Label Propagation Through Linear Neighborhoods

Fei Wang FEIWANG03@MAILS.TSINGHUA.EDU.CN
Changshui Zhang ZCS@MAIL.TSINGHUA.EDU.CN
State Key Laboratory of Intelligent Technology and Systems, Department of Automation, Tsinghua University, Beijing 100084, P.R.China

Abstract

A novel semi-supervised 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 dataset using these linear neighborhoods with sufficient smoothness. 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.

1. Introduction

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. However, 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 (*e.g.* by crawling the web), but only a small part of them are classified by hand.

Consequently, semi-supervised learning methods, which aims to learn from partially labeled data, are proposed (for a detailed literature survey see (Zhu, 2005)). In general, these methods can be categorized into two classes: *transductive learning*(*e.g.* (Joachims, 1999)) and *inductive learning*(*e.g.* (Belkin *et al*, 2005)). The goal of transductive learning is only to estimate the labels of the given unlabeled data, whereas inductive learning tries to induce a decision function which has a low error rate on the whole sample space.

Appearing in *Proceedings of the 23rd International Conference on Machine Learning*, Pittsburgh, PA, 2006.

The key to semi-supervised learning problems is the prior consistency (Zhou *et al*, 2004a) (also called *cluster assumption* (Chapelle *et al*, 2003)): (1) nearby points are likely to have the same label; (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, while the second one is global. The cluster assumption implies us to consider both local and global information contained in the dataset during learning.

It is straightforward to associate cluster assumption with the nonlinear dimensionality reduction methods developed in recent years (Roweis & Saul, 2000; Tenenbaum, 2000; Belkin & Niyogi, 2002), since the central idea of these methods is to construct a low dimensional global coordinate system for the dataset in which the local structure of the data is preserved. *Laplacian Eigenmaps* (Belkin & Niyogi, 2002) is a typical method of such kind. It describes a dataset as a graph in which the vertices representing the data and the edges representing the pairwise relationships, and aims at finding a low dimensional coordinate system in which the data can preserve such graph structure.

Similarly, the graph-based semi-supervised learning methods, which have aroused considerable interests in recent years, model the whole dataset as a graph in the same way. However, although the graph is at the heart of these graph-based methods, its construction has not been studied extensively (Zhu, 2005). More concretely, most of these methods (Szummer & Jaakkola, 2002; Zhou *et al*, 2004a; Zhu *et al*, 2003) adopted a gaussian function to calculate the edge weights of the graph (*i.e.* the edge links data $\mathbf{x}_i$ and $\mathbf{x}_j$ is computed as $e_{ij} = exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/(2\sigma^2))$), but the variance σ of the Gaussian function (will affect the classification results significantly. This problem can be illustrated in Fig.1. Moreover, few of the graph-based methods provide an out-of-sample extension (Belkin *et al*, 2005).

To address the above issues, we propose a novel method called *Linear Neighborhood Propagation*

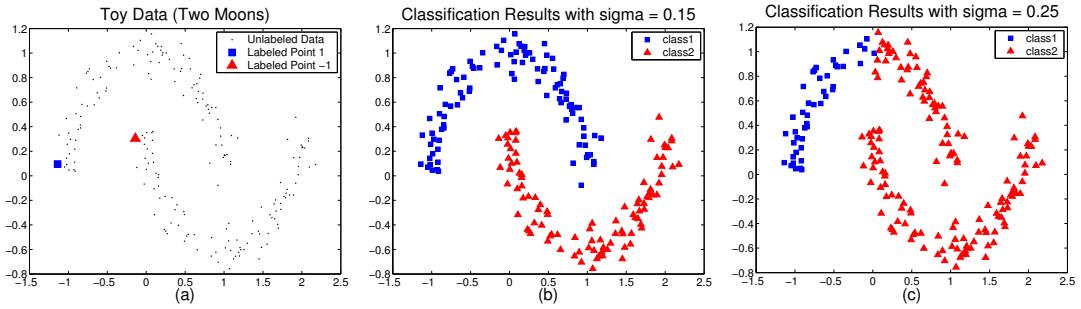

Figure 1. Classification results on the two-moon pattern using the method in (Zhou *et al*, 2004a), which is a powerful transductive approach operating on graph with the edge weights computed by a Gaussian function. (a) toy dataset with two labeled points; (b) classification results with $\sigma = 0.15$; (c) classification results with $\sigma = 0.25$. We can see that a small variation of σ will cause a dramatically different classification result.

(*LNP*) in this paper. The *LNP* algorithm first 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. Then 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 semi-supervised learning algorithms. We also give a easy method for extending *LNP* to out-of-sample data points. Finally the experiments on digits and text classification are presented to show the effectiveness of *LNP*.

It is worthwhile to highlight three aspects of the proposed approach here: (1) the storage requirement of *LNP* is smaller than traditional graph-based methods; (2) the edge weights of the graph constructed by *LNP* can be solved in closed form; (3) *LNP* can be easily extend to out-of-sample data points.

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

2. The Algorithm

We will formally present our *Linear Neighborhood Propagation* algorithm in this section. First let's 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 $\mathbf{x}_i \in \mathcal{X}$ $(i \leqslant l)$ are labeled as $L_i \in \mathcal{L}$ and the remaining points $\mathbf{x}_u \in \mathcal{X}$ $(l+1 \leqslant u \leqslant n)$ are unlabeled. The goal of *LNP* is to predict the labels of $\mathbf{x}_u$(we will show in section 3.3 that it is easy for *LNP* to extend to out-of-sample data), which can be achieved by two steps:

Step 1: Construct the graph $\mathcal{G} =< \mathcal{V}, \mathcal{E} >$, where $\mathcal{V} = \mathcal{X}$ is the vertex set, $\mathcal{E}$ is the edge set associated with each edge e_{ij} representing the relationship between data $\mathbf{x}_i$ and $\mathbf{x}_j$.

Conventionally (Belkin & Niyogi, 2002; Zhou *et al*, 2004a; Szummer & Jaakkola, 2002) , the edges in $\mathcal{E}$ is calculated by

$$e_{ij} = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}} \tag{1}$$

However, as pointed out by (Zhou *et al*, 2004a), there is no reliable approach for model selection if only very few labeled points are available (*i.e.* it is hard to determine an optimal σ in Eq.(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 $\mathcal{G}$.

Instead of considering pairwise relationships as Eq.(1) in traditional graph-based methods, we propose to use the neighborhood information of each point to construct $\mathcal{G}$. For computational convenience, we assume that all these neighborhoods are linear, *i.e.* each data point can be optimally reconstructed using a linear combination of its neighbors (Roweis & Saul, 2000). Hence our objective is to minimize

$$\varepsilon = \sum_i \left\| \mathbf{x}_i - \sum_{j:\mathbf{x}_j \in \mathcal{N}(\mathbf{x}_i)} w_{ij}\mathbf{x}_j \right\|^2 \tag{2}$$

where $\mathcal{N}(\mathbf{x}_i)$ represents the neighborhood of $\mathbf{x}_i$, and w_{ij} is the contribution of $\mathbf{x}_j$ to $\mathbf{x}_i$. We further constrain $\sum_{j \in \mathcal{N}(\mathbf{x}_i)} w_{ij} = 1$, $w_{ij} \geqslant 0$. Obviously, the

more similar $\mathbf{x}_j$ to $\mathbf{x}_i$, the larger w_{ij} will be (as an extreme case, when $\mathbf{x}_i = \mathbf{x}_k \in \mathcal{N}(\mathbf{x}_i)$, then $w_{ik} = 1$, $w_{ij} = 0$, $j \neq k$, $\mathbf{x}_j \in \mathcal{N}(\mathbf{x}_i)$ is the optimal solution). Thus w_{ij} can be used to measure how similar $\mathbf{x}_j$ to $\mathbf{x}_i$. One issue should be addressed here is that usually $w_{ij} \neq w_{ji}$. It can be easily inferred that

$$\begin{aligned}\varepsilon_i &= \left\|\mathbf{x}_i - \sum\nolimits_{j:\mathbf{x}_j\in\mathcal{N}(\mathbf{x}_i)} w_{ij}\mathbf{x}_j\right\|^2 \\ &= \left\|\sum\nolimits_{j:\mathbf{x}_j\in\mathcal{N}(\mathbf{x}_i)} w_{ij}\left(\mathbf{x}_i - \mathbf{x}_j\right)\right\|^2 \\ &= \sum\nolimits_{j,k:\mathbf{x}_j,\mathbf{x}_k\in\mathcal{N}(\mathbf{x}_i)} w_{ij}w_{ik}\left(\mathbf{x}_i - \mathbf{x}_j\right)^T\left(\mathbf{x}_i - \mathbf{x}_j\right) \\ &= \sum\nolimits_{j,k:\mathbf{x}_j,\mathbf{x}_k\in\mathcal{N}(\mathbf{x}_i)} w_{ij}G^i_{jk}w_{ik} \qquad (3)\end{aligned}$$

where G^i_{jk} represents the (j,k)-th entry of the local Gram matrix $(\mathbf{G}^i)_{j,k} = \left(\mathbf{x}_i - \mathbf{x}_j\right)^T\left(\mathbf{x}_i - \mathbf{x}_j\right)$ at point $\mathbf{x}_i$, where $(\cdot)_{j,k}$ represents the (j,k)-th entry of a matrix. Thus the reconstruction weights of each data object can be solved by the following n standard quadratic programming problems

$$\begin{aligned} min_{w_{ij}} \quad & \sum\nolimits_{j,k:\mathbf{x}_j,\mathbf{x}_k\in\mathcal{N}(\mathbf{x}_i)} w_{ij}G^i_{jk}w_{ik} \\ s.t. \quad & \sum_j w_{ij} = 1,\ w_{ij} \geqslant 0 \end{aligned} \qquad (4)$$

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

$$(\mathbf{W})_{i,j} = w_{ij} \qquad (5)$$

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

Step 2: Propagate the labels of the labeled data to the remaining unlabeled data $\mathbf{x}_u \in \mathcal{X}$ $(l+1 \leqslant u \leqslant n)$ using the graph constructed by the first step. We will use an iterative procedure to achieve this goal.

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 $f_u = f(\mathbf{x}_u)$.

In each iteration, we let each data object "absorbs" 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 $t+1$ becomes

$$f_i^{t+1} = \alpha\sum\nolimits_{j:\mathbf{x}_j\in\mathcal{N}(\mathbf{x}_i)} w_{ij}f_j^t + (1-\alpha)y_i \qquad (6)$$

where $0 < \alpha < 1$ is the fraction of label information that $\mathbf{x}_i$ receives from its neighbors. Let $\mathbf{y} = (y_1, y_2, \cdots y_n)^T$ with $y_i = L_i$ $(i \leqslant l)$, $y_u = 0$ $(l+1 \leqslant u \leqslant n)$. $\mathbf{f}^t = (f_1^t, f_2^t, \cdots, f_n^t)^T$ is the prediction label vector at iteration t and $\mathbf{f}^0 = \mathbf{y}$. Then we can rewrite our iteration equation as

$$\mathbf{f}^{t+1} = \alpha\mathbf{W}\mathbf{f}^t + (1-\alpha)\mathbf{y} \qquad (7)$$

We will use Eq.(7) to update the labels of each data object until convergence, here "convergence" means the predicted labels of the data will not change in several successive iterations (the convergence analysis of our algorithm will be presented in 3.1).

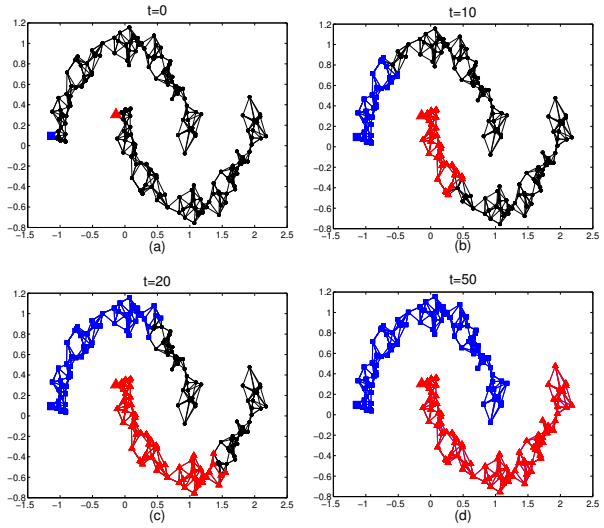

Figure 2. Transduction results on the two-moon pattern using *LNP*. The number of nearest neighbors $k = 4$. $\alpha = 0.99$. The blue squares represent class 1 (whose labels are positive), and the red circle represent class 2 (whose labels are negative). The black dots represent data points whose labels are zero. (a) the initial state, *i.e.* t=0,only two points are labeled; (b) t=15; (c) $t = 20$; (d) $t = 50$.

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

It is easy to extend *LNP* to multi-class 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 non-negative real-value entries. Any matrix $\mathbf{F} = \left[\mathbf{F}_1^T, \mathbf{F}_2^T, \cdots, \mathbf{F}_n^T\right]^T \in \mathcal{M}$ corresponds to a specific classification on $\mathcal{X}$ which labels $\mathbf{x}_i$ as $y_i = argmax_{j\leqslant c}\mathbf{F}_{ij}$. Thus $\mathbf{F}$ can also be viewed as a function which assign labels for each data point. Initially, we set $\mathbf{F}_0 = \mathbf{Y}$, where $\mathbf{Y}_{ij} = 1$ if $\mathbf{x}_i$ is labeled as j, and $\mathbf{Y}_{ij} = 0$ otherwise, and for unlabeled points, $\mathbf{Y}_{uj} = 0$ $(1 \leqslant j \leqslant c)$. The main procedure of *LNP* can be summarized in table 1.

Table 1. 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.(4) 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 $y_i = argmax_{j \leqslant c}\mathbf{F}^\star_{ij}$

3. Analysis and Extension of LNP

In this section we will analyze *LNP* theoretically. First we will analyze the convergence property of iterative *LNP* shown in table 1 and propose a one-shot *LNP*. Second we will derive *LNP* from a regularization framework, thus provide a theoretical guarantee of the feasibility of *LNP*. At last we will give a simple way to extend *LNP* to out-of-sample data.

3.1. Convergence Analysis of LNP

In this subsection we will prove that the sequence $\{\mathbf{f}_t\}$ will converge to $\mathbf{f}^\star = (1-\alpha)(1-\alpha\mathbf{W})^{-1}\mathbf{y}$.

By Eq.(7) and the initial condition that $\mathbf{f}_0 = \mathbf{y}$, we have

$$\mathbf{f}^t = (\alpha\mathbf{W})^{t-1}\mathbf{y} + (1-\alpha)\sum\nolimits_{i=0}^{t-1}(\alpha\mathbf{W})^i\mathbf{y} \qquad (8)$$

Since $w_{ij} \geqslant 0$ and $\sum_j w_{ij} = 1$, from the theorem of *Perron-Frobenius* (Golub & Van Loan, 1989), we know that the spectral radius of $\mathbf{W}$, or $\rho(\mathbf{W}) \leqslant 1$. In addition, $0 < \alpha < 1$, thus

$$\lim_{t\to\infty}(\alpha\mathbf{W})^{t-1} = 0$$
$$\lim_{t\to\infty}\sum\nolimits_{i=0}^{t-1}(\alpha\mathbf{W})^i = (\mathbf{I}-\alpha\mathbf{W})^{-1}$$

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

$$\mathbf{f}' = \lim_{t\to\infty}\mathbf{f}^t = (1-\alpha)(\mathbf{I}-\alpha\mathbf{W})^{-1}\mathbf{y} \qquad (9)$$

Since in classification, we only use the sign of f_i to determine the label of $\mathbf{x}_i$, the constant term $1-\alpha \geqslant 0$ 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*, *i.e.*, we can predict the labels of all the data objects in one step.

Similarly, for the multi-class case, we may simply replace $\mathbf{y}$ by $\mathbf{Y}$ in Eq.(9), which results in

$$\mathbf{F}' = (1-\alpha)(\mathbf{I}-\alpha\mathbf{W})^{-1}\mathbf{Y} \qquad (10)$$

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

3.2. Regularization Framework of LNP

We can also derive our *LNP* algorithm from a regularization framework. Define the loss function as:

$$\mathcal{Q}(\mathbf{f}) = \sum_{i=1}^{n}\sum_{j:\mathbf{x}_j\in\mathcal{N}(\mathbf{x}_i)} w_{ij}(f_i - f_j)^2 + \gamma\sum_{i=1}^{n}(f_i - y_i)^2 \qquad (11)$$

Intuitively, the first term of $\mathcal{Q}(f)$ describes the total variation of the data labels with respect to the neighborhood structures, and we will call it *smoothness term*. The second term measures how well the predicted labels fit the original labels, thus we call it *fit term*. We can easily compute the derivative of $\mathcal{Q}(f)$ with respect to $\mathbf{f} = (f_1, f_2, \cdots, f_n)^T$ as

$$\frac{\partial\mathcal{Q}(f)}{\partial\mathbf{f}} = \left[(\mathbf{I}-\mathbf{W}) + (\mathbf{I}-\mathbf{W})^T\right]\mathbf{f} + 2\gamma(\mathbf{y}-\mathbf{f}) \qquad (12)$$

Since $\mathbf{f}$ can be thought of as a function defined on a graph that 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 2, the graph *LNP* constructs is just a pasted graph. In such a viewpoint, the matrix $\mathbf{I}-\mathbf{W}$ can be regarded as the *graph Laplacian* of this "pasted" graph intuitively. And Belkin (2002) told us that under certain conditions, we have

$$(\mathbf{I}-\mathbf{W})\mathbf{f} \approx \mathcal{L}f \qquad (13)$$

where $\mathcal{L}$ is the *Laplacian-Beltrami* operator defined on the data manifold, and f is the function defined on this manifold. As we know that a graph is the discretized form of a manifold, thus $\mathbf{f}$ can also be regarded as the discretized form of f, with its values equivalent to the values of f at the nodes of the graph. Therefore

$$\left[(\mathbf{I}-\mathbf{W}) + (\mathbf{I}-\mathbf{W})^T\right]\mathbf{f} \approx 2\mathcal{L}f \approx 2\left[(\mathbf{I}-\mathbf{W})\right]\mathbf{f} \qquad (14)$$

Then we can easily get the approximating solution of minimizing $\mathcal{Q}(f)$ by set Eq.(12) to zero

$$\mathbf{f} = (1-\alpha)(\mathbf{I}-\alpha\mathbf{W})^{-1}\mathbf{y} \qquad (15)$$

where $\alpha = \frac{1}{1+\mu}$. It can be easily observed the equivalence between Eq.(15) and Eq.(9). That is to say, *LNP* can also be derived from a regularization framework.

3.3. Induction for Out-of-Sample Data

In section 2 we have introduced the main 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.

According to Delalleu *et al.* (2005), we need to do two things for generalizing *LNP* to out-of-sample data: (1) use the same type of smoothness criterion as in Eq.(11) for a new testing point $\mathbf{y}$; (2) the inclusion of $\mathbf{y}$ will not affect the original $\mathcal{Q}(f)$ value of the training dataset.

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

$$\mathcal{Q}(f(\mathbf{y})) = \sum_{i:\mathbf{x}_i \in \mathcal{X}, \mathbf{x}_i \in \mathcal{N}(\mathbf{y})} w(\mathbf{y}, \mathbf{x}_i)(f(\mathbf{y}) - f_i)^2 \quad (16)$$

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

$$f(\mathbf{y}) = \sum\nolimits_{i:\mathbf{x}_i \in \mathcal{X}, \mathbf{x}_i \in \mathcal{N}(\mathbf{y})} w(\mathbf{y}, \mathbf{x}_i) f_i \quad (17)$$

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 dataset, *i.e.*

$$f(\mathbf{y}) = \min_{f(\mathbf{y})} \left\| f(\mathbf{y}) - \sum_{i:\mathbf{x}_i \in \mathcal{X}, \mathbf{x}_i \in \mathcal{N}(\mathbf{y})} w(\mathbf{y}, \mathbf{x}_i) f_i \right\|^2 \quad (18)$$

To illustrate the effectiveness of this induction method, we also use the problem shown in Fig.1(a). 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, then Eq.(17) 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.3.

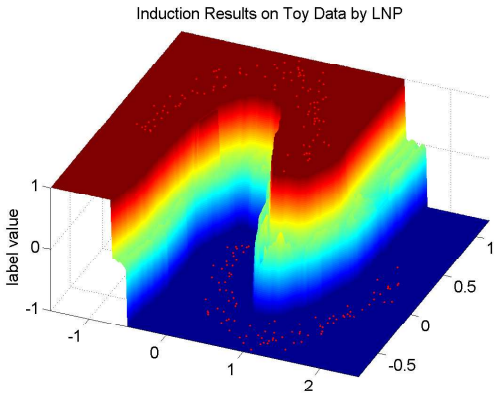

Figure 3. Induction results for all the data points in $\{(x, y)|x \in [-1.5, 2.5], y \in [-0.8, 1.2]\}$ using Eq.(17). The red dots represent the data points in the training dataset. The z-axis indicates the predicted label values associated with different colors. We can see that the induced decision boundary is intuitively satisfying since it is in accordance with the intrinsic structure of the training dataset.

4. Experiments

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

4.1. Toy Problems

In this subsection we shall use two toy examples to illustrate the effectiveness of our *LNP* method. The first problem is shown in Fig.4(a). The whole dataset contains 311 data points with two hidden clusters. The outside cluster, composed of 11 points, is shaped like parabola. It is generated from a half circle centered at $(0, 0)$ with radius 1. The inside cluster contains two subclusters which are two Gaussians centered at $[-0.4, 0.3]^T$ and $[0.4, 0.3]^T$ with the variance $\sigma^2 = 0.05$. Each of the two Gaussians has 150 data points.

Fig.4(a) shows us the original dataset with three labeled points. Fig.4(b) shows the transduction results using *LNP*. We can see that *LNP* can successfully discover the latent clusters contained in the dataset. Fig.4(c) is the transduction result when using the standard *graph Laplacian* as the smoothness regularizer (Belkin & Niyogi, 2002), *i.e.* we replace the first term of $\mathcal{Q}(f)$ (defined in Eq.(11)) by $\sum_{i,j} w_{ij}(f_i - f_j)^2$, where $w_{ij} = exp\{-\|\mathbf{x}_i - \mathbf{x}_j\|^2/(2\sigma^2)\}$. The scalar σ has been adjusted to achieve the best result. The result shown in Fig.4(d) is given by the method of Zhu *et al.* (2003). Obviously, both the methods respectively used in Fig.4(c) and Fig.4(d) fail to capture the outside parabola shaped cluster.

We think the reason why *LNP* can perform better in this toy problem is because its weight matrix $\mathbf{W}$ (defined in Eq.(5)) is normalized (*i.e.* $\sum_j w_{ij} = 1$). Since the density of this toy dataset varies substantially across different clusters, the classification results may be affected when using non-normalized smoothness regularizer (such as the ones used in Fig.4(c) and (d)). Zhou & Schölkopf (2004b) also addressed this problem and recommended to use the normalized smoothness regularizer.

In another toy problem we want to show the capability of *LNP* of automatically erasing the noise in labeled data. Shown in Fig.5(a) is the toy dataset of two circles with 8 labeled points. An ideal classifier should classify the inner circle as one class and the outer circle as the

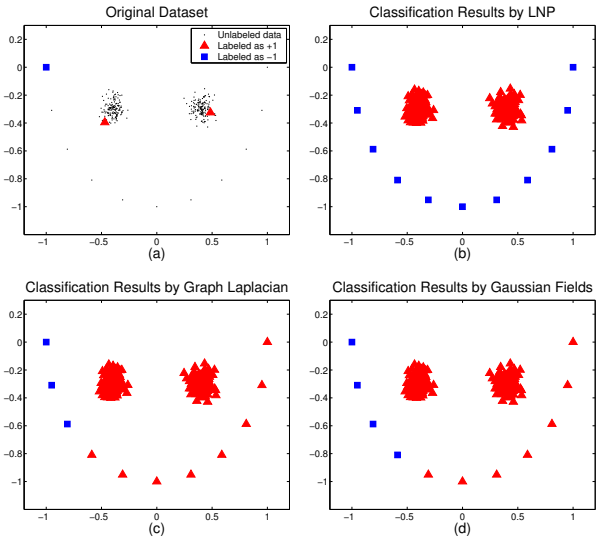

Figure 4. Transduction results on the doll toy data. The blue squares and red triangles represent two classes. (a) the original dataset, with only three data points labeled; (b) classification results with *LNP*, and we construct the neighborhood graph with the number of neighbors $k = 5$;(c) classification results using standard *graph Laplacian* as the smoothness regularizer with $\sigma = 0.15$; (d) classification results using *Gaussian fields* with $\sigma = 0.2$.

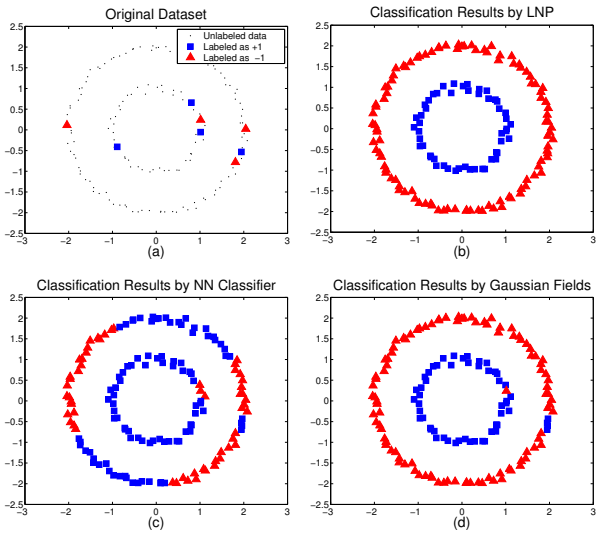

Figure 5. Transduction results on the two ringed toy data. The blue squares represent class 1, and the red triangles represent class 2. (a) the original dataset; (b) classification results with *LNP*, and we construct the neighborhood graph with the number of neighbors $k = 5$;(c) classification results using nearest neighbor classifier; (d) classification results using *Gaussian fields* with $\sigma = 0.28$.

other class. Thus there is a wrongly labeled point in each class, which can be thought of as noise.

Fig.5(b) shows us the classification result by *LNP*, where all the points are correctly classified. Fig.5(c) is the classification result by nearest neighbor (*NN*) classifier, and it is still difficult for *NN* to detect the noise data. The classification result by *Gaussian fields* is shown in Fig.5(d). We can see that the *Gaussian fields* method is also brittle here since the labels of the labeled points remain fixed during its process.

It is very possible that the situation described in Fig.5(a) occurs in real world problems. One may easily get a labeled dataset with some noise (possibly due to the tiredness or careless of the annotator). Since inspecting such a dataset needs extra human labor and time, how to design a robust classifier is of vital importance. Fortunately, *LNP* is such a classifier that can help to erase the noise in the dataset automatically.

4.2. Digit Recognition

In this case study, we will focus on the problem of classifying hand-written digits. The dataset we adopt is the USPS [1] handwritten 16x16 digits dataset. The images of digits 1, 2, 3 and 4 are used in this experiments as four classes, and there are 1269, 929, 824 and 852 examples in each class, with a total of 3874.

We used *Nearest Neighbor* classifier and *one-vs-rest SVMs* (Schölkopf & Smola, 2002) as baselines. The width of the RBF kernel for *SVM* was set to 5. In *LNP*, the number of nearest neighbors k was set to 5 when constructing the graph. For comparison, we also provide the classification results achieved by Zhou *et al.*'s *consistency* method (Zhou *et al*, 2004a) and Zhu *et al.*'s *Gaussian fields* approach (Zhu *et al*, 2003). The affinity matrix in both methods were constructed by a Gaussian function with variance 1.25. Note that the diagonal elements of the affinity matrix in Zhou's consistency method were set to 0. The recognition accuracies averaged over 50 independent trials are summarized in Fig.6(a), from which we can clearly see the advantages of *LNP* and Zhou *et al.*'s consistency method.

It is interesting to discover that our *LNP* method is very stable, *i.e.* 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 (while the image data of the same digit may reside on the same submanifold (Roweis & Saul, 2000)). *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.

[1] http://www.kernel-machines.org/data.html.

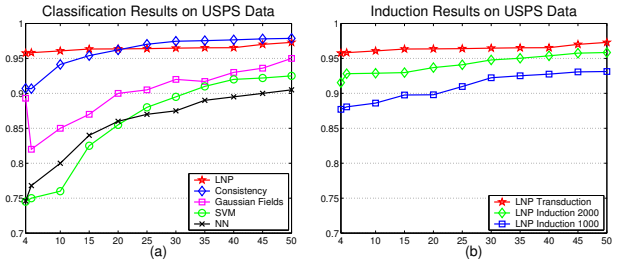

Figure 6. Digit recognition on the USPS dataset. A subset containing digits from 1 to 4 was adopted. (a) shows the recognition accuracies for different algorithms; (b) shows the recognition accuracies with 1000 and 2000 points selected for training, and the remaining data points for testing (induction). In both figures, the horizontal axis represents the number of randomly labeled data in the training set (we guarantee that there are at least one labeled point in each class), and the vertical axis is the total recognition accuracy value averaged over 50 independent runs.

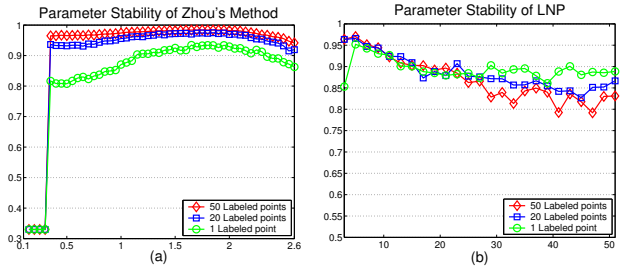

Figure 7. Parameter stability testing results. In both figures, the vertical axis represents the average recognition accuracy of 50 independent runs, and size of the randomly labeled points is set to 1, 20, 50 respectively. (a) shows the results achieved by *Zhou's consistency* method, where the horizontal axis is the variance of the RBF kernel; (b) shows the results achieved by our *LNP* method, where the horizontal axis represents the number of nearest neighbors.

We also test the effectiveness of the induction method for *LNP* introduced in section 3.3. First we split the whole dataset into two non-overlapping 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 testing set using Eq.(17). Fig.6(b) illustrates the induction results. We fix the size of the training set to be 1000 and 2000, and the recognition accuracies on these two datasets are plotted by the blue and green lines. For comparison, the results achieved by standard *LNP* (*i.e.* using the whole dataset for transduction) are also plotted in this figure. It is clearly observed that we can still get a high recognition accuracy using our induction method.

In the third part of this experiment we studied the stability of parameter, *i.e.* the number of the nearest neighbors k in *LNP*. For comparison, we also tested the stability of the variance σ of the RBF kernel in Zhou *et al.*'s consistency method. The results are shown in Fig.7. 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.

4.3. Text Classification

In this experiment, we addressed the task of text classification using 20-newsgroups dataset[2]. The topic *rec* containing *autos*, *motorcycles*, *baseball* and *hockey* was selected from the version 20news-18828. The articles were preprocessed by the same procedure as in (Zhou *et al*, 2004a). The resulted 3970 document vectors were all 8014-dimensional. Finally the document vectors were normalized into *TFIDF* representation.

We use the inner-product distance to find the k nearest neighbors when constructing the neighborhood graph in *LNP*, *i.e.* $d(\mathbf{x}_i, \mathbf{x}_j) = 1 - \mathbf{x}_i^T \mathbf{x}_j / (\|\mathbf{x}_i\| \|\mathbf{x}_j\|)$, where $\mathbf{x}_i$ and $\mathbf{x}_j$ are document vectors. And the value of k is set to 10. For Zhou's *consistency* and Zhu's *Gaussian fields* methods, the affinity matrices were all computed by $(\mathbf{W})_{ij} = exp\left(-(1/2\sigma^2)(1 - \mathbf{x}_i^T \mathbf{x}_j / (\|\mathbf{x}_i\| \|\mathbf{x}_j\|))\right)$ with $\sigma = 0.15$. The *SVM* and *Nearest Neighbor* classifiers were also served as the baseline algorithms, and the width of the *RBF* kernel in *SVM* is set to 1.5. The *accuracy vs. number of labeled points* plot is shown in Fig.8(a), where the accuracy values are averaged over 50 independent trials. From this figure we can clearly see the effectiveness of *LNP*.

Fig.8(b) illustrates the induction results of *LNP* on 20-newsgroup data. We fixed the size of the training set to be 1000, 2000 and 3000, and the remaining data were used for induction. Clearly, 1000 data points are not enough for describing the structure of the whole dataset, as the classification accuracies are dramatically poor. And 3000 points are sufficient for discovering this structure in that the classification accuracies can approximate the accuracies achieved by standard *LNP* which uses the whole dataset 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.9, where the meanings of the axes are the same as in Fig.7. We may find that the consistency method is very unstable in this experi-

[2] http://people.csail.mit.edu/jrennie/20Newsgroups/

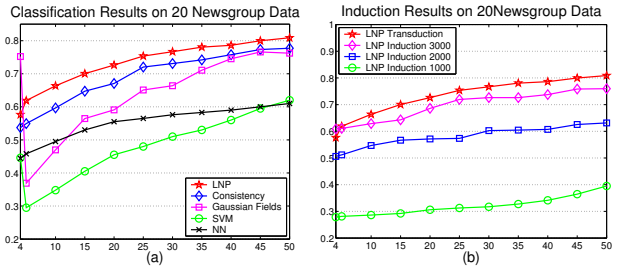

Figure 8. Classification accuracies on 20-newsgroup data. A subset of topic *rec* was adopted. (a) shows the recognition accuracies for different algorithms; (b) shows the recognition accuracies with 1000, 2000 and 3000 points selected for training, and the remaining data points for testing (induction). In both figures, the horizontal axis represents the number of randomly labeled data in the training set (we guarantee that there are at least one labeled point in each class), and the vertical axis is the total recognition accuracy value averaged over 50 independent runs.

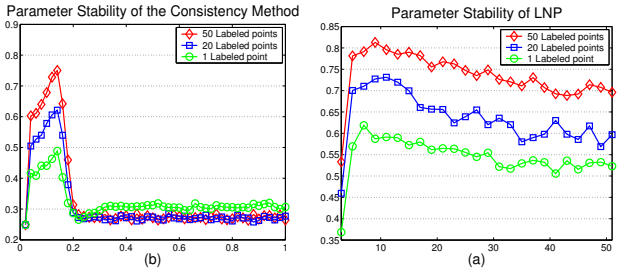

Figure 9. Parameter stability testing results. In both figures, the vertical axis represents the average recognition accuracy of 50 independent runs, and size of the randomly labeled points is set to 1, 20, 50 respectively. (a) shows the results achieved by *Zhou's consistency* method, where the horizontal axis is the width of the *RBF* kernel; (b) shows the results achieved by our *LNP* method, where the horizontal axis represents the number of nearest neighbors.

ment, since it can only achieve a high classification accuracy when σ falls into a very small range (between 0.1 and 0.2). By contrast our *LNP* method is much more stable, and it can hold a high classification accuracy as long as k is not too small.

5. Conclusions and Future Works

In this paper we propose a novel semi-supervised learning algorithm called *Linear Neighborhood Propagation* (*LNP*). It can discover the structure of the whole dataset through synthesizing the linear neighborhood around each data object. We also analyzed theoretically that the resulted data labels can be sufficiently smooth with respect to the data structure. Finally we provide many experiments to show the effectiveness of our method, from which we also find that *LNP* also have a high parameter stability. In our future, we will focus on the theoretical analysis and accelerating issues of our *LNP* algorithm.

References

Belkin, M., & Niyogi, P. (2002) Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering. *Advances in Neural Information Processing Systems 14.*

Belkin, M., Matveeva, I., & Niyogi, P. (2004) Regularization and Semi-supervised Learning on Large Graphs. In *Proceedings of the 17th Conference on Learning Theory.*

Belkin, M., Niyogi, P., & Sindhwani, V. (2005) On Manifold Regularization. In *Proceedings of the 10th International Workshop on Artificial Intelligence and Statistics.*

Chapelle, O., Weston, J., & Schölkopf, B. (2003). Cluster Kernels for Semi-Supervised Learning. In *Advances in Neural Information Processing Systems 15.*

Fan R. K. Chung. (1997). *Spectral Graph Theory.* Regional Conference Series in Mathematics, no. 92.

Delalleu, O., Bengio, Y., & Le Roux, N. (2005). Non-Parametric Function Induction in Semi-Supervised Learning. In *Proceedings of the 10th International Workshop on Artificial Intelligence and Statistics.*

Golub, G. H., Van Loan. C. F.(1989). *Matrix Computation.* 2nd ed. Baltimore.

Joachims, T. (1999). Transductive Inference for Text Classification using Support Vector Machines. *Proceedings of the 16th International Conference on Machine Learning.*

Roweis, S. T., & Saul, L. K. (2000). Noninear Dimensionality Reduction by Locally Linear Embedding. *Science*: vol. 290, 2323-2326.

Schölkopf, B., & Smola, A. J. (2002). *Learning with Kernels.* MIT Press, Cambridge, MA.

Szummer, M., & Jaakkola, T. (2002). Partially Labeled Classification with Markov Random Walks. *Advances in Neural Information Processing Systems 14.*

Tenenbaum, J.B., Silva, V., & Langford, J. C. (2000). A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science*: vol. 290, 2319-2323.

Zhou, D., Bousquet, O., Lal, T. N. Weston, J., & Schölkopf, B. (2004a). Learning with Local and Global Consistency. *Advances in Neural Information Processing Systems 16.*

Zhou, D. & Schölkopf, B. (2004b). Learning from Labeled and Unlabeled Data Using Random Walks. *Pattern Recognition, Proceedings of the 26th DAGM Symposium.*

Zhu, X., Ghahramani, Z., & Lafferty, J.(2003). Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions. In *Proceedings of the 20th International Conference on Machine Learning*

Zhu, X. (2005). Semi-Supervied Learning Literature Survey. *Computer Sciences Technical Report* 1530, University of Wisconsin-Madison.