Label Propagation through Linear Neighborhoods

Fei Wang, *Student Member*, *IEEE*, and Changshui Zhang, *Member*, *IEEE*

Abstract—In many practical data mining applications such as text classification, unlabeled training examples are readily available, but labeled ones are fairly expensive to obtain. Therefore, semisupervised learning algorithms have aroused considerable interests from the data mining and machine learning fields. In recent years, graph-based semisupervised learning has been becoming one of the most active research areas in the semisupervised learning community. In this paper, a novel graph-based semisupervised learning approach is proposed based on a linear neighborhood model, which assumes that each data point can be linearly reconstructed from its neighborhood. Our algorithm, named *Linear Neighborhood Propagation* (*LNP*), can propagate the labels from the labeled points to the whole data set using these linear neighborhoods with sufficient smoothness. A theoretical analysis of the properties of *LNP* is presented in this paper. Furthermore, we also derive an easy way to extend *LNP* to out-of-sample data. Promising experimental results are presented for synthetic data, digit, and text classification tasks.

Index Terms—Structured data mining, semisupervised learning, graph, label propagation, linear neighborhoods.

1 Introduction

Traditional data mining approaches can be categorized into two categories: 1) One is *supervised learning*, which aims to predict the labels of any new data points from the observed data-label pairs. A supervised learning task is called *regression* when the predicted labels take real values and *classification* when the predicted labels take a set of discrete values. Typical supervised learning methods include *Support Vector Machines* [30] and *Decision Trees* [23]. 2) The other is *unsupervised learning*, the goal of which is just to organize the observed data points with no labels. Typical unsupervised learning tasks include *clustering* [15] and *dimensionality reduction* [25].

In this paper, we will focus on *classification*, which is traditionally a *supervised learning* task. To train a classifier, one needs a collection of labeled data points. However, in many practical applications of pattern classification and data mining, one often faces a lack of sufficient labeled data, since labeling often requires expensive human labor and much time. Meanwhile, in many cases, large numbers of unlabeled data can be far easier to obtain. For example, in text classification, one may have an easy access to a large database of documents (for example, by crawling the Web), but only a small part of them are classified by hand.

Consequently, semisupervised learning methods, which aim to learn from partially labeled data, are proposed. Many conventional semisupervised learning algorithms adopt a generative model for the classifier and employ *Expectation Maximization* (*EM*) [12] to model the label prediction or parameter estimation process. For example, the *mixture of Gaussians* [27], *mixture of experts* [21], and *naive Bayes* [22] have been, respectively, used as the generative model, whereas EM is used to combine labeled and unlabeled data for classification. There are also many other algorithms such as *cotraining* [6], using transductive inference for support vector machines to optimize performance on a specific test set [16], and the *Gaussian process* approach with a *null category noise model* [20]. For a detailed literature survey, one can refer to [39].

In recent years, a prominent achievement in the semisupervised learning area is the development of a *graph-based semisupervised learning* strategy, which models the whole data set as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. $\mathcal{V}$ is the vertex set, which is equivalent to the data set, and $\mathcal{E}$ is the edge set. Associated with each edge $e_{ij} \in \mathcal{E}$ is a nonnegative weight $w_{ij} \geq 0$ reflecting how similar datum i is to datum j.[1] Generally, there are two types of semisupervised learning tasks on graphs: 1) *transduction*, which aims at predicting the labels of the unlabeled vertices only [32], and 2) *induction*, which tries to induce a decision function that has a low error rate on the whole sample space [4]. Clearly, *induction* is more difficult and complicated.

The basic assumption behind graph-based semisupervised learning is the *cluster assumption* [9], which states that two points are likely to have the same class label if there is a path connecting them that passes through the regions of high density only. Zhou et al. [32] further explored the geometric intuition behind this assumption: 1) nearby points are likely to have the same label, and 2) points on the same structure (such as a cluster or a submanifold) are likely to have the same label. Note that the first assumption is local, whereas the second one is global. The cluster assumption tells us to consider both the local and global information contained in the data set during learning.

• *The authors are with the Department of Automation, Tsinghua University, Room 3-120, FIT Building, Beijing, China 100084.*
E-mail: {feiwang03, zcs}@mails.thu.edu.cn.

Manuscript received 8 Sept. 2006; revised 18 May 2007; accepted 28 Aug. 2007; published online 10 Sept. 2007.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-0424-0906.
Digital Object Identifier no. 10.1109/TKDE.2007.190672.

1. The graph $\mathcal{G}$ can either be *directed* (which means $w_{ij} \neq w_{ji}$) [34] or *undirected* (which means $w_{ij} = w_{ji}$) [32]. In this paper, we will focus on the *undirected* case.

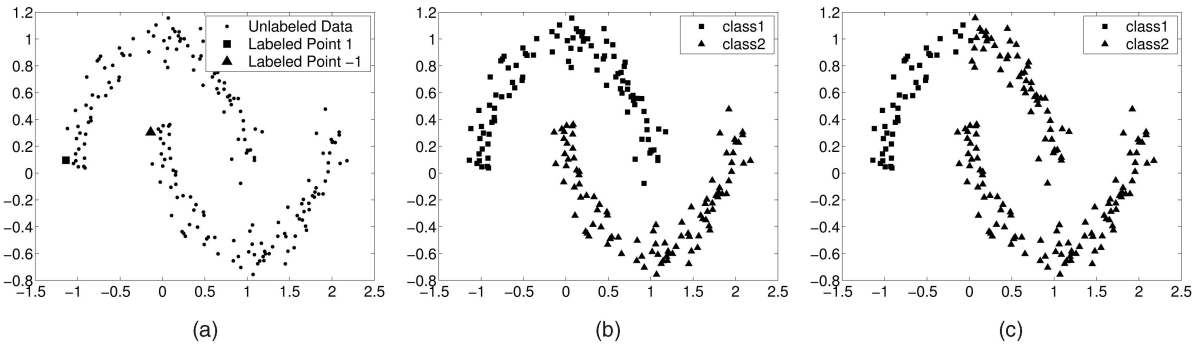


Fig. 1. Classification results on the two-moon pattern using the method in [32], which is a powerful transductive approach operating on graph with the edge weights computed by a Gaussian function. (a) Toy data set with two labeled points. (b) Classification results with $\sigma = 0.15$. (c) Classification results with $\sigma = 0.25$. We can see that a small variation of σ will cause a dramatically different classification result.

It is straightforward to associate cluster assumption with the nonlinear dimensionality reduction methods developed in recent years [24], [29], [2], since the central idea of these methods is to construct a low-dimensional global coordinate system for the data set in which the local structure of the data is preserved. *Laplacian Eigenmaps* [2] is a typical method of such kind. It describes a data set as a graph in the same way as we introduced before and aims at finding a low-dimensional coordinate system in which the data can preserve such a graph structure.

Despite the success of applying graph-based semisupervised learning methods to many fields [39], there are still some problems that are still not properly addressed till now:

1. Although the graph is at the heart of these graph-based methods, its construction has not been studied extensively [39]. More concretely, most of the graph-based semisupervised learning methods [28], [32], [35] adopted a Gaussian function to calculate the edge weights of the graph, that is, the edge links data $\mathbf{x}_i$ and $\mathbf{x}_j$ are computed as

$$e_{ij} = exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/(2\sigma^2)), \tag{1}$$

   where the variance σ is a free parameter that is usually determined empirically. Moreover, σ will affect the classification results significantly. This problem is illustrated by a toy example in Fig. 1. What is worse, according to [32], is that there is no reliable approach for model selection if only very few labeled points are available.
2. Few of the graph-based methods provide an out-of-sample extension [4], that is, most of them only aim to solve the *transduction* task [9], [32], [35].
3. The graph-based methods are generally not robust to the *bridge points* that connect different classes. This can be easily understood from the *label propagation* framework [36], which states that predicting the labels of the unlabeled data on a graph is equivalent to propagating the labels of the labeled data along the edges to the unlabeled ones. Clearly, the existence of the bridge points will cause the labels to wrongly propagate to different classes.

To address the above issues, we propose a novel method called *Linear Neighborhood Propagation* (*LNP*) in this paper. The *LNP* algorithm first *denoises* the data set by a preprocessing step that can remove the bridge points and outliers. Then, it approximates the whole graph by a series of overlapped linear neighborhood patches, and the edge weights in each patch can be solved by a standard quadratic programming procedure. After that, all the edge weights will be aggregated together to form the weight matrix of the whole graph. We prove theoretically that the *Laplacian* matrix of this "pasted" graph can approximate the *Laplacian* matrix of a standard weighted undirected graph. Therefore, this approximated *Laplacian* matrix can be used as a smooth matrix as in standard graph-based semisupervised learning algorithms. We also give an easy method for extending *LNP* to out-of-sample data points. Finally, the experiments on digit and text classification are presented to show the effectiveness of *LNP*.

It is worthwhile to highlight several aspects of the proposed approach here:

1. We propose a preprocessing method that can automatically remove the bridge points and outliers from the data set. Therefore, it is more robust.
2. The storage requirement of *LNP* is smaller than traditional graph-based methods. Assuming that the size of the training set is N and the neighborhood size we choose is k, then the storage requirement of *LNP* is $O(kN)$, which is much smaller than the storage requirement $O(N^2)$ of many traditional graph-based methods.
3. The edge weights of the graph constructed by *LNP* can be solved automatically in closed form once the neighborhood size is fixed. Furthermore, we can show that the final classification result of *LNP* is much more stable with the variation of the neighborhood size compared to the result in [32] with the variation of σ.
4. *LNP* can be easily extended to out-of-sample data points.

The remainder of this paper is organized as follows: We will briefly review some related works in Section 2. Section 3 will show the *LNP* algorithm in detail, and the analysis of *LNP* will be presented in Section 4. Section 5 presents the experimental results on synthetic and real-world data, followed by the conclusions and future works in Section 6.

TABLE 1
Popular Graph-Based Semisupervised Learning Objectives and Their Corresponding Parameters

Method	$\mathcal{C}(\cdot)$	$\mathbf{S}$	η	β	Constraints
Graph Mincuts [7]	Quadratic	$\mathbf{L}$	∞	1/2	$f_i \in \{0,1\}$
Label Propagation [36]	Quadratic	$\mathbf{L}$	∞	1/2	$None$
Gaussian Random Fields [35]	Quadratic	$\mathbf{L}$	∞	1/2	$None$
Discrete Regularization [32]	Quadratic	$\hat{\mathbf{L}}$	1	$\beta > 0$	$None$
Tikhonov Regularization [3]	Quadratic	$\mathbf{L}^p$ or $exp(-t\mathbf{L})$	$1/l$	$\beta \in \mathbb{R}$	$None$
Interpolated Regularization[3]	Quadratic	$\mathbf{L}^p$ or $exp(-t\mathbf{L})$	∞	1	$\sum_i f_i = 0$
Spectral Graph Transducer[17]	Ellipsoidal	$\mathbf{L}$ or $\hat{\mathbf{L}}$	$c > 0$	1	$\sum_i f_i = 0,\ \sum_i f_i^2 = 1$
Manifold Regularization[4]	Arbitrary	$\frac{\gamma_A}{\gamma_I}\mathbf{K} + \mathbf{L}$	1	γ_I	$\sum_i f_i = 0$

2 Notations and Related Works

In this section, we will briefly review some previous works that are closely related to this paper. First, let us introduce some notations. $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_l, \mathbf{x}_{l+1}, \cdots, \mathbf{x}_n\}$ represents a set of n data objects in $\mathbb{R}^d$, and $\mathcal{L} = \{1, -1\}$ is the label set (we consider the two-class case for the moment). The first l points $\mathcal{X}_L = \{\mathbf{x}_i\}_{i=1}^{l}$ are labeled as $t_i \in \mathcal{L}$, and the remaining points $\mathcal{X}_U = \{\mathbf{x}_u\}_{u=l+1}^{n}$ are unlabeled.

Since we will concentrate on *graph-based semisupervised learning* in this paper, we will briefly review some of the related works in this section. For more works on semisupervised learning, one can refer to the excellent literature survey in [39].

As we have mentioned in Section 1, the common denominator of graph-based methods is to model the whole data set as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with the vertex set $\mathcal{V} = \mathcal{X}$, and the weight w_{ij} on edge $e_{ij} \in \mathcal{E}$ reflects the similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$. Generally, the graph-based methods can be viewed as estimating a classification function f on $\mathcal{G}$ subject to the following: 1) it should be close to the given labels on the labeled vertices, and (2) it should be smooth on the whole graph. This will result in a *regularization framework* that aims to minimize

$$\mathcal{J}(f) = \eta\mathcal{C}(f_{\mathcal{X}_L}) + \beta\mathcal{S}(f), \tag{2}$$

where the $\mathcal{C}(\cdot)$ is a loss function measuring the inconsistency between the predicted and initial labels on $\mathcal{X}_L$, and $\mathcal{S}(f)$ is a regularizer to punish the smoothness of the f, which usually takes the form

$$\mathcal{S}(f) = \mathbf{f}^T\mathbf{S}\mathbf{f}, \tag{3}$$

where $\mathbf{f} = [f(\mathbf{x}_1), \cdots, f(\mathbf{x}_n)]^T$, and $\mathbf{S}$ is an $n \times n$ smoothness matrix. Many traditional graph-based methods are similar to each other with different choices of loss functions and regularizers. Table 1[2] shows us some examples of those methods.

As we mentioned in Section 1, *graph construction* is at the heart of *graph-based* methods. However, there are only limited works on such of a research topic. For example, Zhu and Ghahramani [36] proposed to determine the edge weights on the graph using *Minimum Spanning Trees* and to learn those hyperparameters by some *sampling* techniques [37], Balcan et al. [1] built graphs for video surveillance using strong domain knowledge, Carreira-Perpinan and Zemel [8] built robust graphs from multiple minimum spanning trees by perturbation and edge removal, and Kapoor et al. [19] learned the hyperparameters in computing the edge weights via a *Gaussian Process*. These methods are either heuristic or complicated. In this paper, we propose a novel simple method for constructing the graph, and we also provide the theoretical justifications of the feasibility of our method.

2. In Table 1, $\mathbf{L}$ represents the *combinatorial graph Laplacian* [35], $\hat{\mathbf{L}}$ represents the *normalized graph Laplacian* [32], and K is the induced kernel of f in the *reproducing kernel Hilbert space* [4].

3 The Algorithm

We have briefly reviewed some related works in the last section. In this section, we will formally present our *LNP* algorithm, which aims to solve a *transduction* problem. In Section 4.4, we will give an easy way to extend *LNP* for out-of-sample data.

3.1 Graph Construction

As we stated earlier, *graph construction* is at the heart of graph-based methods. Most of the traditional methods adopt (1) to compute the weights on the edges containing the graph. However, as pointed out by [32], there is no reliable approach for model selection if only very few labeled points are available (that is, it is hard to determine an optimal σ in (1)). Moreover, we found in our experiments that even a small perturbation of σ could make the classification results dramatically different. Thus, we derive a more reliable and stable way to construct the graph $\mathcal{G}$ here.

Instead of considering pairwise relationships as (1) in traditional graph-based methods, we propose to use the neighborhood information of each point to construct $\mathcal{G}$. As in [24], we assume that all of these neighborhoods are linear, that is, each data point can be optimally reconstructed using a linear combination of its neighbors.[3] Hence, our objective is to minimize

$$\varepsilon = \sum_i \left\| \mathbf{x}_i - \sum\nolimits_{i_j : \mathbf{x}_{i_j} \in \mathcal{N}(\mathbf{x}_i)} w_{ii_j}\mathbf{x}_{i_j} \right\|^2, \tag{4}$$

where $\mathcal{N}(\mathbf{x}_i)$ represents the neighborhood of $\mathbf{x}_i$, $\mathbf{x}_{i_j}$ is the jth neighbor of $\mathbf{x}_i$, and w_{ii_j} is the contribution of $\mathbf{x}_{i_j}$ to $\mathbf{x}_i$. We further constrain $\sum_{i_j \in \mathcal{N}(\mathbf{x}_i)} w_{ii_j} = 1$, $w_{ij} \geq 0$. Obviously, the more similar $\mathbf{x}_{i_j}$ is to $\mathbf{x}_i$, the larger w_{ii_j} will be (as an extreme case, when $\mathbf{x}_i = \mathbf{x}_{i_k} \in \mathcal{N}(\mathbf{x}_i)$, then $w_{ii_k} = 1$, $w_{ii_j} = 0$,

3. Although this assumption seems very strong, it is reasonable. From the analysis in [2], we know that based on such linear neighborhood assumption, we can use the linear neighborhood patch to approximate the tangent space at each data point.

$i_j \neq i_k$, and $\mathbf{x}_{i_j} \in \mathcal{N}(\mathbf{x}_i)$ is the optimal solution). Thus, w_{ii_j} can be used to measure how similar $\mathbf{x}_{i_j}$ is to $\mathbf{x}_i$. One issue that should be addressed here is that usually $w_{ii_j} \neq w_{i_ji}$. It can be easily inferred that

$$\begin{aligned}\varepsilon_i &= \left\|\mathbf{x}_i - \sum_{i_j:\mathbf{x}_{i_j}\in\mathcal{N}(\mathbf{x}_i)} w_{ii_j}\mathbf{x}_{i_j}\right\|^2 \\ &= \left\|\sum_{i_j:\mathbf{x}_{i_j}\in\mathcal{N}(\mathbf{x}_i)} w_{ii_j}\left(\mathbf{x}_i - \mathbf{x}_{i_j}\right)\right\|^2 \\ &= \sum_{i_j,i_k:\mathbf{x}_{i_j},\mathbf{x}_{i_k}\in\mathcal{N}(\mathbf{x}_i)} w_{ii_j}w_{ii_k}\left(\mathbf{x}_i - \mathbf{x}_{i_j}\right)^T\left(\mathbf{x}_i - \mathbf{x}_{i_k}\right) \\ &= \sum_{i_j,i_k:\mathbf{x}_{i_j},\mathbf{x}_{i_k}\in\mathcal{N}(\mathbf{x}_i)} w_{ii_j}G^i_{i_ji_k}w_{ii_k},\end{aligned} \tag{5}$$

where $G^i_{i_ji_k}$ represents the $(j,k)\text{th}$ entry of the *local Gram matrix* $\mathbf{G}^i$ of the form

$$\begin{bmatrix} (\mathbf{x}_i - \mathbf{x}_{i_1})^T(\mathbf{x}_i - \mathbf{x}_{i_1}) & \cdots & (\mathbf{x}_i - \mathbf{x}_{i_1})^T(\mathbf{x}_i - \mathbf{x}_{i_k}) \\ (\mathbf{x}_i - \mathbf{x}_{i_2})^T(\mathbf{x}_i - \mathbf{x}_{i_1}) & \cdots & (\mathbf{x}_i - \mathbf{x}_{i_2})^T(\mathbf{x}_i - \mathbf{x}_{i_k}) \\ \vdots & \ddots & \vdots \\ (\mathbf{x}_i - \mathbf{x}_{i_k})^T(\mathbf{x}_i - \mathbf{x}_{i_1}) & \cdots & (\mathbf{x}_i - \mathbf{x}_{i_k})^T(\mathbf{x}_i - \mathbf{x}_{i_k}) \end{bmatrix}, \tag{6}$$

where $K = |\mathcal{N}(\mathbf{x}_i)|$ is the size of $\mathbf{x}_i$'s neighborhood. Thus, the reconstruction weights of each data object can be resolved by the following n standard quadratic programming problems:

$$\begin{aligned} min_{w_{ii_j}} \quad & \sum_{i_j,i_k:\mathbf{x}_{i_j},\mathbf{x}_{i_k}\in\mathcal{N}(\mathbf{x}_i)} w_{ii_j}G_{i^i_ji_k}w_{ii_k} \\ s.t. \quad & \sum_{i_j} w_{ii_j} = 1, \quad w_{ii_j} \geq 0. \end{aligned} \tag{7}$$

After all of the reconstruction weights are computed, we will construct a sparse matrix $\mathbf{W}$ by

$$W(i,j) = w_{ij}. \tag{8}$$

Intuitively, this $\mathbf{W}$ can be treated as the weight matrix of $\mathcal{G}$. Moreover, the way we construct the whole graph is to first shear the whole graph into a series of overlapped linear patches and then paste them together.

3.2 Label Propagation through Linear Neighborhoods

After the graph has been constructed, we have to make use of it to predict the labels of the unlabeled vertices. Here, we propose a scheme similar to the one in *Label Propagation* [36], which can iteratively propagate the labels of the labeled data to the remaining unlabeled data $\mathcal{X}_U$ on the constructed graph.

Let $\mathcal{F}$ denote the set of classifying functions defined on $\mathcal{X}$, $\forall\, f \in \mathcal{F}$ can assign a real value f_i to every point $\mathbf{x}_i$. The label of the unlabeled data point $\mathbf{x}_u$ is determined by the sign of $f_u = f(\mathbf{x}_u)$ (let us only consider the two-class case for the time being).

In each propagation step, we let each data object *absorb* a fraction of label information from its neighborhood and *retain* some label information of its initial state. Therefore, the label of $\mathbf{x}_i$ at time $m+1$ becomes

$$f_i^{m+1} = \alpha\sum_{j:\mathbf{x}_j\in\mathcal{N}(\mathbf{x}_i)} w_{ij}f_j^m + (1-\alpha)t_i, \tag{9}$$

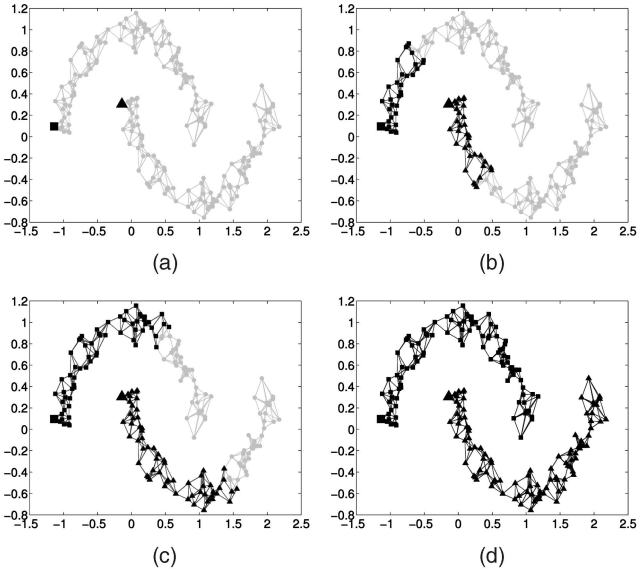


Fig. 2. Transduction results on the two-moon pattern using *LNP*. The number of nearest neighbors $k = 4$. $\alpha = 0.99$. The black squares represent class 1 (whose labels are positive), and the black circles represent class 2 (whose labels are negative). The black dots represent data points whose labels are zero. (a) The initial state, that is, $\mathrm{t} = 0$, only two points are labeled. $m = 0$. (b) $\mathrm{t} = 15$ and $m = 10$. (c) $t = 20$ and $m = 20$. (d) $t = 50$ and $m = 50$.

where $0 < \alpha < 1$ is the fraction of label information that $\mathbf{x}_i$ receives from its neighbors. Let $\mathbf{t} = (t_1, t_2, \cdots t_n)^T$ with $t_i \in \mathcal{L}(i \leq l)$, $u_u = 0(l+1 \leq u \leq n)$. $\mathbf{f}^m = (f_1^m, f_2^m, \cdots, f_n^m)^T$ is the prediction label vector at iteration t, and $\mathbf{f}^0 = \mathbf{t}$. Then, we can rewrite our iteration equation as

$$\mathbf{f}^{m+1} = \alpha\mathbf{W}\mathbf{f}^m + (1-\alpha)\mathbf{t}. \tag{10}$$

We will use (10) to update the labels of each data object until convergence; here, "convergence" means that the predicted labels of the data will not change in several successive iterations (the convergence analysis of our algorithm will be presented in Section 4.1).

To provide some intuition on how *LNP* works, we give a toy example, shown in Fig. 2. In the figures, we can clearly see how the labels propagate through the linear neighborhoods.

It is easy to extend *LNP* to multiclass classification problems. Suppose there are c classes, and the label set becomes $\mathcal{L} = \{1, 2, \cdots, c\}$. Let $\mathcal{M}$ be a set of $n \times c$ matrices with nonnegative real-valued entries. Any matrix $\mathbf{F} = [\mathbf{F}_1^T, \mathbf{F}_2^T, \cdots, \mathbf{F}_n^T]^T \in \mathcal{M}$ corresponds to a specific classification on $\mathcal{X}$ that labels $\mathbf{x}_i$ as $y_i = argmax_{j\leq c}\mathbf{F}_{ij}$. Thus, $\mathbf{F}$ can also be viewed as a function that assign labels for each data point. Initially, we set $\mathbf{F}_0 = \mathbf{T}$, where $\mathbf{T}_{ij} = 1$ if $\mathbf{x}_i$ is labeled as j, and $\mathbf{T}_{ij} = 0$ otherwise, and for unlabeled points, $\mathbf{T}_{uj} = 0(1 \leq j \leq c)$. The main procedure of *LNP* is summarized in Table 2.

4 Analysis and Extension of Linear Neighborhood Propagation

In this section, we will analyze some properties of *LNP*. First, we will analyze the convergence property of the

TABLE 2
Linear Neighborhood Propagation

Input: $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_l, \mathbf{x}_{l+1}, \cdots, \mathbf{x}_n\} \in \mathbb{R}^d$, $\{\mathbf{x}_i\}_{i=1}^{l}$ are labeled, $\{\mathbf{x}_u\}_{u=l+1}^{n}$ are unlabeled. The initial label matrix $\mathbf{Y}$. The number of the nearest neighbors k. The constant α.

Output: The labels of all the data points.

1. Construct the neighborhood graph by solving Eq.(7) for the reconstruction weights of each data object from its k-nearest neighbors. Form the weight matrix $(\mathbf{W})_{ij} = w_{ij}$
2. Construct the propagation matrix $\mathbf{P} = \mathbf{W}$. And iterate $\mathbf{F}_{t+1} = \alpha\mathbf{P}\mathbf{F}_t + (1-\alpha)\mathbf{Y}$ until convergence.
3. Let $\mathbf{F}^\star$ be the limit of the sequence $\mathbf{F}_t$. Output the labels of each data object $\mathbf{x}_i$ by $y_i = argmax_{j \leqslant c}\mathbf{F}^\star_{ij}$

iterative *LNP* shown in Table 2 and propose a one-shot *LNP*. Second, we will derive *LNP* from a regularization framework and thus provide a theoretical guarantee of the feasibility of *LNP*. Then, we analyze the robustness of *LNP* and point out its potential problems; moreover, we also propose a simple preprocessing procedure to make *LNP* much more robust. Finally, we will give a simple way to extend *LNP* to out-of-sample data.

4.1 Convergence Analysis of Linear Neighborhood Propagation

In this section, we will prove that the iterative process shown in Table 2 will converge to a fixed point. More concretely, we have the following theorem:

Theorem 1. *The sequence* $\{\mathbf{f}^m\}$ *computed by (10) will converge to*

$$\mathbf{f}^\star = (1-\alpha)(\mathbf{I} - \alpha\mathbf{W})^{-1}\mathbf{y} \tag{11}$$

when $m \longrightarrow \infty$ *and* $\mathbf{f}^0 = \mathbf{y}$.

Proof. By (10) and the initial condition that $\mathbf{f}^0 = \mathbf{y}$, we have

$$\mathbf{f}^m = (\alpha\mathbf{W})^{m-1}\mathbf{y} + (1-\alpha)\sum_{i=0}^{m-1}(\alpha\mathbf{W})^i\mathbf{y}. \tag{12}$$

Since $w_{ij} \geq 0$ and $\sum_j w_{ij} = 1$, from the theorem of *Perron-Frobenius* [14], we know that the spectral radius of $\mathbf{W}$ or $\rho(\mathbf{W}) \leq 1$. In addition, $0 < \alpha < 1$; thus,

$$\lim_{m\to\infty} (\alpha\mathbf{W})^{m-1} = 0,$$
$$\lim_{m\to\infty} \sum_{i=0}^{m-1}(\alpha\mathbf{W})^i = (\mathbf{I} - \alpha\mathbf{W})^{-1},$$

where $\mathbf{I}$ is the identity matrix of order n. Clearly, the sequence $\{\mathbf{f}^m\}$ will converge to

$$\mathbf{f}' = \lim_{m\to\infty}\mathbf{f}^m = (1-\alpha)(\mathbf{I} - \alpha\mathbf{W})^{-1}\mathbf{y},$$

which proves the theorem. □

Since in classification, we only use the sign of f_i to determine the label of $\mathbf{x}_i$, the constant term $1 - \alpha \geq 0$ will not affect the signs of $\mathbf{f}^\star = (\mathbf{I} - \alpha\mathbf{W})^{-1}\mathbf{y}$. Hence, we can use $\mathbf{f}^\star$ as the classification function, which results in a "one-shot" *LNP*, that is, we can predict the labels of all the data objects in one step.

Similarly, for the multiclass case, we may simply replace $\mathbf{y}$ by $\mathbf{Y}$ in (13), which results in

$$\mathbf{F}' = (1-\alpha)(\mathbf{I} - \alpha\mathbf{W})^{-1}\mathbf{Y}. \tag{13}$$

The labels of each data object can be determined by $y_i = argmax_{j\leq c}\mathbf{F}'_{ij}$.

4.2 Regularization Framework of Linear Neighborhood Propagation

In this section, we will show that *LNP* can also be derived from a regularization framework. First, let us introduce a lemma [2].

Lemma 1. *Assume that the data in* $\mathcal{X}$ *are sampled from a smooth manifold* $\mathcal{M}$, f *is the classification function defined on* $\mathcal{M}$, $\mathbf{f} = (f(\mathbf{x}_1), f(\mathbf{x}_2), \cdots, f(\mathbf{x}_n))^T$ *is the classification vector on* $\mathcal{X}$, $\mathbf{W}$ *is the matrix constructed in the way of (8), and* $\mathcal{L}$ *is the Laplace-Beltrami operator defined on* $\mathcal{M}$*; then, we have*

$$(\mathbf{I} - \mathbf{W})\mathbf{f} \approx \mathcal{L}f \tag{14}$$

and

$$\Big((\mathbf{I} - \mathbf{W}) + (\mathbf{I} - \mathbf{W})^T\Big)\mathbf{f} \approx 2\mathcal{L}f. \tag{15}$$

Based on the above lemma, we can easily derive the following theorem:

Theorem 2. *The prediction result of LNP (11) can approximate the solution that minimizes the following cost function:*

$$\mathcal{Q}(\mathbf{f}) = \mathbf{f}^T(\mathbf{I} - \mathbf{W})\mathbf{f} + \mu(\mathbf{f} - \mathbf{y})^T(\mathbf{f} - \mathbf{y}), \tag{16}$$

where $\mu > 0$ *is a constant.*

Proof. The derivative of $\mathcal{Q}(\mathbf{f})$ with respect to $\mathbf{f} = (f_1, f_2, \cdots, f_n)^T$ is

$$\frac{\partial\mathcal{Q}(\mathbf{f})}{\partial\mathbf{f}} = \Big[(\mathbf{I} - \mathbf{W}) + (\mathbf{I} - \mathbf{W})^T\Big]\mathbf{f} + 2\mu(\mathbf{y} - \mathbf{f}). \tag{17}$$

Using the conclusion of Lemma 1, we can derive that

$$\Big[(\mathbf{I} - \mathbf{W}) + (\mathbf{I} - \mathbf{W})^T\Big]\mathbf{f} \approx 2\mathcal{L}f \approx 2(\mathbf{I} - \mathbf{W})\mathbf{f}, \tag{18}$$

where $\mathcal{L}$ is the *Laplace-Beltrami* operator defined on the manifold. Therefore, by setting

$$\frac{\partial\mathcal{Q}(\mathbf{f})}{\partial\mathbf{f}} = \Big[(\mathbf{I} - \mathbf{W}) + (\mathbf{I} - \mathbf{W})^T\Big]\mathbf{f} + 2\mu(\mathbf{y} - \mathbf{f})$$
$$\approx 2(\mathbf{I} - \mathbf{W})\mathbf{f} - 2\mu(\mathbf{f} - \mathbf{y}) = 0,$$

we can get

$$\mathbf{f} = (1-\alpha)(\mathbf{I} - \alpha\mathbf{W})^{-1}\mathbf{y},$$

which is identical to (11) with $\alpha = \frac{1}{1+\mu}$. □

Intuitively, the first term of $\mathcal{Q}(\mathbf{f})$ in (16) describes the total variation of the data labels with respect to the neighborhood structures, and we will call it the *smoothness term*. The second term measures how well the predicted labels fit the original labels; thus, we call it the *fit term*. Since $\mathbf{f}$ can be thought of as a function defined on a graph where f_i is the function value at node $\mathbf{x}_i$, the matrix $\mathbf{I} - \mathbf{W}$ can be thought of as an operator acting on $\mathbf{f}$. As noted by Section 3, the graph that *LNP* constructs is just a pasted graph. From such a viewpoint, the matrix $\mathbf{I} - \mathbf{W}$ can be regarded as the

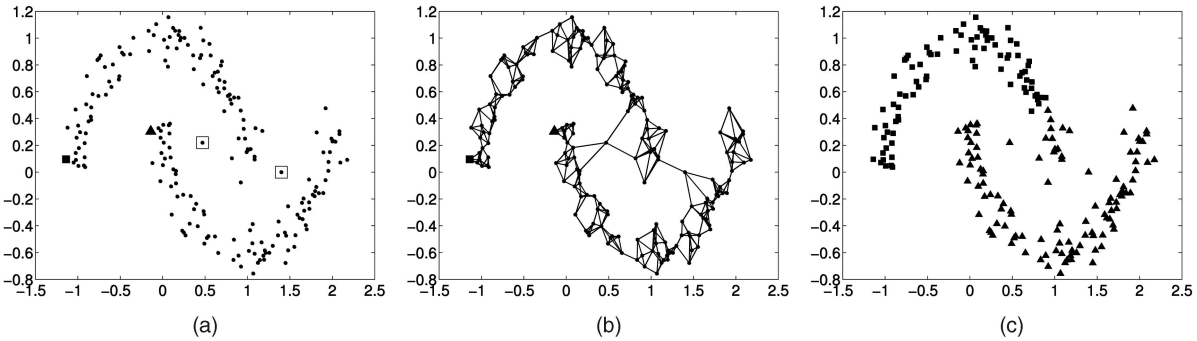


Fig. 3. Classification results on the two-moon pattern with two bridge points using *LNP*. (a) Toy data set with two bridge points, which are in black rectangles. (b) The four-nearest-neighbor graph of the data set. (c) Classification result with the size of the neighborhood equal to four. We can see that the bridge points will cause the label of the lower moon to wrongly propagate to the upper moon.

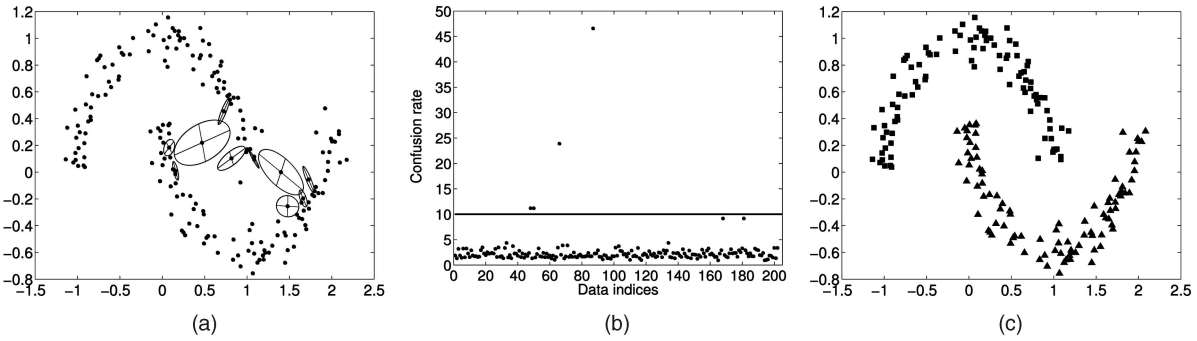


Fig. 4. Classification results on the toy data set shown in Fig. 3a using *LNP* after preprocessing. (a) Local neighborhood shapes of the bridge points and their neighbors, with a neighborhood size of $K = 4$. (b) The *confusion rate* of the data points, where the abscissa represent the data indices, and the red line denotes the threshold $\delta = 10$. (c) Classification result with *LNP* after eliminating the data points whose $c_i > 10$.

graph Laplacian of this "pasted" graph intuitively. Furthermore, this loss function can also be unified in the framework in (3) with the *smoothness matrix* $\mathbf{S} = \mathbf{I} - \mathbf{W}$, $\alpha = 1$, and $\beta = \mu$. That is to say, *LNP* can also be derived from a regularization framework.

4.3 The Robustness of Linear Neighborhood Propagation

A potential problem which exists in *LNP* is that it may produce wrong prediction results when there are some points connecting different data classes (which are usually called *bridge points*). Fig. 3 gives us an intuitive illustration of such a problem.

Therefore, the existence of bridge points will bias the final results. It will be better if we can preprocess the data set by eliminating these bridge points. Since the bridge points usually reside on the low-density places, the local shape and size of their neighborhoods will be different from the neighborhoods of the other points. Based on this intuitive observation, we propose to make use of the local information of each data point to discriminate the bridge points from the others. Moreover, the work in [18] tells us that the local information of a data point can be reflected from its local covariance matrix. More concretely, define the local covariance matrix of $\mathbf{x}_i$ as

$$\mathbf{C}_i = \sum_{\mathbf{x}_j \in \mathcal{N}(\mathbf{x}_i)} (\mathbf{x}_j - \mathbf{x}_i)(\mathbf{x}_j - \mathbf{x}_i)^T,$$

where $K = |\mathcal{N}(\mathbf{x}_i)|$ is the size of $\mathcal{N}(\mathbf{x}_i)$. Then, the neighborhood shape of $\mathbf{x}_i$ can be reflected by $\mathbf{C}_i$'s eigenvalue and eigenvectors.

Let us take the data set shown in Fig. 3a as an example. The neighborhood shapes of the bridge points and their neighbors are plotted in Fig. 4a. For each $\mathbf{x}_i$, its neighborhood is described by an ellipse centered at $\mathbf{x}_i$, whose semiaxes' length are equal to half of $\mathbf{C}_i$'s eigenvalues, and its semiaxes' directions are determined by $\mathbf{C}_i$'s eigenvector. From this toy example, we can easily discover that the ellipses standing for the neighborhoods of the bridge points are significantly larger than other ellipses. Hence, we propose the following *confusion rate*:

$$c_i = \sum\nolimits_{k=1}^{min(dim(\mathbf{x}_i),K)} \frac{\lambda_{ik}}{\overline{\lambda}_{ik}} \tag{19}$$

to discriminate the bridge points from the others, where λ_{ik} is the kth eigenvalue of $\mathbf{C}_i$, and $\overline{\lambda}_{ik} = \sum_{\mathbf{x}_j \in \mathcal{N}(\mathbf{x}_i)} \lambda_{jk}$, $dim(\mathbf{x}_i)$ represents the dimensionality of $\mathbf{x}_i$. For high-dimensional data, the eigendecomposition of $\mathbf{C}_i$ has high computational complexity. However, since in this case

$$min(dim(\mathbf{x}_i), K) = K,$$

the *confusion rate* of $\mathbf{x}_i$ is

$$c_i = \sum\nolimits_{k=1}^{K} \frac{\lambda_{ik}}{\overline{\lambda}_{ik}}. \tag{20}$$

Thus, λ_{ik} can be computed by eigendecomposing the local Gram matrix (6), since

$$\mathbf{C}_i \mathbf{v}_i = \mathbf{X}_i \mathbf{X}_i^T \mathbf{v}_i = \lambda_i \mathbf{v}_i, \tag{21}$$

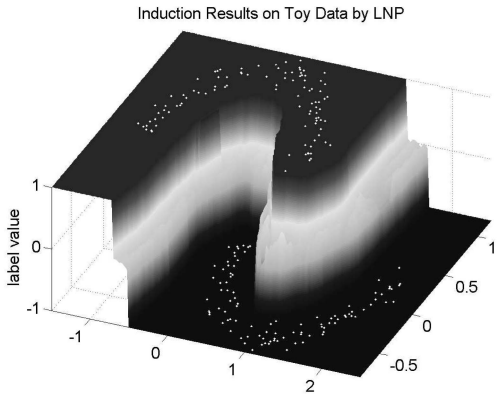


Fig. 5. Induction results for all of the data points in $\{(x,y)|x \in [-1.5, 2.5], y \in [-0.8, 1.2]\}$ using (24). The black dots represent the data points in the training data set. The z-axis indicates the predicted label values associated with different colors. We can see that the induced decision boundary is intuitively satisfying since it is in accordance with the intrinsic structure of the training data set.

where $\mathbf{X}_i = (\mathbf{x}_{i1} - \mathbf{x}_i, \mathbf{x}_{i2} - \mathbf{x}_i, \cdots, \mathbf{x}_{iK} - \mathbf{x}_i)$, and $\mathbf{x}_{ij}$ is the jth neighbor of $\mathbf{x}_i$. Multiplying $\mathbf{X}_i^T$ on the both sides of (22), we can get

$$\mathbf{X}_i^T \mathbf{C}_i \mathbf{v}_i = \mathbf{X}_i^T \mathbf{X}_i \left(\mathbf{X}_i^T \mathbf{v}_i\right) = \mathbf{G}^i \left(\mathbf{X}_i^T \mathbf{v}_i\right) = \lambda_i \left(\mathbf{X}_i^T \mathbf{v}_i\right), \quad (22)$$

that is, the top eigenvalues of $\mathbf{C}_i$ are also the top eigenvalues of $\mathbf{G}^i$. In real-world applications, we only need to determine a threshold δ such that we can throw the points satisfying $c_i > \delta$ and use the remaining points for classification.

4.4 Induction for Out-of-Sample Data

Until now, we have introduced the main procedure and preprocessing procedure of *LNP*, but it is only for transduction. In this section, we will propose an effective method to extend *LNP* to out-of-sample data.

As stated in Section 1, according to [11], we need to do two things for generalizing *LNP* to out-of-sample data: 1) use the same type of smoothness criterion as in (3) for a new testing point $\mathbf{y}$ and 2) ensure that the inclusion of $\mathbf{y}$ will not affect the original $\mathcal{Q}(f)$ value of the training data set.

The smoothness criterion for a new test point $\mathbf{y}$ is

$$\mathcal{Q}(f(\mathbf{y})) = \sum_{i:\mathbf{x}_i \in \mathcal{X}, \mathbf{x}_i \in \mathcal{N}(\mathbf{y})} w(\mathbf{y}, \mathbf{x}_i)(f(\mathbf{y}) - f_i)^2. \quad (23)$$

Since $\mathcal{Q}(f(\mathbf{y}))$ is convex in $f(\mathbf{y})$, it is minimized when

$$f(\mathbf{y}) = \sum\nolimits_{i:\mathbf{x}_i \in \mathcal{X}, \mathbf{x}_i \in \mathcal{N}(\mathbf{y})} w(\mathbf{y}, \mathbf{x}_i) f_i. \quad (24)$$

Interestingly, this is exactly the formula when the label of $\mathbf{y}$ can be optimally reconstructed from the labels of its neighbors in the training data set, that is,

$$f(\mathbf{y}) = \min_{f(\mathbf{y})} \left\| f(\mathbf{y}) - \sum_{i:\mathbf{x}_i \in \mathcal{X}, \mathbf{x}_i \in \mathcal{N}(\mathbf{y})} w(\mathbf{y}, \mathbf{x}_i) f_i \right\|^2. \quad (25)$$

To illustrate the effectiveness of this induction method, we also use the problem shown in Fig. 1a. This is a two-class classification problem with only two points labeled, one for each class. We first adopt the standard *LNP* procedure to predict the labels of the unlabeled points and, then, (24) is used for inducting the labels of all the points in the region $\{(x,y)|x \in [-1.5, 2.5], y \in [-0.8, 1.2]\}$. The results can be seen in Fig. 5.

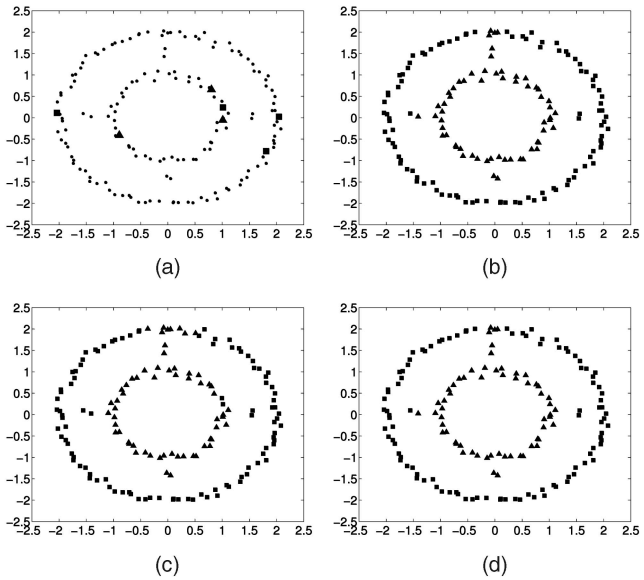


Fig. 6. Transduction results on the two-circle data. The black squares and black triangles represent two classes. (a) The original data set, with seven data points labeled. (b) Classification results with Zhou et al.'s *consistency* method. (c) Classification results using Zhu et al.'s *GRF* approach. (d) Classification results produced by the *LNP* method without preprocessing and with a neighborhood size of $K = 5$.

5 Experiments

In this section, we give a set of experiments where we used *LNP* for semisupervised classification, including toy examples, digits recognition, and text classification. In all of the experiments in this section, α is set to 0.99.

5.1 Toy Problems

In this section, we shall use one toy example to illustrate the effectiveness of our *LNP* method. The original data set is shown in Fig. 6a, which is a two-circle pattern with 160 data points. The outside circle contains 101 points, and the inside circle contains 51 points. There are also eight bridge points that connect these two circles.

As shown in Fig. 6a, seven points are randomly labeled initially, three on the outside cluster and four on the inside cluster. Moreover, there is one data point wrongly labeled in the inside cluster. Fig. 6b shows the transduction results using Zhou et al.'s *consistency* method [32]. Fig. 6c is the transduction result given by the method of *Gaussian Random Fields* (*GRFs*) [35]. The hyperparameters (that is, the variance of the Gaussian function) in these methods are determined by five-fold cross validation. Fig. 6d shows the classification results by the basic *LNP* procedure without the preprocessing step of eliminating the bridge points, and the neighborhood size is set to five manually. We can see that all of these methods will wrongly predict the labels of some data points in the outside circle because of the existence of the bridge points. Moreover, the *GRF* method is more brittle in that it cannot correct the labels of the

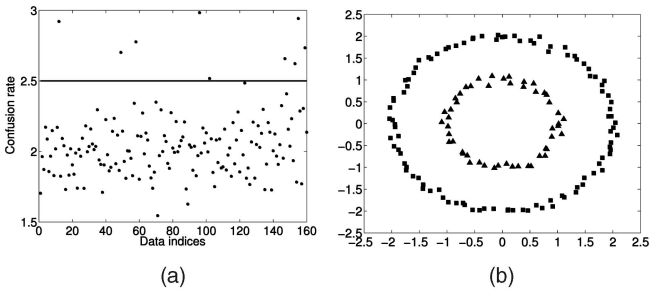


Fig. 7. The preprocessing step of *LNP*. (a) The confusion rate of the data set in Fig. 6a, the line denotes the threshold of 2.5. (b) The classification result of *LNP* after preprocessing.

wrongly labeled points, since their labels will remain fixed throughout the classification procedure.

Fig. 7a shows the confusion rate of the data set in Fig. 6a. Moreover, we set the threshold $\delta = 2.5$ empirically.[4] Fig. 6b illustrates the classification results of *LNP* after eliminating all the points whose *confusion rate* is larger than 2.5; we can see that all the data points are correctly classified.

5.2 Digit Recognition

In this case study, we will focus on the problem of classifying handwritten digits. The data set we adopt is the *USPS*[5] handwritten 16×16 digits data set. The images of digits 1, 2, 3, and 4 are used in these experiments as four classes, and there are 1,269, 929, 824, and 852 examples in each class, with a total of 3,874.

We used the *Nearest Neighbor* classifier and *one*-versus-*rest SVMs* [26] as baselines.[6] The width of the RBF kernel for *SVM* was set to five using five-fold cross validation. For comparison, we also provide the classification results achieved by Zhou et al.'s *consistency* method [32], Zhu et al.'s *Gaussian fields* approach [35], and Belkin et al.'s *Manifold Regularization* method (we adopted *Laplacian Regularized Least Square* (*LapRLS*) as the representative) [4]. The affinity matrix in all of these methods was constructed by a Gaussian function whose variance is set by five-fold cross validation. Note that the diagonal elements of the affinity matrix in Zhou et al.'s consistency method were set to 0. For *LapRLS*, the base kernel was also selected to be Gaussian, and all of the other hyperparameters were set by grid search as in [10]. In *LNP*, the number of nearest neighbors k was set to five manually when constructing the graph. When using preprocessing, the threshold of the *confusion rate* is chosen such that 2 percent of the data are thrown.[7] The recognition accuracies averaged over 50 independent trials are summarized in Fig. 8a, from which we can clearly see the advantage our method.

4. Here, we chose δ to be the smallest value that after denoising, all of the remaining data points are correctly classified.

5. http://www.kernel-machines.org/data.html.

6. Here, we choose these two classifiers as the baselines because of their popularity and empirical success. Note that many semisupervised learning papers also adopt them as baselines, such as [32], [35].

7. In fact, there is a trade-off between the robustness and effectiveness of our method. On one hand, the elimination of the bridge points can improve the performance of *LNP*; on the other hand, eliminating too many points will destroy the structure of the graph, which may degrade the performance of *LNP*. In our experiments, we find that 2 percent is a good point for such trade-off.

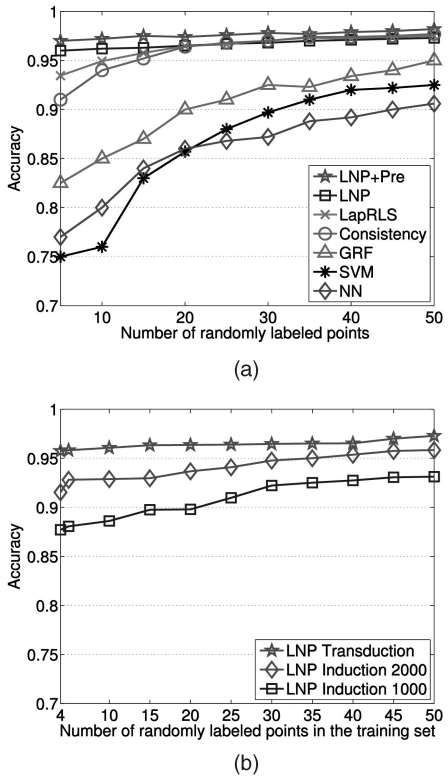


Fig. 8. Digit recognition on the *USPS* data set. A subset containing digits 1 to 4 was adopted. (a) The recognition accuracies for different algorithms. (b) The recognition accuracies with 1,000 and 2,000 points selected for training (transduction) and the remaining data points for testing (induction). In both figures, the abscissa represents the number of randomly labeled data in the training set (we guarantee that there is at least one labeled point in each class), and the ordinate is the total recognition accuracy value averaged over 50 independent runs.

It is interesting to discover that our *LNP* method is very stable, that is, even when we only label a very small fraction of the data, it can still get a high recognition accuracy. We believe this is because the image data of different digits reside on different submanifolds (whereas the image data of the same digit may reside on the same submanifold [24]). *LNP* can effectively reveal these manifold structures so that only a few labeled points are sufficient for predicting the labels of the remaining unlabeled points. After preprocessing, the recognition accuracy becomes higher since the bridge points that connect different digit submanifolds are eliminated.

We also test the effectiveness of the induction method for *LNP* introduced in Section 4.4. First, we split the whole data set into two nonoverlapping subsets, one for training (transduction) and one for testing (induction). In the transduction phase, we simply perform *LNP* on the training set with the number of randomly labeled points varying from 4 to 50, and the resultant label matrix is used for induction on the test set using (24). Fig. 8b illustrates the induction results. We fix the size of the training set to be 1,000 and 2,000, and the recognition accuracies on these two data sets are plotted by the blue and green lines. For comparison, the results achieved by the standard *LNP* (that is, using the whole data set for transduction) are also plotted in this figure. It is clearly observed that we can still

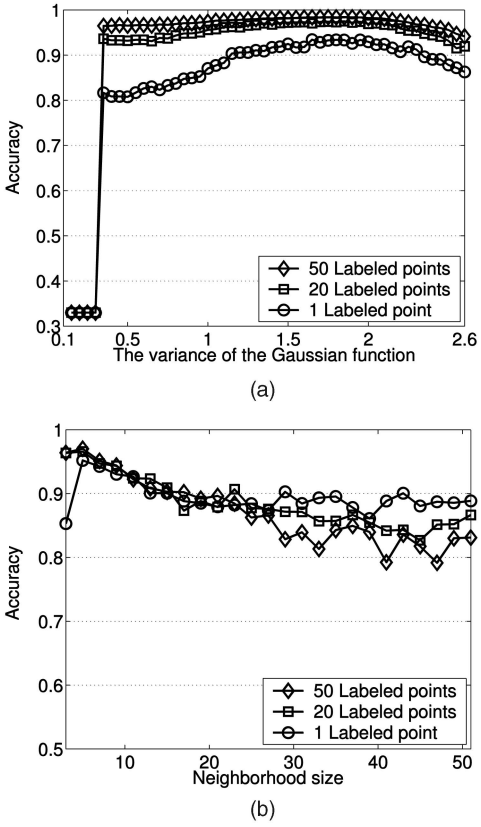


Fig. 9. Parameter stability testing results. In both figures, the ordinate represents the average recognition accuracy of 50 independent runs, and the size of the randomly labeled points is set to 1, 20, and 50, respectively. (a) The results achieved by Zhou et al.'s *consistency* method, where the abscissa is the variance of the RBF kernel. (b) The results achieved by our *LNP* method, where the abscissa represents the number of nearest neighbors.

get a high recognition accuracy using our induction method.

In the third part of this experiment, we studied the stability of our method with respect to the hyperparameters in *LNP*, including the neighborhood size k and the regularization parameter μ (note that $\alpha = \frac{1}{1+\mu}$):

- The result for the neighborhood size k is shown in Fig. 9, where we also tested the stability of the variance σ of the RBF kernel in Zhou et al.'s *consistency method* for comparison. It seems that for this problem, the consistency method is stable as long as σ is not too small, and our *LNP* method is also stable if k is not too large (since a large k will cause a confusion of different digit image manifolds). Doubtless, k is easier to tune since it is selected from only positive integers.
- The result for the regularization parameter α is shown in Fig. 10. From the analysis in Theorem 2, we know that the role of μ is to trade off the prediction loss and smoothness. The figure also verifies that we should consider both the prediction loss and smoothness in classification, since the performance of *LNP* could be poor when μ becomes either too large or too small.

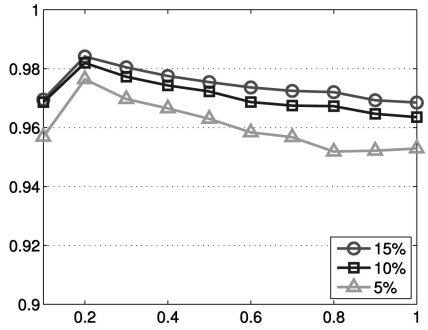


Fig. 10. The sensitivity of *LNP* to the choice of μ on the *USPS* data set, where the x-axis represents the value of μ, and the y-axis represents the average classification accuracies of *LNP*. In the figure, different lines represent the results with different percentage of randomly labeled points, and the values on the line represent the average classification accuracies over 50 independent runs.

5.3 Text Classification

In this experiment, we addressed the task of text classification using the 20-Newsgroups data set.[8] The topic *rec* containing *autos*, *motorcycles*, *baseball*, and *hockey* was selected from the version 20news-18828. The articles were processed by the Rainbow software package with the following options:

1. passing all words through the Porter stemmer before counting them,
2. tossing out any token that is on the stop list of the *SMART* system,
3. skipping any headers, and
4. ignoring words that occur in five or fewer documents.

No further preprocessing was done. Removing the empty documents, we obtained 3,970 document vectors in an 8,014-dimensional space. Finally, the documents were normalized into a *TFIDF* representation.

We use the inner product distance to find the k nearest neighbors when constructing the neighborhood graph in *LNP*, that is.

$$d(\mathbf{x}_i, \mathbf{x}_j) = 1 - \frac{\mathbf{x}_i^T \mathbf{x}_j}{\|\mathbf{x}_i\| \|\mathbf{x}_j\|}, \tag{26}$$

where $\mathbf{x}_i$ and $\mathbf{x}_j$ are document vectors. Moreover, the value of k is set to 10 manually. When using preprocessing, the threshold of the *confusion rate* is chosen such that 2 percent of the data are thrown. For Zhou et al.'s *consistency*, Zhu et al.'s *Gaussian fields*, and *Belkin* et al.'s *LapRLS* methods, the affinity matrices were all computed by

$$(\mathbf{W})_{ij} = exp\left(-\frac{1}{2\sigma^2}\left(1 - \frac{\mathbf{x}_i^T \mathbf{x}_j}{\|\mathbf{x}_i\| \|\mathbf{x}_j\|}\right)\right). \tag{27}$$

The *SVM* and *Nearest Neighbor* classifiers also served as the baseline algorithms, the variance of the *Gaussian kernel* was set by five-fold cross validation, and the other hyperparameters in *LapRLS* were set as in [10]. The *accuracy* versus *number of labeled points* plot is shown in Fig. 11a, where the accuracy values are averaged over 50 independent trials. In

8. http://people.csail.mit.edu/jrennie/20Newsgroups/.

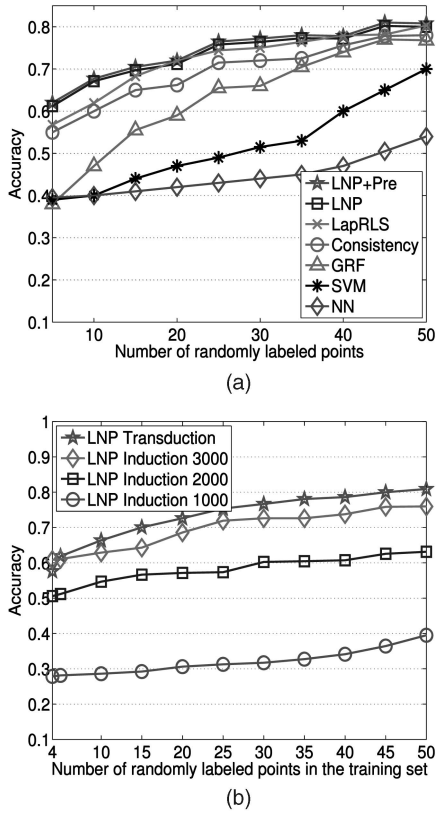


Fig. 11. Classification accuracies on the *20-Newsgroup* data. A subset of topic *rec* was adopted. (a) The classification accuracies for different algorithms. (b) The classification accuracies with 1,000, 2,000, and 3,000 points selected for training (transduction) and the remaining data points for testing (induction). In both figures, the abscissa represents the number of randomly labeled data in the training set (we guarantee that there is at least one labeled point in each class), and the ordinate is the total classification accuracy value averaged over 50 independent runs.

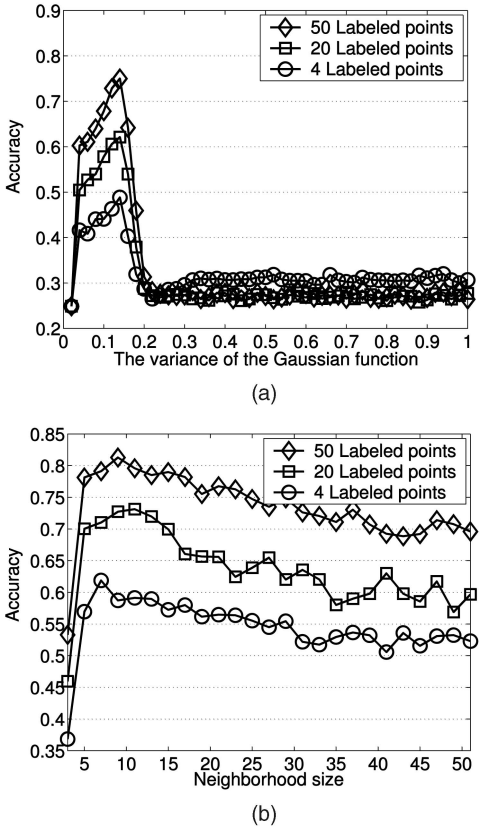


Fig. 12. Parameter stability testing results. In both figures, the ordinate represents the classification recognition accuracy of 50 independent runs, and the size of the randomly labeled points is set to 1, 20, and 50, respectively. (a) The results achieved by Zhou et al.'s *consistency* method, where the abscissa is the width of the *RBF* kernel. (b) The results achieved by our *LNP* method, where the abscissa represents the number of nearest neighbors.

this figure, we can clearly see the effectiveness of our approach.

Fig. 11b illustrates the induction results of *LNP* on the 20-Newsgroup data. We fixed the size of the training set to be 1,000, 2,000, and 3,000, and the remaining data were used for induction. Clearly, 1,000 data points are not enough for describing the structure of the whole data set, as the classification accuracies are dramatically poor. Moreover, 3,000 points are sufficient for discovering this structure in that the classification accuracies can approximate the accuracies achieved by the standard *LNP*, which uses the whole data set for classification.

We also test the parameter stability in Zhou et al.'s consistency and our *LNP* methods. The experimental results are shown in Fig. 12, where the meanings of the axes are the same as in Fig. 9. We may find that the consistency method is very unstable in this experiment, since it can only achieve a high classification accuracy when σ falls into a very small range (between 0.1 and 0.2). In contrast, our *LNP* method is much more stable, and it can hold a high classification accuracy as long as k is not too small. In Fig. 13, we show the sensitivity of *LNP* to the choice of μ, from which we can see that there is a sudden increase when μ changes from 0.1 to 0.2, which indicates that incorporating the smoothness term can greatly improve the effectiveness of *LNP*.

5.4 Web Page Categorization

We also applied our method to Web page categorization using the *WebKB* data set.[9] The categories *student*, *faculty*, *course*, and *project* were selected for experimental evaluations. All of these data were preprocessed by the Rainbow software package with the following options:

1. skip the *MIME* headers,
2. skip everything inside "$<$" and "$>$,"
3. tossing out any token that is on the stop list of the *SMART* system, and
4. ignoring words that occur in five or fewer documents.

No further preprocessing was done. Removing the empty Web pages, we obtained 4,199 Web page vectors of dimension 5,000. Finally, the documents were normalized into a *TFIDF* representation.

We also use (26) to find the k nearest neighbors when constructing the neighborhood graph in *LNP*, with $k = 10$, which is set manually. When using preprocessing, the threshold of the *confusion rate* is chosen such that 2 percent of the data are thrown. The hyperparameters in Zhou et al.'s *consistency*, Zhu et al.'s *Gaussian fields*, and Belkin et al.'s *LapRLS* methods were all set in the same way as in the previous experiment on the 20-Newsgroup data set. The

9. http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/webkb-data.gtar.gz.

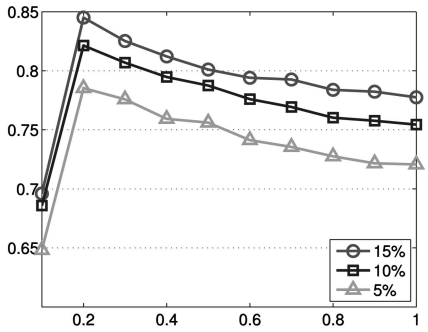
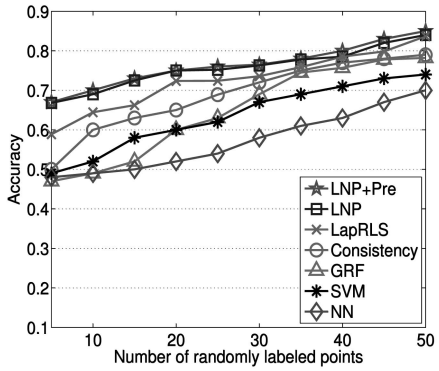
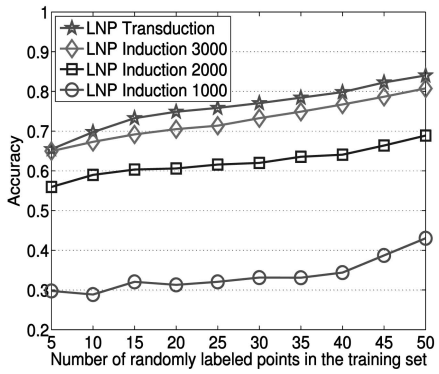


Fig. 13. The sensitivity of *LNP* to the choice of μ on the 20-Newsgroup data set, where the x-axis represents the value of μ, and the y-axis represents the average classification accuracies of *LNP*. In the figure, different lines represent the results with different percentage of randomly labeled points, and the values on the line represent the average classification accuracies over 50 independent runs.

SVM and *Nearest Neighbor* classifiers served as the baseline algorithms. All the hyperparameters in these methods were set by five-fold cross validation. The *accuracy* versus *number of labeled points* plot is shown in Fig. 14a, where the accuracy values are averaged over 50 independent trials. In this figure, we can clearly see the effectiveness of our method.

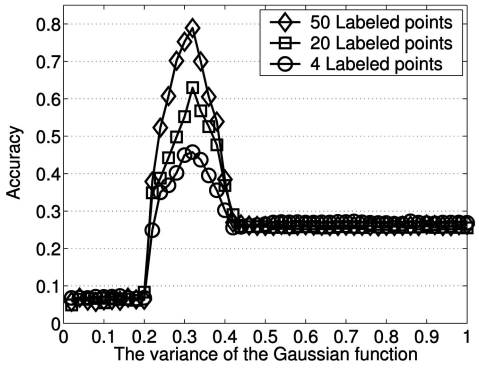


(a)

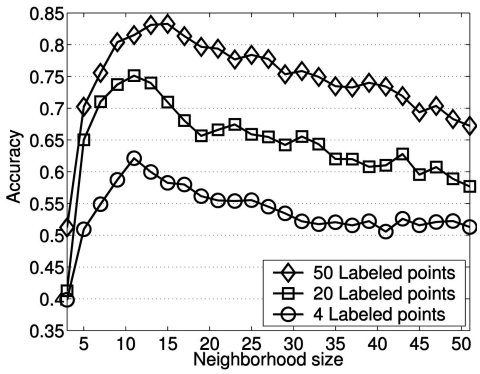


(b)

Fig. 14. Classification accuracies on the *WebKB* data set. (a) The classification accuracies for different algorithms. (b) The classification accuracies with 1,000, 2,000, and 3,000 points selected for training (transduction) and the remaining data points for testing (induction). In both figures, the abscissa represents the number of randomly labeled data in the training set (we guarantee that there is at least one labeled point in each class), and the ordinate is the total recognition accuracy value averaged over 50 independent runs.



(a)



(b)

Fig. 15. Parameter stability testing results on the *WebKB* data set. In both figures, the ordinate represents the average recognition accuracy of 50 independent runs, and the size of the randomly labeled points is set to 1, 20, and 50, respectively. (a) The results achieved by Zhou et al.'s *consistency* method, where the abscissa is the width of the *RBF* kernel. (b) The results achieved by our *LNP* method, where the abscissa represents the number of nearest neighbors.

The induction results are shown in Fig. 14b, where we also use a subset containing 1,000, 2,000, and 3,000 points for training, and the rest are for testing, from which we can draw the same conclusion as in the experiments on the 20-Newsgroup data set, that is, when provided sufficient data, our induction method can produce results sufficiently approximating the transduction results.

We also test the parameter stability in Zhou et al.'s consistency and our *LNP* methods, and the results are shown in Fig. 15. We can also find that the *LNP* method is very stable, and the *consistency* method is not so stable in this experiment. The sensitivity testing result of *LNP* with respect to μ is shown in Fig. 16, from which we can draw similar conclusions as the former experiments.

6 Conclusions and Future Works

In this paper, we propose a novel semisupervised learning algorithm called *LNP*. It can discover the structure of the whole data set through synthesizing the linear neighborhood around each data object. We also analyzed theoretically that the resulting data labels can be sufficiently smooth with respect to the data structure. A bridge point elimination procedure is also proposed to make *LNP* more robust. Finally, we provide many experiments to show the effectiveness of our method, from which we also find that

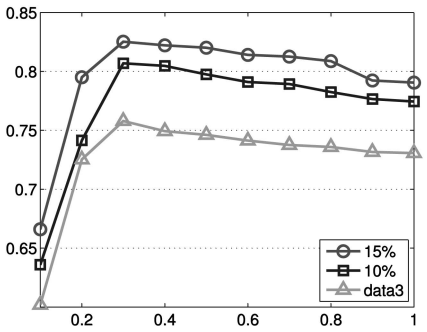


Fig. 16. The sensitivity of *LNP* to the choice of μ on the *WebKB* data set, where the x-axis represents the value of μ, and the y-axis represents the average classification accuracies of *LNP*. In the figure, different lines represent the results with different percentage of randomly labeled points, and the values on the line represent the average classification accuracies over 50 independent runs.

LNP also have high parameter stability. In our future work, we will focus on the theoretical analysis and accelerating issues of our *LNP* algorithm.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their constructive comments. The work was supported by the China National Science Foundation Project No. 60675009.

REFERENCES

[1] M.-F. Balcan, A. Blum, P.P. Choi, J. Lafferty, B. Pantano, M.R. Rwebangira, and X. Zhu, "Person Identification in Webcam Images: An Application of Semi-Supervised Learning," *Proc. ICML Workshop Learning with Partially Classified Training Data,* 2005.

[2] M. Belkin and P. Niyogi, "Laplacian Eigenmaps for Dimensionality Reduction and Data Representation," *Neural Computation,* vol. 15, no. 6, pp. 1373-1396, 2003.

[3] M. Belkin, I. Matveeva, and P. Niyogi, "Regularization and Semi-Supervised Learning on Large Graphs," *Proc. 17th Ann. Conf. Learning Theory (COLT '04),* pp. 624-638, 2004.

[4] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold Regularization: A Geometric Framework for Learning from Labeled and Unlabeled Examples," *J. Machine Learning Research,* vol. 7, pp. 2399-2434, Nov. 2006.

[5] Y. Bengio, M. Monperrus, and H. Larochelle, "Nonlocal Estimation of Manifold Structure," *Neural Computation,* vol. 18, no. 10, pp. 2509-2528, 2006.

[6] A. Blum and T. Mitchell, "Combining Labeled and Unlabeled Data with Co-Training," *Proc. 11th Ann. Conf. Computational Learning Theory (COLT '98),* pp. 92-100, 1998.

[7] A. Blum and S. Chawla, "Learning from Labeled and Unlabeled Data Using Graph Mincuts," *Proc. 18th Int'l Conf. Machine Learning (ICML '01),* pp. 19-26, 2001.

[8] M.A. Carreira-Perpinan and R.S. Zemel, "Proximity Graphs for Clustering and Manifold Learning," *Advances in Neural Information Processing Systems 17,* L.K. Saul, Y. Weiss, and L. Bottou, eds., pp. 225-232, MIT Press, 2005.

[9] O. Chapelle, J. Weston, and B. Schölkopf, "Cluster Kernels for Semi-Supervised Learning," *Advances in Neural Information Processing Systems 15,* S. Becker, S. Thrun, and K. Obermayer, eds., pp. 601-608, MIT Press, 2003.

[10] O. Chapelle, B. Schölkopf, and A. Zien, *Semi-Supervised Learning,* p. 371. MIT Press, 2006.

[11] O. Delalleu, Y. Bengio, and N. Le Roux, "Non-Parametric Function Induction in Semi-Supervised Learning," *Proc. 10th Int'l Workshop Artificial Intelligence and Statistics (AISTAT '05),* pp. 96-103, 2005.

[12] A.P. Dempster, N.M. Laird, and D.B. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," *J. Royal Statistical Soc., Series B,* vol. 39, no. 1, pp. 1-38, 1977.

[13] F.R.K. Chung, "Spectral Graph Theory," *CBMS Regional Conf. Series in Mathematics,* vol. 92, published for the Conf. Board of the Mathematical Sciences, Washington, DC. 1997.

[14] G.H. Golub and C.F Van Loan, *Matrix Computation,* second ed., 1989.

[15] A.K. Jain and R.C Dubes, *Algorithms for Clustering Data,* Prentice Hall Advanced Reference Series. Prentice Hall, 1988.

[16] T. Joachims, "Transductive Inference for Text Classification Using Support Vector Machines," *Proc. 16th Int'l Conf. Machine Learning (ICML '99),* pp. 200-209, 1999.

[17] T. Joachims, "Transductive Learning via Spectral Graph Partitioning," *Proc. 20th Int'l Conf. Machine Learning (ICML '03),* pp. 290-297, 2003.

[18] N. Kambhatla and T.K. Leen, "Dimension Reduction by Local Principal Component Analysis," *Neural Computation,* vol. 9, no. 7, pp. 1493-1516, 1997.

[19] A. Kapoor, Y. Qi, H. Ahn, and R.W. Picard, "Hyperparameter and Kernel Learning for Graph Based Semi-Supervised Classification," *Advances in Neural Information Processing Systems,* 2005.

[20] N.D. Lawrence and M.I. Jordan, "Semi-Supervised Learning via Gaussian Processes," *Advances in Neural Information Processing Systems 17,* L.K. Saul, Y. Weiss, and L. Bottou, eds., MIT Press, 2005.

[21] D.J. Miller and U.S. Uyar, "A Mixture of Experts Classifier with Learning Based on Both Labelled and Unlabelled Data," *Advances in Neural Information Processing Systems 9,* M. Mozer, M.I. Jordan, and T.Petsche, eds., pp. 571-577, MIT Press, 1997.

[22] K. Nigam, A.K. McCallum, S. Thrun, and T. Mitchell, "Text Classification from Labeled and Unlabeled Documents Using EM," *Machine Learning,* vol. 39, no. 2-3, pp. 103-134, 2000.

[23] J.R. Quinlan, "Introduction to Decision Trees," *Machine Learning,* vol. 1, no. 1, pp. 81-106, 1986.

[24] S.T. Roweis and L.K. Saul, "Nonlinear Dimensionality Reduction by Locally Linear Embedding," *Science,* vol. 290, pp. 2323-2326, 2000.

[25] L.K. Saul, K.Q. Weinberger, J.H. Ham, F. Sha, and D.D. Lee, "Spectral Methods for Dimensionality Reduction," *Semisupervised Learning,* O. Chapelle, B. Schölkopf, and A. Zien, eds. MIT Press, 2006.

[26] B. Schölkopf and A.J. Smola, *Learning with Kernels.* MIT Press, 2002.

[27] B. Shahshahani and D. Landgrebe, "The Effect of Unlabeled Samples in Reducing the Small Sample Size Problem and Mitigating the Hughes Phenomenon," *IEEE Trans. Geoscience and Remote Sensing,* vol. 32, no. 5, pp. 1087-1095, 1994.

[28] M. Szummer and T. Jaakkola, "Partially Labeled Classification with Markov Random Walks," *Advances in Neural Information Processing Systems 14,* T.G. Dietterich, S. Becker, and Z. Ghahramani, eds., pp. 945-952, 2002.

[29] J.B. Tenenbaum, V. Silva, and J.C. Langford, "A Global Geometric Framework for Nonlinear Dimensionality Reduction," *Science,* vol. 290, pp. 2319-2323, 2000.

[30] V.N. Vapnik, *The Nature of Statistical Learning Theory.* Springer, 1995.

[31] F. Wang and C. Zhang, "Label Propagation through Linear Neighborhoods," *Proc. 23rd Int'l Conf. Machine Learning (ICML '06),* pp. 985-992, 2006.

[32] D. Zhou, O. Bousquet, T.N. Lal, J. Weston, and B. Schölkopf, "Learning with Local and Global Consistency," *Advances in Neural Information Processing Systems 16,* S. Thrun, L. Saul, and B. Schölkopf, eds., pp. 321-328, 2004.

[33] D. Zhou and B. Schölkopf, "Learning from Labeled and Unlabeled Data Using Random Walks," *Proc. 26th Pattern Recognition Symp. (DAGM '04),* 2004.

[34] D. Zhou, B. Schölkopf, and T. Hofmann, "Semi-Supervised Learning on Directed Graphs," *Advances in Neural Information Processing Systems 17,* L.K., Saul, Y. Weiss, and L. Bottou, eds., pp. 1633-1640, MIT Press, 2005.

[35] X. Zhu, Z. Ghahramani, and J. Lafferty, "Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions," *Proc. 20th Int'l Conf. Machine Learning (ICML '03),* 2003.

[36] X. Zhu and Z. Ghahramani, "Learning from Labeled and Unlabeled Data with Label Propagation," Technical Report CMU-CALD-02-107, Carnegie Mellon Univ., 2002.

[37] X. Zhu and Z. Ghahramani, "Towards Semi-Supervised Classification with Markov Random Fields," Technical Report CMU-CALD-02-106, Carnegie Mellon Univ., 2002.

[38] X. Zhu, J. Lafferty, and Z. Ghahramani, "Semi-Supervised Learning: From Gaussian Fields to Gaussian Processes," Technical Report CMU-CS-03-175, Carnegie Mellon Univ., 2003.
[39] X. Zhu, "Semi-Supervised Learning Literature Survey," Computer Sciences Technical Report 1530, Univ. of Wisconsin, Madison, 2006.

Fei Wang is a PhD student at Tsinghua University, Beijing, China. His main research interests include machine learning, data mining, and pattern recognition. He is a student member of the IEEE.

Changshui Zhang is a professor in the Department of Automation, Tsinghua University, Beijing, China. He is an associate editor of *Pattern Recognition Journal*. He is a member of the IEEE.