Neural Comput & Applic (2010) 19:549–555
DOI 10.1007/s00521-009-0305-8

ORIGINAL ARTICLE

A general graph-based semi-supervised learning with novel class discovery

Feiping Nie · Shiming Xiang · Yun Liu · Changshui Zhang

Received: 18 December 2008 / Accepted: 24 August 2009 / Published online: 11 September 2009

Abstract In this paper, we propose a general graph-based semi-supervised learning algorithm. The core idea of our algorithm is to not only achieve the goal of semi-supervised learning, but also to discover the latent novel class in the data, which may be unlabeled by the user. Based on the normalized weights evaluated on data graph, our algorithm is able to output the probabilities of data points belonging to the labeled classes or the novel class. We also give the theoretical interpretations for the algorithm from three viewpoints on graph, i.e., regularization framework, label propagation, and Markov random walks. Experiments on toy examples and several benchmark datasets illustrate the effectiveness of our algorithm.

Keywords Pattern recognition · Semi-supervised learning · Novel class discovery · Normalized weights

1 Introduction

In many real-world applications in data mining, information retrieval and pattern recognition, labeled data are usually very insufficient and labeling a huge number of data points needs expensive human labor and takes much time. However, unlabeled data may be abundant and can be easily and cheaply obtained. Thus how to use the labeled and unlabeled data to improve the performance becomes an important problem. This motivates a hot research direction of semi-supervised learning [1].

Most semi-supervised learning algorithms [2–5] are constructed under some clustering and manifold assumptions [6, 7]. These assumptions are sensible since in many real-world problems the neighboring data points or the data points forming the same structure (manifold) are likely to have the same label. A typical family of algorithms are those developed on data graph [8–11].

Based on data graph, Zhu et al. [11] proposed an algorithm called Harmonic Energy Minimization (HEM). In HEM, Gaussian fields and harmonic functions are used to propagate the label information to the unlabeled data. The algorithm HEM can be interpreted as a random walks on graph, and can yield output of probability value, i.e., the output can be viewed as the probabilities of the data points belonging to the labeled classes. However, the algorithm clamps the labels for the labeled data, which makes it sensitive to noises in the labeled data. Later, Belkin et al. [8] proposed an algorithm to relax the constraints on the labeled data, which makes it insensitive to noises in the labeled data. However, the interpretation of random walks for it is not clear, and the algorithm may fail to classify the data if the density of the data varies largely across different classes. In addition, the derived matrix may be singular when the constructed graph is not connected, which makes the algorithm unsolvable.

Recently, Zhou et al. [10] proposed an algorithm called Learning with Local and Global Consistency (LLGC). The algorithm uses normalized Laplacian [12] to construct the

F. Nie (✉) · C. Zhang
State Key Laboratory of Intelligent Technology and Systems, Tsinghua National Laboratory for Information Science and Technology, Department of Automation, Tsinghua University, Beijing 100084, China
e-mail: feipingnie@gmail.com

S. Xiang
National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China

Y. Liu
School of Electrical & Electronic Engineering, Nanyang Technological University, Singapore, Singapore

regularizer, in which some drawbacks have been avoided. This algorithm can also be explained in view of random walks on graph. However, under this interpretation, the output of the algorithm are not probabilities but normalized commute times [13]. Thus, there lacks of a mechanism to calculate the probabilities of the data points belonging to the classes, which may be very useful for further data processing.

In this paper, we propose a general graph based algorithm with normalized weights for semi-supervised learning. In our algorithm, the drawbacks mentioned above are eliminated. We use the Laplacian with normalized weights to construct the regularizer, and a novel class label is introduced into the algorithm to discover novel class. Several theoretical interpretations on graph are given, which make our algorithm sound for semi-supervised learning tasks.

The rest of this paper is organized as follows: we propose the algorithm in Sect. 2. In Sect. 3, we give the theoretical interpretations for the proposed algorithm from three viewpoints on graph, i.e., regularization framework, label propagation and random walks. Some discussions are given in Sect. 4. In Sect. 5, the experimental results on several toy examples and benchmark datasets are reported to demonstrate the effectiveness of our algorithm. Finally, we give the conclusions in Sect. 6.

2 The algorithm

Given a point set $\mathcal{X} = \{x_1, \ldots, x_l, x_{l+1}, \ldots, x_n\}$ and a labeled class set $\mathcal{C} = \{1, \ldots, c\}$, the first l points $x_i (i \leq l)$ are labeled as $y_i \in \mathcal{C}$ and the remaining u points x_{l+1}, ..., x_{l+u} are unlabeled. Here $n = l + u$, and usually $l \ll u$. We introduce an additional label to construct the label set as $\tilde{\mathcal{C}} = \{1, \ldots, c, c+1\}$. The label $c + 1$ gives the algorithm a mechanism to discover novel class.

The goal of the algorithm is to predict the labels of the unlabeled points using both the labeled data and the unlabeled data.

Let $F = [F_1^T, \ldots, F_n^T]^T \in \mathbb{R}^{n \times (c+1)}$ be the the soft label matrix, where $F_i \in \mathbb{R}^{c+1} (1 \leq i \leq n)$ are row vectors and each element in F_i belongs to [0,1]. Define the matrix $Y = [Y_1^T, \ldots, Y_n^T]^T \in \mathbb{R}^{n \times (c+1)}$, where $Y_i \in \mathbb{R}^{c+1} (1 \leq i \leq n)$ are row vectors. For the labeled data, $Y_{ij} = 1$ if x_i is labeled as j and $Y_{ij} = 0$ otherwise. For the unlabeled data x_i, $Y_{ij} = 1$ if $j = c + 1$ and $Y_{ij} = 0$ otherwise. Our algorithm is described as follows:

1. Construct the neighborhood weighted graph. Points x_i and x_j are linked by a weight calculated by

   $$W_{ij} = e^{-\|x_i - x_j\|^2/\sigma^2} \tag{1}$$

   if x_i is in the k-neighbors of x_j or x_j is in the k-neighbors of x_i, otherwise, $W_{ij} = 0$. Here $\|\cdot\|$ is the 2-norm of vector, i.e., $\|x\|^2 = x^T x$.
2. Calculate the normalized weights by

   $$\tilde{W}_{ij} = W_{ij} / (\sqrt{d_i d_j}) \tag{2}$$

   and the normalized weight matrix can be written as $\tilde{W} = D^{-1/2} W D^{-1/2}$, where D is a diagonal matrix with entries $d_i = \sum_j W_{ij}$.
3. Calculate $P = \tilde{D}^{-1} \tilde{W}$, where $\tilde{D}$ is a diagonal matrix with entries $\tilde{d}_i = \sum_j \tilde{W}_{ij}$.
4. Calculate the soft label matrix $F \in \mathbb{R}^{n \times (c+1)}$ by

   $$F = (I - I_\alpha P)^{-1} I_\beta Y \tag{3}$$

   where I is an $n \times n$ identity matrix, I_α is an $n \times n$ diagonal matrix with the ith entry being α_i, and $I_\beta = I - I_\alpha$. Here $\alpha_i (0 \leq \alpha_i < 1)$ is a parameter for data x_i, which will be discussed later. Then the label of data point x_i is assigned as

   $$y_i = argmax_{j \leq c+1} F_{ij} \tag{4}$$

If $y_i = c + 1$, then x_i can be seen as a sample coming from a novel class. This mechanism of novel class discovery is useful since the unlabeled data may not belong to all of the labeled classes. On other hand, if the prior knowledge tells us that the number of classes is just c, then if $y_i = c + 1$, x_i can be seen as an outlier, or be assigned as

$$y_i = argmax_{j \leq c} F_{ij} \tag{5}$$

3 Interpretations on graph for the algorithm

In this section, we give some theoretical interpretations from the viewpoint of graph for the algorithm proposed in Sect. 2. We will show that the algorithm can be derived from a regularization framework, and also can be seen as a label propagation process and a special Markov random walks.

Consider a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with nodes $\mathcal{V}$ corresponding to the n data points, nodes $\mathcal{L} = \{1, \ldots, l\}$ corresponding to the labeled points with labels y_1, ..., y_l, and nodes $\mathcal{U} = \{l+1, \ldots, l+u\}$ corresponding to the unlabeled points. The normalized weight $\tilde{W}$ used below is constructed on the edges of graph.

3.1 Regularization framework

Denote $tr(\cdot)$ as the trace operator, and denote $\|\cdot\|_F$ is the Frobenius norm of matrix, i.e., $\|M\|_F^2 = tr(M^T M)$. Consider a regularization framework on graph, the cost function associated with F is defined as

$$\mathcal{J}(F) = \sum_{i,j=1}^{n} \tilde{W}_{ij} \|F_i - F_j\|_F^2 + \sum_{i=1}^{n} \mu_i \tilde{d}_i \|F_i - Y_i\|_F^2 \tag{6}$$

where $\tilde{W}_{ij}, F_i, Y_i$, and $\tilde{d}_i$ are defined as the same as those in Sect. 2.

The first term in the cost function is a regularization term, which measures the smoothness of the resulted labels on graph. The second term is a fitting term, which measures the difference between the resulted labels and the initial labels assignment. The trade off between these two competing constraints is controlled by μ_i and $\tilde{d}_i$. Here $\mu_i > 0$ is a regularization parameter for the ith data point x_i and $\tilde{d}_i = \sum_j \tilde{W}_{ij}$ is the degree of the ith data point x_i.

For the purpose of analyzing conveniently, we rewritten (6) in the matrix form as

$$\mathcal{J}(F) = tr(F^T \tilde{L} F) + tr((F - Y)^T U \tilde{D} (F - Y)) \tag{7}$$

where $\tilde{L} = \tilde{D} - \tilde{W}$ is a Laplacian matrix with normalized weights $\tilde{W}$, and U is a diagonal matrix with the ith entry being μ_i.

The optimal solution for the optimization problem can be easily solved by setting the derivative of $\mathcal{J}(F)$ to zero, i.e.,

$$\left.\frac{\partial \mathcal{J}}{\partial F}\right|_{F=F^*} = 2\tilde{L}F^* + 2U\tilde{D}(F^* - Y) = 0 \tag{8}$$

Let us introduce a set of variables

$$\alpha_i = 1/(1 + \mu_i) \quad (i = 1, 2, \ldots, n) \tag{9}$$

note that $P = \tilde{D}^{-1}\tilde{W}$, then the solution can be derived as

$$\begin{aligned} F^* &= (L + U\tilde{D})^{-1} U\tilde{D} Y \\ &= (I - P + U)^{-1} UY \\ &= (I_\alpha - I_\alpha P + I_\beta)^{-1} I_\beta Y \\ &= (I - I_\alpha P)^{-1} I_\beta Y \end{aligned} \tag{10}$$

which is just the classifying function (3) used in the proposed algorithm.

3.2 Label propagation

Let us consider an iterative process for label propagation. In each iteration, the label information of each data point is partly received from its neighbors, and the rest is received from its initial label (see Fig. 1a). The label information of the data at time $t + 1$ is propagated based on the following equations

$$F(t + 1) = \hat{P} F(t) + I_\beta Y \tag{11}$$

where $\hat{P} = I_\alpha P$, and I_α, I_β and P are defined as those used in Sect. 2.

We now show that the sequence $F(t)$ will converge to the same solution as in (3). By the iteration (11), we have

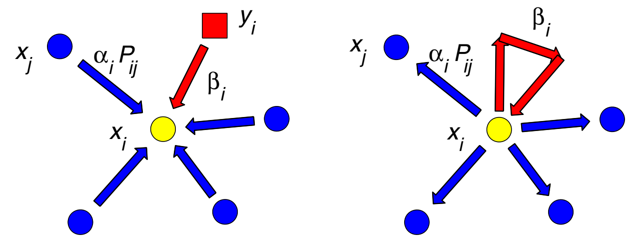
$$F(t) = \hat{P}^t F(0) + \sum_{i=0}^{t-1} \hat{P}^i I_\beta Y \tag{12}$$

Note that the ∞-norm of matrix $\hat{P}$ is lower than 1 in the case of $0 \le \alpha_i < 1 (1 \le i \le n)$. According to the matrix property, the spectral radius of $\hat{P}$ is not greater than the ∞-norm, i.e., $\rho(\hat{P}) < 1$. Therefore, $I - \hat{P}$ is invertible and we have $\lim_{t\to\infty} \hat{P}^t = 0$ and $\lim_{t\to\infty} \sum_{i=0}^{t-1} \hat{P}^i I_\beta Y = (I - \hat{P})^{-1} I_\beta Y$. Hence the iteration process is convergent and converges to

$$F^* = \lim_{t\to\infty} F(t) = (I - \hat{P})^{-1} I_\beta Y = (I - I_\alpha P)^{-1} I_\beta Y \tag{13}$$

and does not depend on the initial value $F(0)$.

Therefore, the proposed algorithm in Sect. 2 can be interpreted from an iterative process of label propagation on graph, with the transition matrix being $\hat{P}$.



(a) Label propagation **(b)** Random walks

Fig. 1 The label propagation and random walks on graph. (**a**) In each iteration of the label propagation process, the label information of each data is partly received from its neighbors' labels, and the rest is received from its initial label y_i. (**b**) Each data point x_i randomly walks to its neighbors with the probability determined by P. There is a probability β_i to return to itself at one walk. The walks will stop when hits one of the data points on graph twice consecutively

3.3 A special random walks

Imagining a random walks on graph (see Fig. 1b), and the transition probability matrix $\tilde{P}$ is

$$\tilde{P} = I_\beta + I_\alpha P \tag{14}$$

where I_α, I_β and P are defined as the same in Sect. 2. Note that each row of $\tilde{P}$ sum to 1, which indicates $\tilde{P}$ is a stochastic matrix. The stop rule of the special random walks are defined as following:

Stop rule: Each point walks randomly on the graph based on the transition probability matrix $\tilde{P}$, and **stops** when it consecutively hits one of the points on the graph **twice**. It is considered to have hit the starting point once before the walks.

Denote G as

$$G = I_\beta + \hat{P} I_\beta + \hat{P}^2 I_\beta + \cdots + \hat{P}^n I_\beta + \cdots \tag{15}$$

Note that the value of $(\hat{P}^k I_\beta)_{ij}$ is the probability of the ith point stopping the walks at the jth point at the kth step, so G_{ij} is the probability of the ith point stopping the walks at the jth point.

G can be written as $G = (I - I_\alpha P)^{-1} I_\beta$, therefore (3) can be written as

$$F = GY \tag{16}$$

From (15) and (16) we see that $F_{ij}(j \leq c)$ is just the probability of the ith point which stops the random walks at the labeled data point whose label is j, and $F_{ij}(j = c + 1)$ is the probability of the ith point which stops the random walks at one of the unlabeled data point.

Therefore, the proposed algorithm in Sect. 2 can also be interpreted as a special random walks on graph, with the transition probability matrix being $\tilde{P}$ defined in (14) and with the stop condition being twice hitting one of the data point consecutively.

Several properties of the proposed algorithm can be clearly understood from the viewpoint of this random walks. The stop condition of **twice** hitting one of the data makes the starting data point having the chance to stop the walks at another data point, which means the resulted label of the labeled data can be changed from its initial label.

The method proposed by Zhu et al. [11] can also be interpreted as random walks, but the transition probability matrix and the stop condition are different from ours. In their method, the walks can only stop at the labeled data points, while in our algorithm, the random walks can stop at the unlabeled data points, which makes our algorithm having the mechanism to discover novel class in data.

4 Discussions

It is interesting to note that the label propagation procedures and the random walks defined in Sect. 3 seem to be very similar, these two procedures, however, are essentially different. First, the transition matrices are different ($\hat{P}$ in the label propagation procedures while $\tilde{P}$ in the random walks). Second, the transition directions in these two procedures are inverted (see Fig. 1).

The proposed algorithm is an extension to HEM [11]. The introduced parameters α_i for each data x_i make the algorithm more general. HEM is a special case in this algorithm where $\alpha_i = 0$ for the labeled data x_i and $\alpha_i = 1$ for the unlabeled data x_i. In contrast, we can set the parameters α_i with more freedom in the general algorithm. Usually, for the labeled point x_i, if we are sure that the initial label is definitely correct, α_i can be set to zero, which means the resulted label of x_i will be equal to the initial label and remains unchanged, otherwise α_i may be set to a positive value such that the resulted label of x_i can be changed from the initial label, which is important to detect noises in the labeled data. For the unlabeled point x_i, α_i can be set to a large value but lower than 1, $\alpha_i = 1$ means that the resulted label of x_i will definitely be 1 to c, and thus lose the capability to discover the novel class. Moreover, $\alpha_i = 1$ may make the matrix $(I - I_\alpha P)$ singular. Therefore, we constrain $\alpha_i < 1$ in our algorithm.

The algorithm LLGC [10] is also derived from a regularized framework, but the two terms in the regularized framework are both different from those of us. The output of LLGC are not probability values, while in our algorithm, denote $\mathbf{1}_n = [1, \ldots, 1]^T \in \mathbb{R}^{n \times 1}$, we have

$$\left.\begin{array}{r} P\mathbf{1}_n = \mathbf{1}_n \\ Y\mathbf{1}_{c+1} = \mathbf{1}_n \end{array}\right\} \Rightarrow I_\alpha P\mathbf{1}_n + I_\beta Y\mathbf{1}_{c+1} = \mathbf{1}_n$$
$$\Rightarrow I_\beta Y\mathbf{1}_{c+1} = (I - I_\alpha P)\mathbf{1}_n \tag{17}$$
$$\Rightarrow (I - I_\alpha P)^{-1} I_\beta Y\mathbf{1}_{c+1} = \mathbf{1}_n$$

which indicates that the output are probability values, and thus might be more convenient for further data processing. It is worth to note that if we remove $\tilde{d}_i$ from the second term of the regularization framework in (6), the results are not probability values anymore, and thus cannot be interpreted by the subsequent label propagation and random walks.

The effect of the normalized weights is illuminated in Fig. 2. Recall that the normalized weight between x_i and x_j is defined as $\tilde{W}_{ij} = W_{ij}/(\sqrt{d_i d_j})$.

The normalization can strengthen the weights in the low density region and weaken the weights in the high density region, which make the overall weights normalized. Therefore, the normalized weights might make the classification more easier in the case that the density of the data varies largely across different classes. It is worth to note that the normalized Laplacian matrix $I - D^{-1/2}WD^{-1/2} = I - \tilde{W}$ also exploits the effectiveness of the normalized weight $\tilde{W}$. However, when the Laplacian matrix $\tilde{L}$ in (7) is replaced by the normalized Laplacian matrix as in LLGC [10], the results are not probability values anymore as the normalized Laplacian matrix is usually not a Laplacian matrix.

5 Experiments

In this section, we first validate our algorithm with some toy examples, and then evaluate it on several benchmark

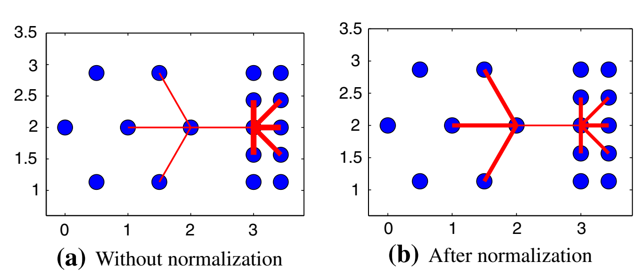


Fig. 2 The relative changes of weights before and after normalization. *Thicker line* denotes larger weight. These changes clearly indicate that the weights with normalization make the classification between the data with very different density region more easier

datasets. Finally, we give an experiment to verify the capability of our algorithm to discover novel class in data.

In our experiments, as there is no prior knowledge to be used, we simply set the regularization parameters α_i in (9) for the labeled data x_i to the same value α_l, and the regularization parameters α_i for the unlabeled data x_i to the same value α_u.

5.1 Toy examples

We give several toy examples to analyze and validate our algorithm. The effect of the normalized weights, the α_l, and the α_u are discussed in these toy problems.

Figure 3a shows the toy data which consists of two classes with very different density distributions. The results illustrate that by the normalized weights, our method can effectively classify the data in the case that the density of the data varies largely across different classes.

Figure 4a shows the toy data of two rings with 8 labeled points. From the cluster assumption and the manifold assumption, ideal classifier should classify the points on the outside ring as one class and the points on the inside ring as another class. Thus there is one incorrectly labeled point in each class, which can be viewed as noise. The situation described in Fig. 4a may very possibly exist in real world problems since the noise in labeling is easily to occur possibly due to the tiredness or careless of the annotator. Therefore, developing the robust classifier which can automatically detect the noise is of vital importance.

To some extent, our method can automatically detect the noise in the labeled data if we set α_l to a positive value. However, if we set α_l to zero, the noise in the labeled data can not be detected, since $\alpha_l = 0$ means the resulted label will not change from its initial label. Therefore, if the label information for the labeled data x_i is not very convincible, we can set the corresponding α_i to a larger value. On the contrary, if we can ensure that the label information for the labeled data x_i is correct, the corresponding α_i could set to zero.

Figure 5a shows the toy data of three rings with only two labeled points. From the cluster and manifold assumptions, ideal classifier should classify the three rings as three classes. However, there are only two classes being labeled, it is desirable to discover the intermediate ring as novel class. The results clearly demonstrate that our algorithm has the capability to discover novel class with $\alpha_u < 1$.

5.2 Experiments on benchmark datasets

We evaluate our algorithm with the benchmark datasets provided in [6], and compare with k nearest neighbor classifier (kNN), SVM, and several popular semi-supervised learning algorithms, including Transductive SVM (TSVM) [4], Low Density Separation (LDS) [14], Cluster Kernels (CK) [15], Laplacian Regularized Least Squares (LRLS) [2] and Learning with Local and Global Consistency (LLGC) [10]. We denote our algorithm without normalized weights as GGSSL_1 and that with normalized weights as GGSSL_2.

The benchmark consists of seven datasets. A brief description for the datasets are summarized in Table 1. The first two datasets were generated from two Gaussians without the manifold structure. For the image datasets of Digit1, USPS and COIL, the manifold assumption is expected to be held. For each dataset, 12 splits are

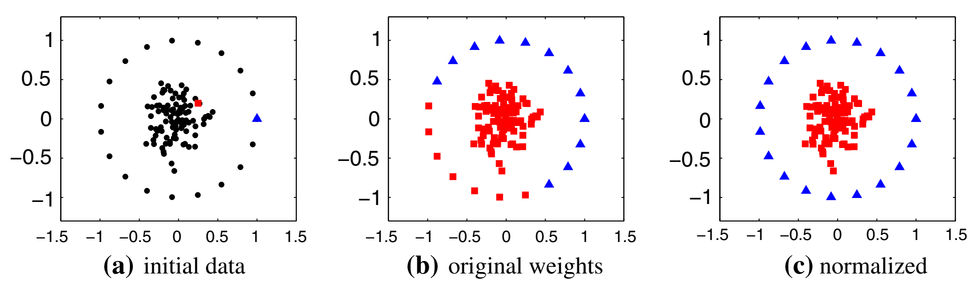


Fig. 3 **a** The partially labeled data. **b** Classification result without normalized weights. **c** Classification result with normalized weights. The results demonstrate that by the normalized weights, it can successfully classify the data with very different density between classes

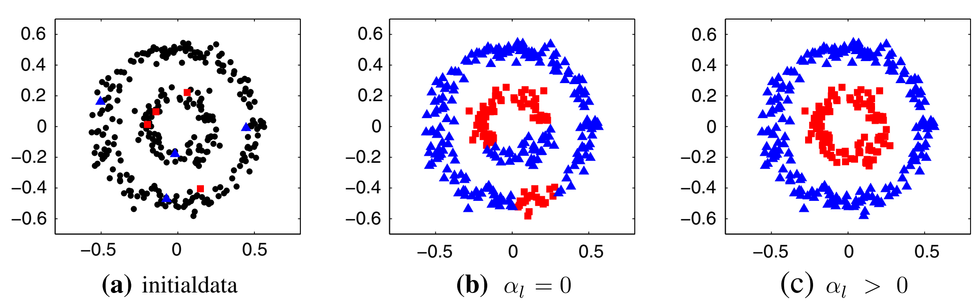
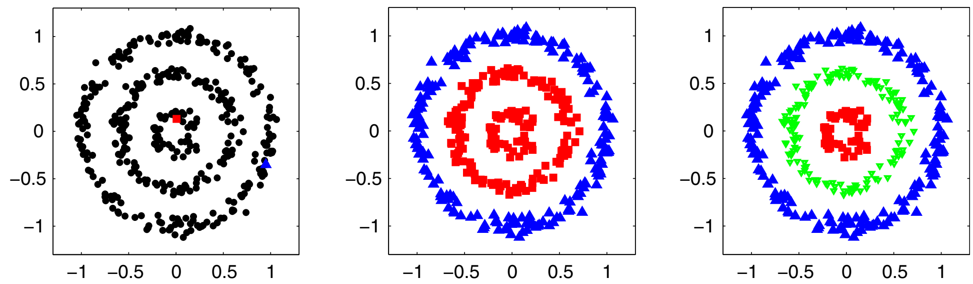


Fig. 4 **a** The partially labeled data with noise. **b** Classification result with $\alpha_l = 0$. **c** Classification result with $\alpha_l = 0.99$. The results demonstrate that when α_l is set to a positive value, our method can effectively detect the noises in labeled data



(a) initial data (b) $\alpha_u = 1$ (c) $\alpha_u < 1$

Fig. 5 **a** The partially labeled data. **b** Classification result with $\alpha_u = 1$. **c** Classification result with $\alpha_u = 0.99999$; The *green, down triangles* denote the novel class data discovered by our algorithm. The results demonstrate that when α_u is set to a value less than 1, our method can effectively discover novel class in data

Table 1 Description of the benchmark datasets

Dataset	Classes	Dimension	Points	Comment
g241c	2	241	1,500	artificial
g241d	2	241	1,500	artificial
Digit1	2	241	1,500	artificial
USPS	2	241	1,500	imbalanced
COIL	6	241	1,500	–
BCI	2	117	400	–
Text	2	11,960	1,500	sparse discrete

Table 2 Average test errors (%) with 100 labeled training data points

Method	g241c	g241d	Digit1	USPS	COIL	BCI	Text
1-NN	40.28	37.49	6.12	7.64	23.27	44.83	30.77
SVM	23.11	24.64	5.53	9.75	22.93	34.31	26.45
TSVM	18.46	22.42	6.15	9.77	25.80	33.25	24.52
LDS	18.04	23.74	3.46	4.96	13.72	43.97	**23.15**
CK	**13.49**	**4.95**	3.79	9.68	21.99	35.17	24.38
LRLS	24.36	26.46	2.92	4.68	11.92	**31.36**	23.57
LLGC	48.57	46.74	2.55	**3.98**	8.98	48.44	24.58
$GGSSL_1$	45.96	42.74	2.39	6.38	9.48	45.44	23.51
$GGSSL_2$	44.19	40.95	**2.29**	5.29	**8.79**	45.61	27.01

provided. Each split contains 100 labeled data and at least one labeled point for each class, and there is no bias in the labeling process. In these experiments, for the kNN classifier, we use the nearest neighbor classifier (1-NN). For the LLGC and our algorithm, the parameter k in the construction of k-neighborhood graph is simply set to 6, and the parameter σ in (1) is determined by $\sigma = \sqrt{-\frac{\bar{d}}{ln(s)}}$, where $\bar{d}$ is the average of squared Euclidean distances for all the edged pairs, i.e., $\bar{d} = \frac{1}{z}\sum_{ij,W_{ij}\neq 0}\left\|x_i - x_j\right\|^2$($z$ is the number of all the edged pairs). s is searched from: $s \in \{0.0001*1/k, 0.001*1/k, 0.01*1/k, 0.1*1/k, 1/k\}$.

In our algorithm, the regularization parameter α_l is simply set to 0 and α_u is simply set to 0.99999.

The results are summarized in Table 2, in which the results of SVM, TSVM, LDS, CK and LRLS have been reported in [6]. The experimental results demonstrate that there is no algorithm uniformly better than the others. Therefore, how to select an algorithm for a specific dataset is an open problem.

The performance of our algorithm for these benchmark datasets is comparable to LLGC method, and behaves better on Digit1 and COIL, which indicates that our algorithm is expected to perform well for the data having manifold structure.

It is worthy noting that the mainly computation time of our algorithm is spent on the first step, i.e., constructing of W, which is a necessary step for graph based method. Equation 3 in our algorithm is actually solved by a large sparse linear system, which has been intensively studied and there exist efficient algorithms whose computational time is nearly linear [16].

5.3 Novel class discovery

We present an experiment to validate the capability of our algorithm to discover novel class in data. Since, the COIL dataset consists of six classes in the benchmark, it is selected in this experiment. We only use the labeled information from the first three classes, and remove the labeled information from the last three classes. Therefore, the last three classes can be seen as a novel class in this setting.

We use the kNN classifier as the baseline, and compare our algorithm with LLGC. The parameters in the algorithms are set as those in previous experiments. Essentially, LLGC has not the mechanism to discover novel class. For each data x_i, LLGC outputs three values corresponding the first three classes, and x_i is classified to the class whose corresponding value is maximum. Here, we make a slight modification for it. If the maximum value is lower than a threshold t, then x_i is classified to the novel class.

We record the test error rate for the data from the first three classes, the test error rate for the data from the last

Table 3 Test errors (%) for the COIL dataset. Data from the first three classes are seen as "data with labeled class", and data from the last three classes are seen as "data with novel class"

Method	Data with labeled class	Data with novel class	Overall
1-NN	19.29	100.00	59.36
LLGC	14.18	4.17	9.21
$GGSSL_1$	17.16	0.00	8.64
$GGSSL_2$	13.05	0.00	6.57

three classes, and the overall test error rate for the all data, respectively.

The results are presented in Table 3. For LLGC, the threshold t is set to the value that the overall test error rate is minimum, and $t = 0.00008$ in this experiment. It is worth to point out that the way of setting t is favorable to LLGC and practically infeasible since the test error rate is usually unavailable in practice. Our algorithm has the intrinsic mechanism to discover novel class, and the results in this experiment demonstrate that the mechanism is effective in practice.

6 Conclusions

In this paper, we propose a general algorithm for semi-supervised learning based on graph. The algorithm is formulated as an optimization problem which can be effectively and efficiently solved. Several drawbacks in traditional graph based method have been eliminated in our algorithm. Moreover, our algorithm has the mechanism to discover novel class in data, which is useful in the practice in data mining, information retrieval, and pattern recognition. We also give three theoretical interpretations for our algorithm. Experimental results on several toy examples and benchmark datasets have demonstrated the effectiveness of our algorithm.

References

1. Zhu XJ (2006) Semi-supervised learning literature survey, Technical Report Computer Sciences 1530, University of Wisconsin-Madison
2. Belkin M, Niyogi P, Sindhwani V (2006) Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. J Mach Learn Res 7:2399–2434
3. Blum A, Chawla S (2001) Learning from labeled and unlabeled data using graph mincuts. In: ICML, pp 19–26
4. Joachims T (1999) Transductive inference for text classification using support vector machines. In: ICML, pp 200–209
5. Szummer M, Jaakkola T (2001) Partially labeled classification with markov random walks. In: NIPS, pp 945–952
6. Chapelle O, Schölkopf B, Zien A (2006) Semi-supervised learning. MIT Press, Cambridge
7. Seeger M (2000) Learning with labeled and unlabeled data, Technical report, The University of Edinburgh
8. Belkin M, Matveeva I, Niyogi P (2004) Regularization and semi-supervised learning on large graphs. In: COLT, pp 624–638
9. Chung FRK (1997) Spectral graph theory. In: CBMS regional conference series in mathematics, No. 92, American Mathematical Society
10. Zhou D, Bousquet O, Lal TN, Weston J, Schölkopf B (2004) Learning with local and global consistency. In: NIPS
11. Zhu X, Ghahramani Z, Lafferty JD (2003) Semi-supervised learning using Gaussian fields and harmonic functions. In: ICML, pp 912–919
12. Shi J, Malik J (2000) Normalized cuts and image segmentation. IEEE Trans PAMI 22(8):888–905
13. Zhou D, Schölkopf B (2004) Learning from labeled and unlabeled data using random walks. In: DAGM-symposium, pp 237–244
14. Chapelle O, Zien A (2005) Semi-supervised classification by low density separation. In: The tenth international workshop on artificial intelligence and statistics
15. Chapelle O, Weston J, Schölkopf B (2002) Cluster kernels for semi-supervised learning. In: NIPS, pp 585–592
16. Spielman DA, Teng SH (2004) Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In: Annual ACM symposium on theory of computing