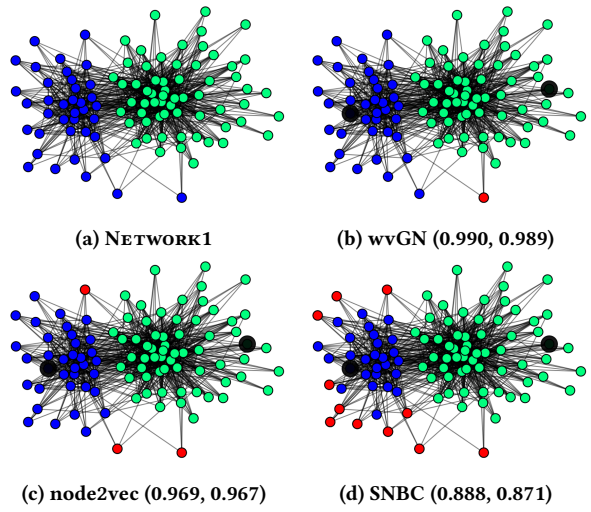
4.1 Synthetic Data

We use the benchmark graph generator [14] to generate two synthetic networks Network1 and Network2. The generator makes the vertex degree and community size follow power law distributions which reflect the real properties of vertices and communities found in real networks. Network1 has 100 vertices and 1008 edges, which are grouped into two classes (37 and 63 vertices, respectively). The average degree is 21. Network2 has 120 vertices and 607 edges, which are grouped into three classes (19, 43 and 58 vertices, respectively). The average degree is 10. For each network, we vary the number of labeled vertices in each class from one to five and report the average and standard deviation of Micro-F1 and Macro-F1 scores in Table 1 and Table 2.

From Table 1, we can see that our method wvGN achieves the best results compared to the baselines. wvGN achieves an average Micro-F1 score 0.987 and an average Macro-F1 score 0.986 even just given one labeled vertex in each class. In this case, wvGN achieves a gain of 1.86% (min) over node2vec and 161.1% (max) over TSVM. Thus, there is no much space left for wvGN to improve its results when given more labeled vertices. However, for its baselines, we can see that most of them have an ascending performance when given more labeled vertices because they have a relatively worse starting point. Note that the network embedding method node2vec is very competitive. Compared with the Network1, Network2 is more complex. From Table 2, we can see that every method continuously increases their performance with increasing number of labeled vertices. When given five labeled vertices in each class, wvGN achieves a gain of 34.1% (min) over node2vec and 183.2% (max) over LGC in terms of Macro-F1 score. "wvGN (full)" means wvGN with the full use of CD, i.e., all elements in $\mathbf{w}$ are updated by CD. Table 1 reveals that wvGN achieves approximately the same performance as wvGN (full) while Table 2 shows that wvGN is better than wvGN (full) when the number of labeled vertices exceeds two.

Figure 4 and Figure 5 give us the intuitive demonstrations of classification on Network1 and Network2 (the labeled vertices are in black). To reduce the clutter, we only show the results of the top three methods here. Note that wvGN only misclassifies one blue vertex (highlighted in red in Figure 4(b)). This vertex has three edges connected to the green class but only one edge to the blue class. Since the graph is unweighted, the more a vertex has edges connected to a class, the more similar the vertex to the class. According to the weighted vote strategy (see Section 3.1), it makes sense to classify it as the green class. Figure 4(c) depicts the classification result of node2vec. Three blue vertices are misclassified. Figure 4(d) shows that SNBC misclassifies eleven blue vertices. Figure 5(b)–(d) show the classification results of wvGN, node2vec and TSVM on Network2. wvGN misclassifies 25 vertices. node2vec misclassifies 41 vertices. TSVM misclassifies 54 vertices. wvGN achieves the best result and has a gain of 20.9% over the second best method node2vec and 45.8% over the third best method TSVM in terms of Micro-F1 score.

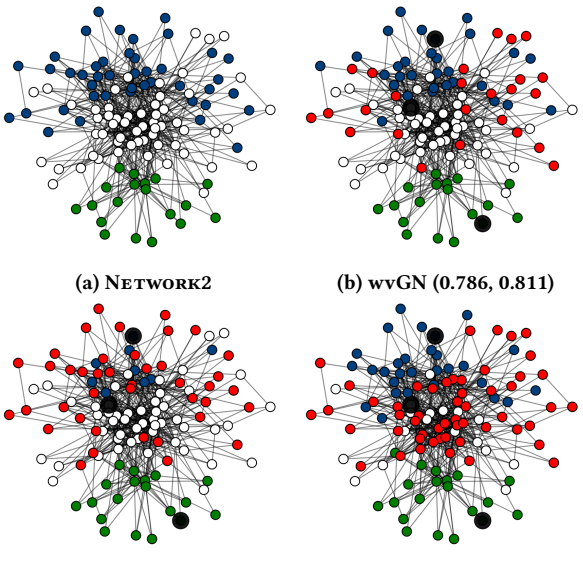
Figure 6 (a) and (b) compare our optimization method GCD with GD, CD and SGD on Network1 and Network2. We can see that all GD, CD and SGD get trapped in local optimal, which leads to poor performance, especially on Network1. When combining GD and CD, our method GCD improves the performance.



(a) Network1 **(b) wvGN (0.990, 0.989)**

(c) node2vec (0.969, 0.967) **(d) SNBC (0.888, 0.871)**

Figure 4: Classification results on Network1. The two labeled vertices are in black and the misclassified vertices are in red. Subcaption: Method (Micro-F1, Macro-F1).



(a) Network2 **(b) wvGN (0.786, 0.811)**

(c) node2vec (0.650, 0.622) **(d) TSVM (0.539, 0.537)**

Figure 5: Classification results on Network2. The three labeled vertices are in black and the misclassified vertices are in red. Subcaption: Method (Micro-F1, Macro-F1).

4.2 Real-world Data

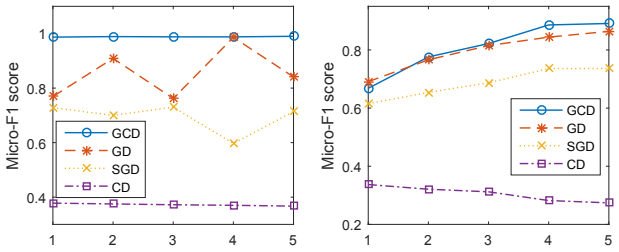
For the real-world data, we use four popular relational datasets CoRA, PubMed, IMDb from [21] and Wikipedia from [10].

Table 1: Classification results on NETWORK1 with the varying number of labeled vertices (#LV) in each class.

	Micro-F1 (%)					Macro-F1 (%)				
#LV	1	2	3	4	5	1	2	3	4	5
wvGN	**98.7±0.7**	**98.9±0.3**	**99.3±0.5**	**99.0±0.3**	**99.2±0.5**	**98.6±0.7**	**98.8±0.4**	**99.2±0.6**	**98.9±0.3**	**99.2±0.5**
wvGN (full)	**98.7±0.5**	98.8±0.7	**99.3±0.5**	**99.0±0.6**	**99.2±0.5**	**98.6±0.5**	98.7±0.7	**99.2±0.6**	**98.9±0.7**	**99.2±0.6**
node2vec	96.9±0.5	94.5±6.0	94.8±5.1	94.5±5.6	94.2±5.4	96.7±0.5	94.3±5.9	94.6±5.1	94.2±5.5	94.0±5.4
Deepwalk	74.6±26.7	75.6±20.7	77.8±18.9	75.9±20.1	79.6±12.0	74.3±26.9	74.5±22.4	77.0±19.9	75.0±21.2	78.7±12.4
SNBC	85.6±15.6	81.2±17.4	71.4±10.7	71.3±10.5	74.9±10.3	84.0±16.6	80.9±17.8	70.9±10.8	70.5±10.6	73.8±10.0
wvRN	77.9±19.2	66.6±18.5	76.3±21.1	86.7±17.7	92.6±7.3	71.5±25.2	55.1±24.0	70.8±25.7	84.3±20.4	91.7±8.2
SCRN	77.3±22.4	74.2±21.8	81.2±17.2	86.3±13.6	90.2±10.6	66.6±32.3	61.0±31.8	73.5±25.6	80.6±21.5	87.0±16.2
SocDim	55.2±9.9	52.0±8.5	53.7±5.8	57.2±6.2	60.0±6.3	51.8±10.2	49.7±7.5	52.4±4.6	54.9±5.8	58.4±5.4
HeatKernel	82.4±19.8	68.8±20.0	78.6±21.2	88.2±16.3	90.9±11.9	77.7±25.9	57.9±26.1	73.8±25.8	86.6±18.0	90.2±12.3
LGC	60.4±18.6	55.6±18.3	60.5±21.9	62.1±22.1	62.4±17.0	45.1±22.4	39.5±19.7	47.5±26.4	46.9±27.0	45.1±20.1
TSVM	37.8±0	37.5±0	37.2±0	37.0±0	36.7±0	27.4±0	27.3±0	27.1±0	27.0±0	26.8±0
LapSVM	51.9±9.4	52.9±8.8	56.0±8.5	58.2±14.4	65.1±12.8	41.6±8.9	45.6±9.9	49.7±8.2	52.9±16.7	62.4±12.4

Table 2: Classification results on NETWORK2 with the varying number of labeled vertices (#LV) in each class.

	Micro-F1 (%)					Macro-F1 (%)				
#LV	1	2	3	4	5	1	2	3	4	5
wvGN	66.7±11.4	71.8±11.0	**77.3±14.5**	**89.0±2.7**	**88.9±2.5**	67.7±12.2	73.3±11.6	**79.1±15.0**	**90.9±2.5**	**90.9±2.1**
wvGN (full)	**67.4±10.1**	**79.9±7.6**	69.6±15.8	86.3±3.5	84.1±5.8	**68.1±10.4**	**82.3±7.1**	71.4±15.1	87.7±4.2	85.3±6.0
node2vec	60.4±9.7	70.5±7.3	61.8±11.5	67.9±10.9	67.9±10.9	58.9±12.3	71.8±8.2	61.2±12.6	67.8±12.3	67.8±12.3
Deepwalk	42.4±7.4	46.5±7.1	42.9±12.6	51.2±5.9	51.2±5.9	40.7±7.2	45.2±6.8	41.7±11.7	48.5±5.8	48.5±5.8
SNBC	35.8±6.7	45.5±12.8	48.7±12.3	56.0±5.0	57.2±4.8	35.3±12.0	43.8±13.1	46.0±13.9	56.7±5.8	57.9±4.9
wvRN	45.0±12.1	56.0±15.7	56.9±11.3	68.1±7.5	68.1±7.5	36.2±13.9	49.4±17.8	53.9±14.9	64.8±11.6	64.8±11.6
SCRN	45.9±14.6	50.3±14.8	57.3±10.3	57.2±4.8	65.2±10.3	36.3±16.4	39.7±18.3	50.1±15.2	57.9±4.9	58.7±17.4
SocDim	31.0±7.9	37.0±7.6	42.9±7.2	44.5±4.7	44.5±4.7	29.1±6.7	35.4±6.5	41.3±6.7	42.1±5.2	42.1±5.2
HeatKernel	48.3±10.1	57.3±16.4	55.3±12.6	65.1±8.0	65.1±8.0	40.2±14.5	52.2±17.7	52.5±14.7	59.6±12.0	59.6±12.0
LGC	31.2±5.1	31.3±4.7	31.5±8.0	35.2±4.8	35.2±4.8	29.6±3.5	29.0±3.6	28.4±6.7	32.1±3.8	32.1±3.8
TSVM	54.2±5.0	64.6±7.5	51.7±4.1	52.0±4.9	52.0±4.9	53.5±5.0	64.7±6.7	51.4±4.1	51.7±4.9	51.7±4.9
LapSVM	37.6±13.2	49.4±10.7	58.3±10.5	63.1±8.5	63.1±8.5	23.4±8.6	39.1±11.2	56.9±12.3	66.0±7.9	66.0±7.9



(a) NETWORK1 (b) NETWORK2

Figure 6: Optimization comparison on the synthetic data. The x-axis represents the number of labeled vertices in each class.

CoRA is a collection of research articles in computer science and PUBMED is a collection of research articles in diabetes. Both CoRA and PUBMED are sparse citation networks. Vertices tend to have low degree in such networks. By using global neighborhood information, we can see from Table 3 and Table 4 that wvGN improves the classification results when the percent of labeled vertices exceeds 1%. From Table 3, we can see that wvRN and HeatKernel are two very competitive baselines considering their classification results and simplicity. When the percent of labeled vertices is 1%, wvRN is superior to our method wvGN in terms of Micro-F1 score. When the percent of labeled vertices is increased to 3%, wvGN achieves a gain of 334.5% (max) over Deepwalk and 6.24% (min) over wvRN in terms of Macro-F1 score. Table 4 demonstrates that HeatKernel is better than wvGN when the percent of labeled vertices is 1%. wvGN achieves a gain of 120.6% (max) over wvRN and 2.92% (min) over HeatKernel in terms of Macro-F1 score when the percent of labeled vertices is 3%.

Differing from CoRA and PUBMED, IMDB is produced based on the vertex similarity. Most of the vertices in the network have similar degrees. Compared with CoRA and PUBMED, IMDB is a more difficult network to classify. We can see from Table 5 that wvGN is superior to its competitors in terms of Micro-F1 score when the percent of labeled vertices is greater than 1%. wvGN achieves a gain of 183.7% (max) over node2vec and 6.37% (min) over SocDim in terms of Micro-F1 score when the percent of labeled vertices is 7%. However, it is defeated by SocDim in terms of Micro-F1 score when the percent of labeled vertices is 1%. It is also defeated by HeatKernel and SNBC in terms of Macro-F1 score. To sum up, wvGN achieves four out of ten best results; HeatKernel achieves three; SNBC achieves two; SocDim achieves one.

Wikipedia is a co-occurrence network of words appearing in the first million bytes of the Wikipedia dump. It is a highly noisy network with lots of interclass edges. Our method wvGN is better than its competitors in terms of Micro-F1 score when the percent of labeled vertices exceeds 1%. To be specific, wvGN achieves a gain of 336.5% (max) over wvRN and 6.82% (min) over SNBC when the percent of labeled vertices is 7%. SNBC outperforms wvGN when the percent of labeled vertices is 1%, but the gap is very narrow, only a gain of 0.72%. In terms of Macro-F1 score, node2vec outperforms wvGN when the percent of labeled vertices is greater than 1%. To sum up, wvGN achieves five out of ten best results; node2vec achieves four; SNBC achieves one.

5 RELATED WORK

Semi-supervised Learning. Transductive SVM (TSVM) [12, 27] achieves the aim of max-margin classification while ensuring that the unlabeled instances are put backward from the margin as far as possible. TSVM first uses SVM to label the unlabeled instances and then switches labels to improve the objective function. This process is susceptible to local optima and requires a large number of label switches before converging. Frequent label switches lead to higher training time compared with SVM. Laplacian Regularized SVM (LapSVM) [1] extends SVM by including the intrinsic smoothness penalty term $\mathbf{f}^\intercal \cdot \mathbf{L} \cdot \mathbf{f}$ in SVM's objective function, where $\mathbf{L}$ is the Laplacian matrix. Because LapSVM needs to compute the inverse of a dense Gram matrix, its time complexity is $O(n^3)$ which is impractical for learning on large-scale networks. Local and Global Consistency (LGC) [31] predicts the labels of unlabeled instances following the prior assumption of consistency, i.e., nearby instances tend to have the same labels, and instances on the same structure (cluster or manifold) tend to have the same labels. During each iteration, each vertex not only receives the label information from its neighbors, but also retains its initial label information. The closed form expression for the vertices is $\beta(\mathbf{I} - \alpha\mathbf{S})^{-1}\mathbf{Y}$, where $\mathbf{S} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ and $\mathbf{Y}$ keeps the initial label information. The closed form expression needs to invert the matrix. For learning on large-scale networks, the inversion operation often consumes a lot of time and resources. In addition, if the assumption of consistency is not met, LGC tends to fail. Compared with LGC and other label-propagation based semi-supervised learning methods, our method wvGN makes no such local and global label consistency assupmtions but directly learns geometic neighborhood information from data, which is then used to infer the labels of vertices. Linear Neighborhood Propagation (LNP) [28] assumes that each data point can be linearly reconstructed from its neighborhood and it uses this assumption to construct the graph from data. For the networked data from which we do not need to construct the graph, LNP deteriorates to LGC.

Relational Learning. The Relational Neighbor (RN) [17] classifier is a simple classifier which only uses the class labels of known related instances without doing learning. RN works by making two strong yet often reasonable assumptions: 1) some instances' class labels are known within the same linked structure and 2) instances related to each other are similar and likely belong to the same class (also called homophily [20]). However, RN may not perform well if the labeled instances in the network are isolated. Instead of making a hard labeling during the inference process, the weighted-vote Relational Neighbor classifier (wvRN) [17] extends RN by assigning class labels to instances with some probabilities. Since the number of the labeled instances in networks is small, both RN and wvRN need to propagate the known label information to the related instances by a collective inference procedure. The above two relational classifiers focus on the single-label classification problem. However, in many real relational networks, each entity may belong to multiple classes. SocDim [25] first extracts latent social dimensions via the top eigenvectors of the modularity matrix and then uses them as features for discriminative learning. The extracted social dimensions describe each instance's hidden relations in the network, which is specially useful when the network has multiple diverse relations inside. Since the extracted latent social dimensions by SocDim are dense which is not scalable for large-scale networks, EdgeCluster [26] partitions the edges into disjoint sets such that each set represents one latent social dimension. To achieve this, a variant of k-means is proposed to handle clustering of many edges. Then a linear SVM is adopted to classify those extracted social dimensions. SCRN [29] is a method designed for the multi-label networks. It starts by constructing a social feature space which is an edge-centric representation of social dimensions to capture the vertex's potential affiliations. To describe each vertex's intrinsic correlation to each class, SCRN assigns each vertex a class-propagation probability. Finally, it assigns the label to a vertex considering its neighbors' class labels, the similarity to its neighbors and its class-propagation probability. ghostEdge [8] works by adding ghost edges to a network to enable the flow of information from labeled vertices to unlabeled vertices. It combines the aspects of statistical relational learning and semi-supervised learning in one framework. Within-Network Classification (WNC) [11] proposes structural-aware vertex features to deal with the situation where the theory of homophily does not hold. WNC only considers the patterns within a given radius threshold, which is incapable of capturing long distance relationships in networks.

Random Walk Based Learning in Networks. Deepwalk [22] uses local information obtained from truncated random walks to learn latent representations of vertices in a network. It models a stream of short random walks on networks as natural language sentences, which is reasonable because both the degree distribution of a connected network and the distribution of words in the natural language follow power law distributions. node2vec [10] is a semi-supervised method for scalable feature learning in networks. It learns a mapping of vertices to a low-dimensional feature space, which maximizes the likelihood of preserving network neighborhoods of vertices. To efficiently explore diverse neighborhood, a biased random walk procedure is proposed, which compromises breadth-first sampling (BFS) and depth-first sampling (DFS). SNBC [21] is a novel structural neighborhood-based learning method based on the lazy random walk. The classification of a vertex is decided based on how it is labeled in the respective k-th level neighborhood. The classification results are affected seriously by the form of the regularization on $\mathbf{w}$. Our method wvGN exploits random walks to explore geometirc one- to m-hop neighborhood information of a vertex. And the label of the vertex is determined by the accumulated geometric one- to m-hop neighborhood information. wvGN uses a proposed gradient and coordinate descent

Table 3: Classification results on CoRA (#vertices: 24,519, #edges: 92,207, #classes: 10) with the varying percent of labeled vertices (%LV). N/A means the results are not available because the algorithm is not finished in one week.

	Micro-F1 (%)					Macro-F1 (%)				
%LV	1%	3%	5%	7%	9%	1%	3%	5%	7%	9%
wvGN	62.8±2.8	**71.6±0.9**	**74.2±0.6**	**75.4±0.5**	**75.8±0.4**	**52.8±2.0**	**63.0±1.0**	**66.1±1.0**	**67.7±0.6**	**68.3±0.8**
node2vec	49.7±15.4	50.3±7.3	50.4±7.7	54.2±7.5	54.8±8.0	42.2±10.0	43.3±6.5	46.4±5.9	49.2±5.3	50.7±7.0
Deepwalk	23.2±1.6	16.5±1.9	18.8±1.4	24.1±0.2	26.7±0.9	15.0±0.5	14.5±1.2	16.9±1.0	20.0±0.1	21.6±0.7
SNBC	50.1±2.6	63.0±1.1	66.5±0.9	68.0±0.9	68.2±0.9	27.2±2.4	49.2±2.1	54.4±0.1	57.3±1.5	57.8±1.0
wvRN	**65.2±1.4**	70.0±0.7	72.1±0.5	73.2±0.6	74.3±0.3	52.5±2.2	59.3±1.0	61.9±0.8	63.7±0.6	64.9±0.6
SCRN	64.3±1.9	70.3±0.7	72.6±0.4	73.6±0.4	74.6±0.3	50.8±3.0	59.2±0.9	62.2±0.7	63.9±0.5	65.2±0.5
SocDim	49.3±0.9	55.6±0.5	59.6±0.6	62.5±0.8	63.6±0.4	27.8±1.3	44.8±0.8	49.9±0.9	53.4±1.2	54.8±0.8
HeatKernel	64.3±2.0	69.6±0.7	72.0±0.5	73.1±0.6	74.2±0.2	51.7±3.1	59.1±1.0	62.0±0.9	63.7±0.6	64.9±0.4
LGC	47.4±2.6	48.7±2.5	48.6±1.8	48.7±2.0	48.5±1.7	23.1±0.3	24.5±2.9	23.1±2.7	23.3±1.6	22.3±2.3
TSVM	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
LapSVM	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Table 4: Classification results on PubMed (#vertices: 19,717, #edges: 44,324, #classes: 3) with the varying percent of labeled vertices (%LV).

	Micro-F1 (%)					Macro-F1 (%)				
%LV	1%	3%	5%	7%	9%	1%	3%	5%	7%	9%
wvGN	63.0±5.7	**75.8±1.2**	**78.4±0.8**	**79.0±0.6**	**79.8±0.5**	59.5±6.4	**73.9±1.1**	**76.7±1.0**	**77.3±0.8**	**78.2±0.5**
node2vec	64.0±4.0	63.2±6.1	64.5±5.5	65.6±5.7	61.1±6.2	61.8±4.5	69.5±8.1	61.6±6.8	62.7±6.8	57.4±7.4
Deepwalk	33.2±1.4	35.0±1.0	35.2±0.8	35.5±0.8	36.1±0.9	32.7±1.2	34.5±0.9	34.7±0.9	35.0±0.6	35.7±0.9
SNBC	52.1±7.2	71.3±1.8	72.3±0.6	78.4±0.6	79.5±0.5	44.2±10.0	68.0±1.5	73.7±1.0	76.3±1.0	77.7±0.6
wvRN	35.8±0.8	35.9±0.5	35.9±0.4	35.9±0.3	36.0±0.3	33.2±0.3	33.5±0.3	33.3±0.4	33.4±0.3	33.3±0.2
SCRN	36.4±0.9	36.2±0.3	36.3±0.5	36.2±0.3	36.2±0.3	33.1±0.3	33.6±0.2	33.4±0.4	33.4±0.4	33.3±0.3
SocDim	42.6±1.6	47.6±1.9	51.5±1.2	55.2±2.1	57.3±1.7	37.7±3.2	43.3±4.1	47.7±3.1	53.1±3.3	55.4±2.2
HeatKernel	**67.9±2.4**	73.3±1.1	76.5±0.6	77.7±0.6	78.8±0.4	**65.9±2.4**	71.8±1.0	75.0±0.6	76.4±0.7	77.5±0.4
LGC	62.2±10.0	71.5±6.3	75.6±2.3	76.6±1.6	76.6±1.7	56.8±11.9	68.9±0	72.7±3.1	74.1±1.8	74.1±1.8
TSVM	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
LapSVM	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Table 5: Classification results on IMDb (#vertices: 19,359, #edges: 362,079, #classes: 21) with the varying percent of labeled vertices (%LV).

	Micro-F1 (%)					Macro-F1 (%)				
%LV	1%	3%	5%	7%	9%	1%	3%	5%	7%	9%
wvGN	24.5±7.7	**40.8±1.9**	**41.5±0.9**	**43.4±0.9**	**44.1±0.8**	8.7±1.2	11.6±0.5	12.5±0.5	13.4±0.7	13.6±0.5
node2vec	11.5±2.7	14.7±1.1	14.1±0.5	15.3±1.0	18.0±1.1	8.6±1.9	11.9±0.4	11.7±0.2	12.1±0.3	12.8±0.4
Deepwalk	15.2±1.7	13.3±0.7	15.9±0.2	24.8±1.9	32.8±1.2	11.2±0.6	10.7±0.3	11.1±0.5	11.2±0.5	11.0±0.2
SNBC	20.3±8.3	35.8±2.4	35.3±1.0	34.8±0.7	34.8±0.8	7.6±1.5	12.3±0.7	14.4±0.4	**15.6±0.4**	**16.5±0.3**
wvRN	33.3±5.0	36.0±0.5	36.4±0.4	37.0±0.3	37.4±0.3	10.0±0.7	10.4±0.3	10.3±0.2	10.4±0.2	10.3±0.2
SCRN	33.6±6.2	36.7±0.9	37.1±0.3	37.8±0.4	38.0±0.4	9.3±0.9	9.5±0.4	9.5±0.2	10.0±0.3	10.0±0.2
SocDim	**37.2±1.8**	38.6±1.2	40.3±0.1	40.8±0.4	41.1±0.3	7.6±0.5	8.0±0.5	8.8±0	9.2±0.4	0.095±0.003
HeatKernel	30.8±6.8	35.6±1.2	37.2±0.6	39.3±0.7	41.2±0.8	**11.7±1.4**	**13.9±0.6**	**14.7±0.5**	15.5±0.6	16.4±0.6
LGC	37.1±4.0	39.4±0.1	39.7±0.1	39.9±0.1	39.9±0.1	8.3±0.4	9.0±0.2	9.4±0.2	9.4±0.1	9.5±0.1
TSVM	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
LapSVM	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

(GCD) method to optimize its objective function. GCD is robust to the starting point and does not easily get trapped in local optima.

Graph Diffusion Based Learning. The most related graph diffusion method to our work is the heat kernel diffusion [5]. It is defined as $\mathbf{h} = \exp(-\rho)\left(\sum_{i=0}^{\infty}\frac{\rho^i}{i!}\mathbf{P}^i\right)\mathbf{s} = \exp\left(-\rho(\mathbf{I}-\mathbf{P}^{-1})\right)\mathbf{s}$, where $\mathbf{s}$ stores the initial class label information of the labeled vertices. It diffuses the class label information of the labeled vertices to the whole network through the above formula. In the classification procedure, it first compares the amount of information a vertex receives from different classes. Then it assigns the vertex to the class which diffuses the most information to the vertex. MultiRankWalk [15] is based on the personalized pagerank diffusion and we find it is inferior to the heat kernel diffusion in the experiments.

Table 6: Classification on Wikipedia (#vertices: 4,777, #edges: 184,812, #classes: 40) with the varying percent of labeled vertices (%LV).

	Micro-F1 (%)					Macro-F1 (%)				
%LV	1%	3%	5%	7%	9%	1%	3%	5%	7%	9%
wvGN	41.6±1.0	**45.3±0.6**	**44.9±1.0**	**45.4±1.0**	**45.4±0.7**	**6.5±0.4**	6.8±0.4	6.3±0.4	6.8±0.5	6.6±0.3
node2vec	29.3±5.0	31.2±1.8	31.1±2.4	31.7±2.6	34.3±4.2	6.4±0.4	**7.1±0.2**	**7.6±0.5**	**8.0±0.6**	**8.3±0.5**
deepwalk	17.7±1.5	14.0±1.2	13.5±1.4	13.8±1.8	15.7±1.8	4.2±0.4	4.1±0.2	4.1±0.3	4.0±0.3	4.0±0.2
SNBC	**41.9±0.5**	42.5±0.4	42.4±0.6	42.5±0.6	42.6±0.5	4.4±0.2	4.4±0.3	4.4±0.5	4.5±0.3	4.6±0.4
wvRN	1.6±1.1	4.2±2.5	7.8±4.0	10.4±3.5	13.0±6.6	0.7±0.4	1.1±0.6	1.6±0.5	2.0±0.7	2.2±0.6
SCRN	1.7±1.3	4.2±2.1	8.3±4.1	11.8±3.1	15.0±6.2	0.7±0.4	1.1±0.5	1.7±0.5	2.1±0.5	2.4±0.5
SocDim	33.9±1.5	32.4±1.6	32.6±0.8	33.3±0.9	33.5±1.0	5.6±0.2	6.4±0.3	6.5±0.4	6.8±0.5	7.2±0.4
HeatKernel	1.3±1.0	4.3±3.3	7.7±4.3	10.5±3.8	12.8±8.7	0.6±0.4	1.0±0.7	1.5±0.6	1.7±0.6	2.0±0.7
LGC	36.7±6.5	38.9±0.1	39.0±0.1	39.0±0.1	39.1±0.1	3.0±0.3	3.0±0	3.0±0	3.0±0	2.9±0
TSVM	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
LapSVM	27.4±13.9	41.9±0.6	42.0±0.6	42.2±0.3	42.2±0.6	2.4±0.7	4.1±0.4	4.2±0.3	4.3±0.3	4.2±0.4

6 CONCLUSION

In this work, we have proposed wvGN to tackle the problem of intra-network classification when the number of labeled vertices is limited. wvGN is a semi-supervised learning framework which exploits both labeled and unlabeled vertices to achieve better classification. Conventional graph-based semi-supervised and relational learning methods either make assumptions of local and global label consistency or do not learn from data, and thus they do not perform well if the assumption is not met or the starting inferene procedure has some errors. To conquer these, wvGN learns geometic neighborhood information directly from data. It optimizes an objective function based on L2-loss SVM. A search strategy based on the gradient and coordinate descent methods has been developed to solve the problem of local optima. Empirical studies prove that our method wvGN is superior to state-of-the-art methods. One challenge raised in networks is that networks are highly dynamic. The out-of-sample extension of wvGN from the current static networks to dynamic networks will be dealt with in the future work.

REFERENCES

[1] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. 2006. Manifold Regularization: A Geometric Framework for Learning from Labeled and Unlabeled Examples. *Journal of Machine Learning Research* 7 (2006), 2399–2434.

[2] Christopher M Bishop. 2006. Pattern recognition and Machine Learning. *Springer* (2006).

[3] Christopher J. C. Burges. 1998. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Min. Knowl. Discov.* 2, 2 (1998), 121–167.

[4] Kai-Wei Chang, Cho-Jui Hsieh, and Chih-Jen Lin. 2008. Coordinate descent method for large-scale l2-loss linear support vector machines. *Journal of Machine Learning Research* 9, Jul (2008), 1369–1398.

[5] Fan Chung. 2007. The heat kernel as the pagerank of a graph. *Proceedings of the National Academy of Sciences* 104, 50 (2007), 19735–19740.

[6] Corinna Cortes and Vladimir Vapnik. 1995. Support-Vector Networks. *Machine Learning* 20, 3 (1995), 273–297.

[7] Rong-En Fan and Chih-Jen Lin. 2007. A study on threshold selection for multi-label classification. *Department of Computer Science, National Taiwan University* (2007), 1–23.

[8] Brian Gallagher, Hanghang Tong, Tina Eliassi-Rad, and Christos Faloutsos. 2008. Using ghost edges for classification in sparsely labeled networks. In *SIGKDD*. 256–264.

[9] Lise Getoor. 2007. *Introduction to statistical relational learning*. MIT press.

[10] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. In *SIGKDD*. 855–864.

[11] Jialong Han, Ji-Rong Wen, and Jian Pei. 2014. Within-network classification using radius-constrained neighborhood patterns. In *CIKM*. 1539–1548.

[12] Thorsten Joachims. 1999. Transductive Inference for Text Classification using Support Vector Machines. In *ICML*. 200–209.

[13] Kyle Kloster and David F Gleich. 2014. Heat kernel based community detection. In *SIGKDD*. ACM, 1386–1395.

[14] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. 2008. Benchmark graphs for testing community detection algorithms. *Physical review E* 78, 4 (2008).

[15] Frank Lin and William W Cohen. 2010. Semi-supervised classification of network data using very few labels. In *ASONAM*. 192–199.

[16] Frank Lin and William W Cohen. 2010. A Very Fast Method for Clustering Big Text Datasets.. In *ECAI*. 303–308.

[17] Sofus A Macskassy and Foster Provost. 2003. A Simple Relational Classifier. In *MRDM workshop at KDD*.

[18] Sofus A Macskassy and Foster Provost. 2007. Classification in networked data: A toolkit and a univariate case study. *Journal of Machine Learning Research* 8, May (2007), 935–983.

[19] Olvi L Mangasarian. 2002. A finite Newton method for classification. *Optimization Methods and Software* 17, 5 (2002), 913–929.

[20] Miller McPherson, Lynn Smith-Lovin, and James M Cook. 2001. Birds of a feather: Homophily in social networks. *Annual review of sociology* (2001), 415–444.

[21] Sharad Nandanwar and M. Narasimha Murty. 2016. Structural Neighborhood Based Classification of Nodes in a Network. In *SIGKDD*. 1085–1094.

[22] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *SIGKDD*. ACM, 701–710.

[23] John Platt and others. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers* 10, 3 (1999), 61–74.

[24] Shai Shalev-Shwartz, Yoram Singer, and Nathan Srebro. 2007. Pegasos: Primal estimated sub-gradient solver for svm. In *ICML*. ACM, 807–814.

[25] Lei Tang and Huan Liu. 2009. Relational learning via latent social dimensions. In *SIGKDD*. ACM, 817–826.

[26] Lei Tang and Huan Liu. 2009. Scalable learning of collective behavior based on sparse social dimensions. In *CIKM*. ACM, 1107–1116.

[27] Vladimir Naumovich Vapnik and Vlamimir Vapnik. 1998. *Statistical learning theory*. Vol. 1. Wiley New York.

[28] Fei Wang and Changshui Zhang. 2006. Label propagation through linear neighborhoods. In *ICML*. 985–992.

[29] Xi Wang and Gita Sukthankar. 2013. Multi-label relational neighbor classification using social context features. In *SIGKDD*. ACM, 464–472.

[30] Wayne W Zachary. 1977. An information flow model for conflict and fission in small groups. *Journal of anthropological research* 33, 4 (1977), 452–473.

[31] Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. 2003. Learning with Local and Global Consistency. In *NIPS*. 321–328.