Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

A general learning framework using local and global regularization

Fei Wang [*]

State Key Laboratory of Intelligent Technology and Systems, Tsinghua National Laboratory for Information Science and Technology (TNList), Department of Automation, Tsinghua University, Beijing 100084, China

ARTICLE INFO

Article history:
Received 30 June 2009
Received in revised form
25 March 2010
Accepted 27 March 2010

Keywords:
Machine learning
Local
Global
Regularization

ABSTRACT

In this paper, we propose a general learning framework based on local and global regularization. In the local regularization part, our algorithm constructs a regularized classifier for each data point using its neighborhood, while the global regularization part adopts a Laplacian regularizer to smooth the data labels predicted by those local classifiers. We show that such a learning framework can easily be incorporated into either unsupervised learning, semi-supervised learning, and supervised learning paradigm. Moreover, many existing learning algorithms can be derived from our framework. Finally we present some experimental results to show the effectiveness of our method.

1. Introduction

Most machine learning problems can be viewed as function estimation from the example data or the past experience by optimizing some objective criteria. In general, traditional machine learning algorithms can be categorized into three classes:

- *Supervised learning*. Supervised learning is a machine learning technique for estimating a classification function from training data set. The training data set is composed of pairs of input data objects and desired outputs. The output of the function is usually discrete associated with the *classification* task. The goal of the supervised learner is to predict the value of the function for any valid input object after having seen a number of training examples (i.e. pairs of input and target output).
- *Unsupervised learning*. In unsupervised learning, we are interested in capturing the inherent structure of the data. Typically, such a structure can be recovered by fitting a model (function) to the observations. It is different from supervised learning as there is no a priori output, i.e. only a data set of input objects is used.
- *Semi-supervised learning*. Semi-supervised learning [7] is a class of machine learning techniques that make use of both labeled and unlabeled data for training—typically a small amount of labeled data with a large amount of unlabeled data. The goal of semi-supervised learning is to predict the labels of the unlabeled data (*transduction*) and even the unseen data (*induction*).

From the above definitions we can see that supervised learning and unsupervised learning can all be viewed as special cases of semi-supervised learning, since in supervised learning, there are no unlabeled data in the training set, while in unsupervised learning, there are no labeled data in the training set.

Another taxonomy for machine learning algorithms is that they can either be *global* or *local*. A global learning machine trains the classifier using the whole data set as the training set, such as *SVM*, *Decision Tree*, while a local learning algorithm [5] aims to train the classifier system by using only useful local information of the data set. It has been noted that [21] the local learning algorithms usually demonstrate better empirical results since it is hard to find a unique function which has a good predictability in the whole sample space. Therefore, "thinking locally" might be a smart idea for many practical machine learning problems [18,19,26,27]. However, despite its empirical success, one problem for local learning is that the size of the samples used to train a local classifier is usually small, which usually makes the classifier not sufficiently trained. Therefore how to improve the performances of the local learning algorithms is still a challenging problem.

In this paper, we propose a novel learning framework based on local and global regularizations. From a semi-supervised learning perspective, our algorithm first constructs a set of local classifiers which can predict the labels of all the data points, a loss term will be minimized to make the predicted labels and the initial labels sufficiently close on the labeled points. Moreover, we add a global

* Tel.: 1 305 348 1218.
E-mail address: feiwang03@gmail.com
URL: http://feiwang03.googlepages.com

0031-3203/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2010.03.025

smoothness term to penalize the smoothness of the data labels, such smoothing can also alleviate the insufficient training of the local classifiers. We also derive the supervised and unsupervised counterparts of our framework and show that many existing algorithms can be naturally derived from this framework. Finally the experiments on several real world data sets are presented to show the effectiveness of our method.

2. The framework

2.1. Notations

Before we go into the details of our framework, first let us see some notations and symbols that will be frequently used in this paper.

Throughout this paper, we will use bold lowercase characters, e.g. $\mathbf{x}$, to denote vectors, bold uppercase characters, e.g. $\mathbf{X}$, to denote matrices. The bold italic lowercase/uppercase characters are used to denote the vector/matrix variables. Table 1 summarizes some frequently used notations in the rest of this paper.

2.2. Backgrounds

As we know that a general learning process can be described by three components [21]:

- A *generator* of the data set $\mathcal{X}$, such that each $\mathbf{x}_i \in \mathcal{X}$ is drawn from a fixed but usually unknown distribution $p(\boldsymbol{x})$;
- A *supervisor* which returns the output for each $\mathbf{x}_i \in \mathcal{X}$ according to $p(y|\boldsymbol{x})$;
- A *learning machine* which is capable of implementing a set of functions $f(\boldsymbol{x},\boldsymbol{w}), \boldsymbol{w} \in \mathcal{W}$ ($\mathcal{W}$ is the space of parameters).

The goal of learning is to choose from a given set of functions the one which best approximates the supervisor's responses. In traditional supervised learning, we are given a set of training sample-label pairs $\{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$, then a common principle for selecting a good f is to minimize the following *structural risk*

$$\mathcal{J}_g = \sum_{i=1}^{l} \mathcal{L}(y_i, f(\mathbf{x}_i, \mathbf{w})) + \lambda \|f\|_{\mathcal{F}}^2 \tag{1}$$

where $\mathcal{L}(\cdot,\cdot)$ is the loss function (e.g. square loss in regularized least square regression, hinge loss in SVM), $\|f\|_{\mathcal{F}}$ is the induced norm of f in the functional space $\mathcal{F}$ (e.g. $\mathcal{F}$ can be a *reproducing kernel Hilbert space* (*RKHS*) induced by some kernel k if we construct a kernel machine).

Clearly, Eq. (1) is a *supervised formulation*, which assumes that for the data set $\mathcal{X}$, we can construct an f whose prediction error can be measured by the first term of $\mathcal{R}$. The problem is that in many real world applications, it might be expensive and time-consuming for collecting enough labeled data to train a good classification function. Therefore semi-supervised learning, which aims to learn from partially labeled data, has been becoming more and more popular since usually the unlabeled data are much easier to obtain. Belkin et al. [4] proposed a general geometric framework for semi-supervised learning called *manifold regularization*, which seeks an optimal classification function f minimizing the following objective

$$\mathcal{R}_g = \sum_{i=1}^{l} \mathcal{L}(y_i, f(\mathbf{x}_i, \mathbf{w})) + \gamma_A \|f\|_{\mathcal{F}}^2 + \gamma_I \|f\|_{\mathcal{I}}^2, \tag{2}$$

where the additional penalty term $\|f\|_{\mathcal{I}}$ reflects the intrinsic geometric information of the marginal distribution $p(\boldsymbol{x})$. Since in the semi-supervised learning scenario, we only have a small portion of labeled data (i.e. l is small), which are not enough to train a good learner by minimizing $\mathcal{R}$. Therefore, we need some *prior knowledge* to guide us to learn a good f. What $p(\boldsymbol{x})$ reflects is just such type of prior information. Moreover, it is usually assumed [4] that there is a direct relationship between $p(\boldsymbol{x})$ and $p(y|\boldsymbol{x})$, i.e. if two points $\mathbf{x}_1$ and $\mathbf{x}_2$ are close in the intrinsic geometry of $p(\boldsymbol{x})$, then the conditional distributions $p(y|\mathbf{x}_1)$ and $p(y|\mathbf{x}_2)$ should be similar. In other words, $p(y|\boldsymbol{x})$ should vary smoothly along the geodesics in the intrinsic geometry of $p(\boldsymbol{x})$.

Specifically, Belkin et al. [4] showed that $\|f\|_{\mathcal{I}}^2$ can have the following form

$$\|f\|_{\mathcal{I}}^2 = \int_{\boldsymbol{x} \in \mathcal{M}} \|\nabla_{\mathcal{M}} f\|^2 \, dp(\boldsymbol{x}), \tag{3}$$

where $\mathcal{M}$ represents the low dimensional data manifold, $\nabla_{\mathcal{M}} f$ denotes the gradient of f with respect to $\mathcal{M}$. Intuitively, $\|f\|_{\mathcal{I}}^2$ measures how f varies on $\mathcal{M}$, i.e. the *smoothness* of f.

Belkin et al. [4] further points out that $\|f\|_{\mathcal{I}}^2$ can be approximated by

$$\hat{\mathcal{S}} = \frac{\gamma_I}{n^2} \sum_{i,j} (f(\mathbf{x}_i) - f(\mathbf{x}_j))^2 W_{ij} = \frac{\gamma_I}{n^2} \mathbf{f}^T \mathbf{L} \mathbf{f}, \tag{4}$$

where n is the total number of data points, and W_{ij} are the edge weights in the *data adjacency graph*, $\mathbf{f} = (f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n))^T$. $\mathbf{L} = \mathbf{D} - \mathbf{W} \in \mathbb{R}^{n \times n}$ is the *graph Laplacian* where $\mathbf{W}$ is the graph weight matrix with its (i,j)-th entry $\mathbf{W}(i,j) = W_{ij}$, and $\mathbf{D}$ is a diagonal *degree* matrix with $\mathbf{D}(i,i) = \sum_j W_{ij}$. There has been extensive discussion on that under certain conditions choosing *Gaussian weights* for the adjacency graph leads to convergence of the graph Laplacian to the *Laplace–Beltrami operator* $\triangle_{\mathcal{M}}$ (or its weighted version) on the manifold $\mathcal{M}$ [2,10].

2.3. A general principle based on local loss

Although [4] provides us an excellent framework for learning from labeled and unlabeled data, the structural loss (Eq. (1)) is defined in a global way, i.e. for the whole data set, we only need to pursue one classification function f that can minimize $\mathcal{J}_g$. According to [21], selecting a good f in such a global way might not be a good strategy because the function set $f(\boldsymbol{x},\boldsymbol{w}), \boldsymbol{w} \in \mathcal{W}$ may not contain a good predictor for the entire input space. But it is much easier for the set to contain some functions that are capable of producing good predictions on some specified regions of the input space. Therefore, if we split the whole input space into C local regions, then it is usually more effective to minimize the following local cost function for each region $\mathcal{R}_i (1 \leqslant i \leqslant k)$.

$$\hat{\mathcal{J}}_l^i = \sum_{j=1}^{l} K(\mathbf{x}_j - \mathbf{c}_i, \varepsilon_i) \mathcal{L}(y_j, f(\mathbf{x}_j, \mathbf{w}_i)), \tag{5}$$

where $K(\mathbf{x}_j - \mathbf{c}_i, \varepsilon_i)$ is a *local kernel* centered at $\mathbf{c}_i$ with the width ε_i. Fig. 1 shows two commonly used local kernel examples.

Table 1
Some frequently used notations.

$\mathcal{X}$	The data set
$\mathcal{M}$	The data manifold
$\mathcal{L}$	The loss function
$\mathbf{x}_i$	The i-th data point
$\mathbf{w}$	The parameters of the classifier
$\mathbf{f}$	The predicted label vector
y_i	The initial hard label of the i-th point
f_i	The predicted soft label of the i-th point
f	The classification function
l,u	The number of labeled/unlabeled points
n	The size of the training data set $n=l+u$

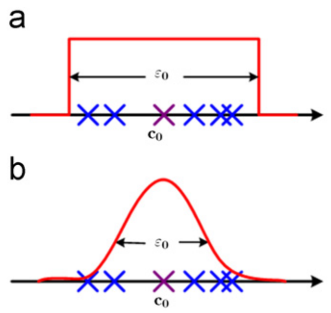

Fig. 1. Two examples of the local kernel K. (a) shows a *square kernel*, which only selects the data points in a specific neighborhood with "diameter" ε_0, (b) shows a *smooth kernel*, which gives different weights for all data points according to their position with respect to $\mathbf{c}_0$, and its size is controlled by ε_0.

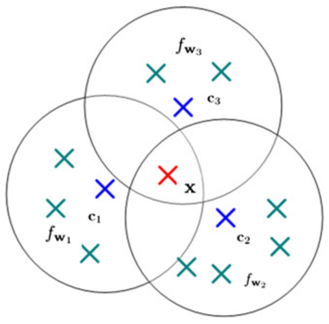

Fig. 2. A graphical illustration of the case when the local regions are overlapped with each other. In the figure, there are three overlapping local regions centered at $\mathbf{c}_1$,$\mathbf{c}_2$,$\mathbf{c}_3$, and we can construct three classification functions $f_{\mathbf{w}_1}$,$f_{\mathbf{w}_2}$,$f_{\mathbf{w}_3}$ from those three regions respectively. Then the label of $\mathbf{x}$ can be predicted via combining $f(\mathbf{x},\mathbf{w}_1)$,$f(\mathbf{x},\mathbf{w}_2)$,f($\mathbf{x}$,$\mathbf{w}_3$).

Then similar to Eq. (1), we can define the *local structural loss* for $f(\boldsymbol{x},\mathbf{w}_i)$ as

$$\mathcal{J}_l^i = \sum_{j=1}^{l} K(\mathbf{x}_j - \mathbf{c}_i, \varepsilon_i)\mathcal{L}(y_j, f(\mathbf{x}_j, \mathbf{w}_i)) + \frac{\gamma_A}{n_i^2}\|f_{\mathbf{w}_i}\|_{\mathcal{F}}^2, \quad (6)$$

which penalizes the complexity of f locally to increase its capacity (where we use $f_{\mathbf{w}_i}$ to denote the local classification function with parameter $\mathbf{w}_i$). In this way, we define a function f with parameter $\mathbf{w}_i$ for each local region $\mathcal{R}_i$ centered at $\mathbf{c}_i$, that is, we define C local classification functions $\{f_{\mathbf{w}_i}\}_{i=1}^C$, one for each $\mathbf{x}_i$, and the *total local structural loss* is

$$\mathcal{J}_l = \sum_{i=1}^{C} \mathcal{J}_l^i. \quad (7)$$

When there are some *overlappings* for those local regions, i.e. one data point $\mathbf{x}$ may belong to several local regions (see Fig. 2 as an example), then the classification function in each neighborhood can predict a class label for $\mathbf{x}$, and we can apply ensemble methods, like *majority voting*, to determine the final label of $\mathbf{x}$. In this way, the genuine function for classifying $\mathbf{x}$ is

$$\overline{f}(x) = Ensemble(f_i(\mathbf{x}), \mathbf{x} \in \mathcal{R}_i) \quad (8)$$

and the final label of $\mathbf{x}$ is directly determined by $\overline{f}$.

Now let us return to the semi-supervised learning scenario, which aims to learn from both labeled and unlabeled data, and the number of labeled points, i.e. l is usually very small. This may cause a problem that in each local region (neighborhood) defined by K we only have a few, or none known y_i, which will make f insufficiently trained (which will be called *small training sample size* (*STSS*) problem in the following). To solve this problem, we propose to introduce a set of *pseudo labels* $\{f_1, f_2, \ldots f_n\}$ playing the same role as in some *Bayesian methods* [13,24], such that f_i directly determines the genuine label of $\mathbf{x}_i$, i.e. y_i (f_i can be regarded as the soft label of $\mathbf{x}_i$, see Fig. 3 for the graphical illustration). Then we can redefine the *total local loss* as

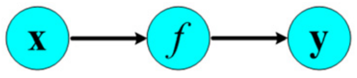

Fig. 3. The graphical illustration of introducing the pseudo labels, in which the data vector $\mathbf{x}$ determines the "pseudo label" f, and f determines y.

$$\mathcal{J}_l = \sum_{i=1}^{l} \mathcal{L}(f_i, y_i) + \lambda \cdot \left(\sum_{i=1}^{C}\sum_{j=1}^{n} K(\mathbf{x}_j - \mathbf{c}_i, \varepsilon_i)\mathcal{L}(f_j, f(\mathbf{x}_j, \mathbf{w}_i)) + \frac{\gamma_A}{n_i^2}\|f_{\mathbf{w}_i}\|_{\mathcal{F}}^2 \right),$$

where n is the size of the whole data set, and $\lambda > 0$ is a tradeoff parameter. The first term is called as *prediction loss*, and its minimization will cause f_i sufficiently close to y_i on the labeled data points. The second term is called as *local structural loss*, and its minimization will cause f to have the desired properties as we minimize Eq. (6) regarding f_i as the label of $\mathbf{x}_i$. In this way, f is trained locally using $\{\mathbf{x}_i, f_i\}_{i=1}^n$, which avoids the *small training sample size* problem we introduced previously since for every $\mathbf{x}_i$ there is a corresponding f_i. Note that by minimizing $\mathcal{J}_l$ we can obtain the optimal $\{f_i\}_{i=1}^n$ and $\{\mathbf{w}_i\}_{i=1}^C$. $\{f_i\}_{i=1}^n$ are important for *transduction* since they can directly predict the genuine labels of the data set,[1] $\mathbf{w}_i$ are important for *induction* since they directly determine the local prediction functions.

Recalling the *manifold regularization* framework introduced in Section 2.2, we may also expect f_i to have some geometrical properties. More concretely, we hope $\{f_i\}_{i=1}^n$ to be sufficiently smooth with respect to the intrinsic data graph, then we can define the following criterion similar to Eq. (2) for semi-supervised learning based on the local structural loss.

$$\begin{aligned}\mathcal{R}_l &= \mathcal{J}_l + \frac{\gamma_I}{n^2}\mathbf{f}^T\mathbf{L}\mathbf{f} \\ &= \sum_{i=1}^{l} \mathcal{L}(f_i, y_i) + \frac{\gamma_I}{n^2}\sum_{i,j} w_{i,j}(f_i - f_j)^2 \\ &\quad + \lambda \cdot \left(\sum_{i=1}^{C}\sum_{j=1}^{n} K(\mathbf{x}_j - \mathbf{c}_i, \varepsilon_i)\mathcal{L}(f_j, f(\mathbf{x}_j, \mathbf{w}_i)) + \frac{\gamma_A}{n_i^2}\|f_{\mathbf{w}_i}\|_{\mathcal{F}}^2 \right), \end{aligned} \quad (9)$$

where $\mathbf{f} = (f_1, f_2, \ldots f_n)^T$, $\mathbf{L} \in \mathbb{R}^{n \times n}$ is the *Laplacian* defined on the data graph. Clearly, the first term $\mathcal{J}_l$ punishes the predictability and complexity of the local prediction functions (which is thus called *local regularization*), and the second term penalizes the smoothness of the data labels over the entire data graph (which is thus referred to as *global regularization*).

Therefore, to derive a practical semi-supervised learning algorithm by minimizing $\mathcal{R}_l$ defined in Eq. (9), we should define: (1) a proper local kernel K; (2) a proper loss function $\mathcal{L}(\cdot,\cdot)$; (3) a proper form of local learning functions. In the following we will introduce a concrete, and simple algorithm that can carry out the principles introduced in this subsection.

2.4. A simple algorithm

In this subsection, we introduce a simple algorithm based on *local and global regularization* (*LGReg*). In our algorithm, the local

[1] Associated with Eq. (8), f_i can be regarded as the output of $\overline{f}(\mathbf{x}_i)$

kernel K is defined on the *k-nearest neighborhood* around each data point $\mathbf{x}_i$. Mathematically,

$$K(\mathbf{x}_j - \mathbf{c}_i, \varepsilon_i) = \begin{cases} 1 & \text{if } \mathbf{x}_j \in \mathcal{N}(\mathbf{x}_i), \\ 0 & \text{otherwise,} \end{cases} \tag{10}$$

where $\mathcal{N}(\mathbf{x}_i)$ denotes the *k-nearest neighborhood* of $\mathbf{x}_i$, and the parameter $\mathbf{c}_i = 1/|\mathcal{N}(\mathbf{x}_i)|\sum_{\mathbf{x}_j \in \mathcal{N}(\mathbf{x}_i)} \mathbf{x}_j$ ($|\mathcal{N}(\mathbf{x}_i)|$ represents the cardinality of $\mathcal{N}(\mathbf{x}_i)$), $\varepsilon_i = 2\max_{\mathbf{x}_j}\|\mathbf{x}_j - \mathbf{c}_i\|$. Such a kernel is one type of *square kernel* as shown in Fig. 1. Using this kernel, we in fact split the whole input space into n overlapping local regions, and each region corresponds to one *k-nearest neighborhood* of a data point. The loss function $\mathcal{L}(\cdot,\cdot)$ is set to be the square loss because of its simplicity and wide applications, i.e.

$$\mathcal{L}(y_i, f_i) = (y_i - f_i)^2 \tag{11}$$

$$\mathcal{L}(f_j, f(\mathbf{x}_j, \mathbf{w}_i)) = (f_j - f(\mathbf{x}_j, \mathbf{w}_i))^2. \tag{12}$$

We select *linear classifier* as the local classifiers, i.e.

$$f(\mathbf{x}_j, \{\mathbf{w}_i, b_i\}) = \mathbf{w}_i^T\mathbf{x}_j + b_i, \tag{13}$$

where $\mathbf{w}_i$, b_i are the *weight vector* and *bias* of the linear classifier. Note that we can define the *augmented* weight vector $\tilde{\mathbf{w}}_i = [\mathbf{w}_i^T, b_i]^T$ and the *augmented* data vector $\tilde{\mathbf{x}}_j = [\mathbf{x}_j^T, 1]^T$ such that

$$f(\mathbf{x}_j, \{\mathbf{w}_i, b_i\}) = f(\tilde{\mathbf{x}}_j, \tilde{\mathbf{w}}_i) = \tilde{\mathbf{w}}_i^T\tilde{\mathbf{x}}_j. \tag{14}$$

Then the *total local loss* becomes

$$\begin{aligned}\mathcal{J}_l &= \sum_{i=1}^{l}\mathcal{L}(f_i, y_i) \\ &+ \lambda \cdot \left(\sum_{i=1}^{C}\sum_{j=1}^{n} K(\mathbf{x}_j - \mathbf{c}_i, \varepsilon_i)\mathcal{L}(f_j, f(\mathbf{x}_j, \mathbf{w}_i)) + \frac{\gamma_A}{n_i^2}\|f_{\mathbf{w}_i}\|_{\mathcal{F}}^2\right) \\ &= \sum_{i=1}^{l}(f_i - y_i)^2 + \lambda\hat{\mathcal{J}}_l,\end{aligned}$$

where the *local structural loss*

$$\hat{\mathcal{J}}_l = \left(\sum_{i=1}^{n}\sum_{\mathbf{x}_j \in \mathcal{N}(\mathbf{x}_i)}(\mathbf{w}_i^T\mathbf{x}_j + b_i - f_j)^2 + \frac{\gamma_A}{n_i^2}\|\mathbf{w}_i\|^2\right). \tag{15}$$

One problem in the above formulation is that there are only a few data points in each neighborhood (the *STSS* problem), then the *structural penalty* term $\|\mathbf{w}_i\|$ will pull the weight vector $\mathbf{w}_i$ toward some arbitrary origin. For isotropy reasons, we translate the origin of the input space to the *neighborhood medoid* $\mathbf{x}_i$, by subtracting $\mathbf{x}_i$ from the training points $\mathbf{x}_j \in \mathcal{N}(\mathbf{x}_i)$, i.e.

$$f(\mathbf{x}_j, \{\mathbf{w}_i, b_i\}) = \mathbf{w}_i^T(\mathbf{x}_j - \mathbf{x}_i) + b_i. \tag{16}$$

Then the *local structural loss* becomes

$$\hat{\mathcal{J}}_l = \sum_i\sum_{\mathbf{x}_j \in \mathcal{N}_i}(\mathbf{w}_i^T(\mathbf{x}_j - \mathbf{x}_i) + b_i - f_j)^2 + \frac{\gamma_A}{n_i^2}\|\mathbf{w}_i\|^2. \tag{17}$$

Now we define

$$\mathcal{J}_l^i = \sum_{\mathbf{x}_j \in \mathcal{N}_i}(\mathbf{w}_i^T(\mathbf{x}_j - \mathbf{x}_i) + b_i - f_j)^2 + \frac{\gamma_A}{n_i^2}\|\mathbf{w}_i\|^2 \tag{18}$$

to be the local structural loss on each data point $\mathbf{x}_i$, which can be rewritten in its matrix form as

$$\mathcal{J}_i^l = \left\|\mathbf{G}_i\begin{bmatrix}\mathbf{w}_i \\ b_i\end{bmatrix} - \tilde{\mathbf{f}}_i\right\|^2,$$

where

$$\mathbf{G}_i = \begin{bmatrix}\mathbf{x}_{i_1}^T - \mathbf{x}_i^T & 1 \\ \mathbf{x}_{i_2}^T - \mathbf{x}_1^T & 1 \\ \vdots & \vdots \\ \mathbf{x}_{i_{n_i}}^T - \mathbf{x}_i^T & 1 \\ \dfrac{\sqrt{\gamma_A}}{n_i}\mathbf{I}_d & \mathbf{0}\end{bmatrix}, \quad \tilde{\mathbf{f}}_i = \begin{bmatrix}f_{i_1} \\ f_{i_2} \\ \vdots \\ f_{i_{n_i}} \\ \mathbf{0}\end{bmatrix},$$

where $\mathbf{x}_{i_j}$ represents the j-th neighbor of $\mathbf{x}_i$, n_i is the cardinality of $\mathcal{N}_i$, and $\mathbf{0}$ is a $d \times 1$ zero vector, d is the dimensionality of the data vectors. By taking $\partial\mathcal{J}_i^l/\partial(\mathbf{w}_i, b_i) = 0$, we can get that

$$\begin{bmatrix}\mathbf{w}_i \\ b_i\end{bmatrix}^* = (\mathbf{G}_i^T\mathbf{G}_i)^{-1}\mathbf{G}_i^T\tilde{\mathbf{f}}_i. \tag{19}$$

Then the total loss we want to minimize becomes

$$\mathcal{J}^l = \sum_i\mathcal{J}_i^l = \sum_i\tilde{\mathbf{f}}_i\tilde{\mathbf{G}}_i^T\tilde{\mathbf{G}}_i\tilde{\mathbf{f}}_i, \tag{20}$$

where

$$\tilde{\mathbf{G}}_i = \mathbf{I} - \mathbf{G}_i(\mathbf{G}_i^T\mathbf{G}_i)^{-1}\mathbf{G}_i^T.$$

Then we have the following lemma.

Lemma 1. *$\tilde{\mathbf{G}}_i$ is an orthogonal projection matrix* [20].

Proof. See Appendix I.

Then we can rewrite $\mathcal{J}^l$ in Eq. (20) as

$$\mathcal{J}^l = \sum_i\tilde{\mathbf{f}}_i^T\tilde{\mathbf{G}}_i\tilde{\mathbf{f}}_i.$$

If we partition $\tilde{\mathbf{G}}_i$ into four block as

$$\tilde{\mathbf{G}}_i = \begin{bmatrix}\mathbf{A}_i^{n_i \times n_i} & \mathbf{B}_i^{n_i \times d} \\ \mathbf{C}_i^{d \times n_i} & \mathbf{D}_i^{d \times d}\end{bmatrix}.$$

Let $\mathbf{f}_i = [f_{i_1}, f_{i_2}, \ldots, f_{i_{n_i}}]^T$, then

$$\tilde{\mathbf{f}}_i^T\tilde{\mathbf{G}}_i\tilde{\mathbf{f}}_i = [\mathbf{f}_i^T \quad \mathbf{0}]\begin{bmatrix}\mathbf{A}_i & \mathbf{B}_i \\ \mathbf{C}_i & \mathbf{D}_i\end{bmatrix}\begin{bmatrix}\mathbf{f}_i \\ \mathbf{0}\end{bmatrix} = \mathbf{f}_i^T\mathbf{A}_i\mathbf{f}_i.$$

Thus

$$\mathcal{J}^l = \sum_i\mathbf{f}_i^T\mathbf{A}_i\mathbf{f}_i. \tag{21}$$

Furthermore, we have the following theorem, which is used to imply that $\hat{\mathbf{G}}$ has the same property with traditional Laplacian matrix.

Theorem 1.

$$\begin{aligned}\mathbf{A}_i = \mathbf{I}_{n_i} - \Bigg(&\mathbf{X}_i^T\mathbf{H}_i^{-1}\mathbf{X}_i + \frac{\mathbf{X}_i^T\mathbf{H}_i^{-1}\mathbf{X}_i\mathbf{1}\mathbf{1}^T\mathbf{X}_i^T\mathbf{H}_i^{-1}\mathbf{X}_i}{n_i - c} \\ &- \frac{\mathbf{X}_i^T\mathbf{H}_i^{-1}\mathbf{X}_i\mathbf{1}\mathbf{1}^T}{n_i - c} - \frac{\mathbf{1}\mathbf{1}^T\mathbf{X}_i^T\mathbf{H}_i^{-1}\mathbf{X}_i}{n_i - c} + \frac{\mathbf{1}\mathbf{1}^T}{n_i - c}\Bigg),\end{aligned} \tag{22}$$

where $\mathbf{H}_i = \mathbf{X}_i\mathbf{X}_i^T + \frac{\gamma_A}{n_i^2}\mathbf{I}_d, c = \mathbf{1}^T\mathbf{X}_i^T\mathbf{H}_i^{-1}\mathbf{X}_i\mathbf{1}$, $\mathbf{1} \in \mathbb{R}^{n_i \times 1}$ *is a column vector with all its elements equaling to* 1, *and* $\mathbf{A}_i\mathbf{1} = \mathbf{0}$.

Proof. See Appendix II.

If we define the label vector $\mathbf{f} = [f_1, f_2, \ldots, f_n]^T \in \mathbb{R}^{n \times 1}$, the *concatenated* label vector $\hat{\mathbf{f}} = [\mathbf{f}_1^T, \mathbf{f}_2^T, \ldots, \mathbf{f}_n^T]^T$ and the *concatenated* block-diagonal matrix

$$\hat{\mathbf{G}} = \begin{bmatrix} \mathbf{A}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{A}_n \end{bmatrix},$$

which is of size $\sum_i n_i \times \sum_i n_i$. Then from Eq. (21) we can derive that

$$\mathcal{J}^l = \hat{\mathbf{f}}^T \hat{\mathbf{G}} \hat{\mathbf{f}}.$$

Define the *selection matrix* $\mathbf{S} \in \{0,1\}^{\sum_i n_i \times n}$, which is a 0-1 matrix with $S_{ij} = 1$ if $\mathbf{x}_j \in \mathcal{N}_i$, such that

$$\hat{\mathbf{f}} = \mathbf{S}\mathbf{f}$$

Then

$$\mathcal{J}^l = \mathbf{f}^T \mathbf{S}^T \hat{\mathbf{G}} \mathbf{S} \mathbf{f}.$$

Let

$$\mathbf{M} = \mathbf{S}^T \hat{\mathbf{G}} \mathbf{S} \in \mathbb{R}^{n \times n}, \tag{23}$$

which is a square matrix, then we can rewrite $\mathcal{J}^l$ in the form of

$$\mathcal{J}^l = \mathbf{f}^T \mathbf{M} \mathbf{f}. \tag{24}$$

To compute $\mathbf{M}$, we need to construct $\mathbf{S}, \hat{\mathbf{G}}$ first. According to its definition in Eq. (22), for each block matrix $\mathbf{A}_i$ on the diagonal line of $\hat{\mathbf{G}}$, we need to compute $\mathbf{H}_i^{-1}$, which is the inverse of a $d \times d$ matrix, and this brings a high computational burden. To make our algorithm computationally more efficient, we introduce the following theorem.

Theorem 2. *For* $\forall \mathbf{r} \in \mathbb{R}^{n_i \times 1}$, *we have*

$$\mathbf{H}_i^{-1} \mathbf{X}_i \mathbf{r} = \mathbf{X}_i \overline{\mathbf{H}}_i^{-1} \mathbf{r},$$

where

$$\overline{\mathbf{H}}_i = \mathbf{X}_i^T \mathbf{X}_i + \frac{\gamma_A}{n_i^2} \mathbf{I}_{n_i}, \tag{25}$$

and $\mathbf{I}_{n_i}$ *is the identity matrix of size* $n_i \times n_i$.

Proof. See Appendix III.

Therefore, using Theorem 2, we can compute

$$\hat{\mathbf{A}}_i = \mathbf{I}_{n_i} - \left(\mathbf{X}_i^T \mathbf{X}_i \overline{\mathbf{H}}_i^{-1} + \frac{\mathbf{X}_i^T \mathbf{X}_i \overline{\mathbf{H}}_i^{-1} \mathbf{1}\mathbf{1}^T \mathbf{X}_i^T \mathbf{X}_i \overline{\mathbf{H}}_i^{-1}}{n_i - c} - \frac{\mathbf{X}_i^T \mathbf{X}_i \overline{\mathbf{H}}_i^{-1} \mathbf{1}\mathbf{1}^T}{n_i - c} - \frac{\mathbf{1}\mathbf{1}^T \mathbf{X}_i^T \mathbf{X}_i \overline{\mathbf{H}}_i^{-1}}{n_i - c} + \frac{\mathbf{1}\mathbf{1}^T}{n_i - c} \right),$$

and construct $\hat{\mathbf{G}}$ as

$$\hat{\mathbf{G}} = \begin{bmatrix} \hat{\mathbf{A}}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \hat{\mathbf{A}}_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \hat{\mathbf{A}}_n \end{bmatrix}, \tag{26}$$

where $\overline{\mathbf{H}}_i$ is defined in Eq. (25). In this way, we only need to compute the inverse of a matrix of size $n_i \times n_i$ when constructing $\hat{\mathbf{A}}_i$ and usually $n_i \ll d$.

Therefore, combining Eqs. (9),(11) and (11) together, we can derive that the criterion our algorithm tries to minimize is

$$\mathcal{R}_l = \sum_{i=1}^{l} (y_i - f_i)^2 + \lambda \mathbf{f}^T \mathbf{M} \mathbf{f} + \frac{\gamma_I}{n^2} \mathbf{f}^T \mathbf{L} \mathbf{f}. \tag{27}$$

Then

$$\frac{\partial \mathcal{R}_l}{\partial \mathbf{f}} = 2\mathbf{J}(\mathbf{f} - \mathbf{y}) + \lambda(\mathbf{M} + \mathbf{M}^T)\mathbf{f} + \frac{2\gamma_I}{n^2} \mathbf{L}\mathbf{f}, \tag{28}$$

where $\mathbf{J} \in \mathbb{R}^{n \times n}$ is a diagonal matrix with its (i,i)-th entry

$$\mathbf{J}(i,i) = \begin{cases} 1 & \text{if } \mathbf{x}_i \text{ is labeled,} \\ 0 & \text{otherwise.} \end{cases} \tag{29}$$

By taking $\partial \mathcal{R}_l / \partial \mathbf{f} = 0$, we can get that

$$\mathbf{f} = \left(2\mathbf{J} + \lambda(\mathbf{M} + \mathbf{M}^T) + \frac{2\gamma_I}{n^2} \mathbf{L} \right)^{-1} \mathbf{J}\mathbf{y}, \tag{30}$$

where $\mathbf{y}$ is an $n \times 1$ column vector with its i-th entry

$$\mathbf{y}(i) = \begin{cases} y_i & \text{if } \mathbf{x}_i \text{ is labeled,} \\ 0 & \text{otherwise.} \end{cases}$$

To summarize, the basic flowchart of our algorithm is shown in Table 2.

A natural question would be how to predict the label of an unseen testing data point, which has not appeared in $\mathcal{X}$. We propose a three-step approach here to solve this problem.

Step 1. Solve the optimal label vector $\mathbf{f}^*$ using the algorithm shown in Table 2.

Step 2. Solve the parameters $\{\mathbf{w}_i^*, b_i^*\}$ of the optimal local classification functions using Eq. (19).

Step 3. For a new testing point $\mathbf{x}$, first identify the local regions that $\mathbf{x}$ falls in, then apply the local prediction functions of the corresponding regions to predict its label, and finally ensembling those results to get the final predicted label of $\mathbf{x}$.

2.5. Purely unsupervised and supervised cases

While the framework introduced in Section 2 is concentrated on the semi-supervised case, it also covers both purely *unsupervised* and *supervised* cases as well. We briefly discuss each of them in this subsection.

2.5.1. Unsupervised learning with local and global regularization

In the unsupervised case one is given a collection of unlabeled data points $\mathbf{x}_1, \ldots, \mathbf{x}_u$. Then the *local structural loss* defined in Eq. (17) would not contain the first term, since no labeled data is available. Therefore combining with the global geometrical regularization term as in Eq. (2), we are left with the optimization problem of minimizing the following criterion [23,22].

$$\mathcal{J}_{un} = \sum_{i=1}^{C} \sum_{j=1}^{n} K(\mathbf{x}_j - \mathbf{c}_i, \varepsilon_i) \mathcal{L}(f_j, f(\mathbf{x}_j, \mathbf{w}_i)) + \gamma_A \sum_{i=1}^{C} \|f_{\mathbf{w}_i}\|_{\mathcal{F}}^2 + \frac{\gamma_I}{u^2} \mathbf{f}^T \mathbf{L} \mathbf{f}$$

Specifically, following the simple algorithm presented in Section 2.4, in which we define $K(\mathbf{x}_j - \mathbf{c}_i, \varepsilon_i)$ as in Eq. (10), $f(\mathbf{x}_j, \mathbf{w}_i)$ as in Eq.

Table 2
Semi-supervised learning with local and global regularization.

Inputs
The data set $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n$
The neighborhood size k
The Gaussian kernel variance when constructing $\mathbf{L}$
The regularization parameter $\lambda, \gamma_A, \gamma_i$
Output
The predicted soft data label vector $\mathbf{f}$
Procedure
1. Construct $\mathbf{L}$ using *Gaussian* functions
2. Construct $\mathbf{M}$ as in Eq. (23)
3. Compute $\mathbf{f}$ as in Eq. (30)

(14), and using the square loss, we can derive that

$$\mathcal{J}_{un}=\mathbf{f}^T\mathbf{M}\mathbf{f}+\frac{\gamma_I}{u^2}\mathbf{f}^T\mathbf{L}\mathbf{f}, \tag{31}$$

where the matrix $\mathbf{M}$ is defined in Eq. (23).

If we further restrict $\mathbf{f}^T\mathbf{f}=1$, then from the *Ky Fan* theorem [9], f can be solved via the eigenvalue decomposition of the matrix $\mathbf{M}+(\gamma_I/u^2)\mathbf{L}$. We can further restrict f to be nonnegative and minimize $\mathcal{J}_{un}$ via *nonnegative matrix factorization*, since some empirical comparison shows that the nonnegative versions of those clustering algorithms usually achieve better performances [25].

2.5.2. Purely supervised learning with local and global regularization

The fully supervised case achieves the other end of the spectrum of learning which, as we have introduced in the introduction, can also be viewed as a special case of semi-supervised learning. Following the discussions in Section 2, if the local regions we split are not intersected with each other, then we can learn the local classifiers by minimizing the following cost function for the two-class case.

$$\mathcal{J}_{su}=\sum_{i=1}^{C}\sum_{j=1}^{n}K(\mathbf{x}_j-\mathbf{c}_i,\varepsilon_i)\mathcal{L}(y_j,f(\mathbf{x}_j,\mathbf{w}_i))$$
$$+\gamma_A\sum_{i=1}^{C}\|f_{\mathbf{w}_i}\|_{\mathcal{F}}^2+\frac{\gamma_I^+}{u^2}\mathbf{f}_+^T\mathbf{L}_+\mathbf{f}_++\frac{\gamma_I^-}{u^2}\mathbf{f}_-^T\mathbf{L}_-\mathbf{f}_-,$$

where $\mathbf{f}_+=[f(\mathbf{x}_1^+,\mathbf{w}_{\mathcal{R}_{\mathbf{x}_1^+}}),\dots,f(\mathbf{x}_{n^+}^+,\mathbf{w}_{\mathcal{R}_{\mathbf{x}_{n^+}^+}})]^T$, $\mathbf{x}_i^+$ denotes the i-th data point whose label is positive, and $\mathcal{R}_{\mathbf{x}_i^+}$ represents the region that $\mathbf{x}_i^+$ belongs to, n^+ is the number of positive points, $\mathbf{L}^+$ is the graph Laplacian constructed on the positive data points, and similarly for $\mathbf{f}^-$ and $\mathbf{L}^-$. In this case, we can usually get closed form solution for $\{\mathbf{w}_i\}_{i=1}^C$. Then for a new testing point $\mathbf{x}$, we can predict its label by the local function of the region that it falls in.

However, in the case when the local regions are intersected with each other, the genuine classification function we used to predict the data labels is $\overline{f}$ in Eq. (8). Then we can also introduce a set of pseudo labels $\mathbf{f}=[\overline{f}(\mathbf{x}_1),\overline{f}(\mathbf{x}_2),\dots,\overline{f}(\mathbf{x}_n)]^T$, and define the following criterion

$$\mathcal{J}'_{su}=\sum_{i=1}^{n}(f_i-y_i)^2+\frac{\gamma_I^+}{u^2}\mathbf{f}_+^T\mathbf{L}_+\mathbf{f}_++\frac{\gamma_I^-}{u^2}\mathbf{f}_-^T\mathbf{L}_-\mathbf{f}_-$$
$$+\lambda\sum_{i=1}^{C}\sum_{j=1}^{n}K(\mathbf{x}_j-\mathbf{c}_i,\varepsilon_i)\mathcal{L}(f_j,f(\mathbf{x}_j,\mathbf{w}_i))+\gamma_A\|f_{\mathbf{w}_i}\|_{\mathcal{F}}^2,$$

in which there are two sets of variables: $\{\mathbf{w}_i\}$ and $\{f_i\}$, then we can apply the same trick as in Eq. (19), i.e. we first represent $\{\mathbf{w}_i\}$ via $\mathbf{f}$, and then solve $\overline{\mathbf{f}}$ by minimizing $\mathcal{J}'_{su}$, then we can obtain $\{\mathbf{w}_i\}$ afterwards. For a new testing point $\mathbf{x}$, we can first check which region it falls in, and then predict its labels via ensembling the results of those local functions of the corresponding regions. Note that this method has the same spirit as the *induction* method introduced in Section 2.4.

3. Discussions

In this section, we discuss the relationships between the proposed framework with some existing related approaches, and present another mixed regularization framework for the algorithm presented in Section 2.4.

3.1. Relationship with related approaches

There has already been some semi-supervised learning algorithms based on different regularizations. In this subsection, we will discuss the relationships between our algorithm with those existing approaches.

3.1.1. Relationship with Gaussian–Laplacian regularized approaches

Most of traditional graph based semi-supervised learning algorithms (e.g. [1,30,31]) are based on the following framework [1]

$$\mathbf{f}=\underset{\mathbf{f}}{\operatorname{argmin}}\sum_{i=1}^{l}(f_i-y_i)^2+\zeta\mathbf{f}^T\mathbf{L}\mathbf{f}, \tag{32}$$

where $\mathbf{f}=[f_1,f_2,\dots,f_l,\dots,f_n]^T$, $\mathbf{L}$ is the *graph Laplacian* constructed by Gaussian functions. Clearly, the above framework is just a special case of our algorithm if we set $\lambda=0,\gamma_I=n^2\zeta$ and $\mathcal{L}$ to be the quadratic loss in Eq. (9).

3.1.2. Relationship with local learning regularized approaches

Recently, Wu et al. [27] proposed a novel transduction method based on local learning, which aims to solve the following optimization problem

$$\mathbf{f}=\underset{\mathbf{f}}{\operatorname{argmin}}\sum_{i=1}^{l}(f_i-y_i)^2+\zeta\sum_{i=1}^{n}\|f_i-o_i\|^2, \tag{33}$$

where o_i is the label of $\mathbf{x}_i$ predicted by the local classifier constructed on the neighborhood of $\mathbf{x}_i$, and the parameters of the local classifier can be represented by $\mathbf{f}$ via minimizing some local structural loss functions as in Eq. (19).

This approach can be understood as a two-step approach for optimizing Eq. (9): in the fist step, it optimizes the classifier parameters by minimizing local structural loss (Eq. (17)), since only this term contains the parameters in R_l (Eq. (9); in the second step, it minimizes the prediction loss of each data points by the local classifier constructed just on its neighborhood. It may not be appropriate for the second step, since each data point may belongs to multiple neighborhoods, and there exists one classifier on each neighborhood, then it is more reasonable to expect all those local classifiers to have better prediction abilities, which may leads to the minimization of $\mathcal{R}_l$ with $\gamma_I=0$, just like what the authors in [15] did.

3.1.3. Relationship with dimensionality reduction

Dimensionality reduction (DR) is another subfield in machine learning. Its goal is to seek a low dimensional representation to facilitate further processing. One of the most well-known DR method is *principal component analysis* (PCA) [12], which is a linear DR method making use of the entire data set to find a projection direction on which the data variance can be maximized. In the last decade many nonlinear DR method (e.g., *locally linear embedding* (LLE) [17] and *Laplacian embedding* (LE) [3]) are proposed, where they usually want to preserve the local structure contained in the data set. Recently Zhang et al. [29] proposed a general DR framework called *patch alignment* (*PA*), which first explore the data structures locally and then align them globally. The authors showed that many existing dimensionality reduction algorithms could be derived from their framework. Although the idea of PA is quite similar to the method proposed in this paper, their formulation is completely different because our method is for classification, while PA is for dimensionality reduction. [16] proposed a semi-supervised DR method called *transductive component analysis* (*TCA*), which makes use of the whole data set to get a projection direction. However, the method proposed in this paper constructs a linear classifier for each local region.

3.2. A mixed-regularization viewpoint

In Table 2, we have proposed an algorithm implementing the principle in Section 2.3, which aims to minimize the following criterion

$$\mathcal{R}_l = \sum_{i=1}^{l}(y_i - f_i)^2 + \lambda \mathbf{f}^T\mathbf{M}\mathbf{f} + \frac{\gamma_I}{n^2}\mathbf{f}^T\mathbf{L}\mathbf{f}, \tag{34}$$

where $\mathbf{M}$ is defined in Eq. (23) and $\mathbf{L}$ is the conventional graph Laplacian constructed by Gaussian functions. It is easy to prove that $\mathbf{M}$ has the following property.

Theorem 3. $\mathbf{M1} = \mathbf{0}$, *where* $\mathbf{1} \in \mathbb{R}^{n\times 1}$ *is a column vector with all its elements equaling to* 1.

Proof. From the definition of $\mathbf{M}$ (Eq. (23)), we have

$$\mathbf{M1} = \mathbf{S}^T\hat{\mathbf{G}}\mathbf{S1} = \mathbf{S}^T\hat{\mathbf{G}}\mathbf{1} = \mathbf{0},$$

which proves the theorem. □

Therefore, the row sum of $\mathbf{M}$ is also 0, which is similar to the Laplacian matrix. That is, the last two terms of $\mathcal{R}_l$ can all be viewed as regularization terms with different regularization matrices, one is derived from *local learning*, the other is derived from the *heat kernel*. Hence our algorithm can also be understood from a *mixed regularization* viewpoint [6,32]. Just like the *multi-view learning* algorithm [28], which trains the same type of classifier using different data features, our method trains different classifiers using the same data features. Different types of regularization matrices may better reveal different (maybe complementary) information and thus provide a more powerful classifier.

4. Experiments

In this section, we present a set of experiments to show the effectiveness of our method. First let us describe the basic information of the data sets.

4.1. The data sets

Twelve datasets, including 2 artificial datasets and 10 real datasets, are used in our experiments to evaluate the performances of the methods. Table 3 summarizes the characteristics of the datasets.

- *Toy data sets: g241c and g241n.* Each data set contains two classes with 350 points in each class, and the data sets are generated in a way of violating the cluster assumptions or misleading class structures. For details one can refer to [7].

Table 3
Descriptions of the datasets.

Datasets	Sizes	Classes	Dimensions
g241c	1500	2	241
g241n	1500	2	241
USPS	1500	2	241
COIL	1500	6	241
digit1	1500	2	241
cornell	827	7	4134
texas	814	7	4029
wisconsin	1166	7	4189
washington	1210	7	4165
BCI	400	2	117
diabetes	768	2	8
ionosphere	351	2	34

- *Image data sets: USPS, COIL and digit1.* The first two data sets are generated from the famous USPS[2] and COIL[3] databases, such that the resultant image data did not appear to be manifolds explicitly. The *digit 1* data set was generated by transforming the image of digit 1, and the image data appears a manifold structure strongly. For the detailed generation information of the three data sets one can refer to [7].
- *Text data sets: cornell, texas, wisconsin and washington.* All these four data sets are selected from the famous WebKB database,[4] and the web pages are classified into seven categories.
- *Other data sets: BCI, diabetes and ionosphere.* The BCI data set is provided in [14], which originates from research toward the development of a brain computer interface (BCI). The Diabetes and Ionosphere data sets are downloaded from the UCI repository,[5] which are usually used as the benchmark data sets for semi-supervised learning algorithms.

4.2. Methods and parameter settings

Besides our method, we also implement some other competing methods for experimental comparison. For all the methods, their hyperparameters were set by five-fold cross validation from some grids introduced in the following.

- *Local and global regularization* (*LGReg*). The implementation of the algorithm is based on Table 2, in which we use the *mutual neighborhood* and the size of the neighborhood is searched from {5,10,50}, γ_A is searched from $\{4^{-3},4^{-2},4^{-1},4^0,4^1,4^2,4^3\}$, λ is also searched from $\{4^{-3},4^{-2},4^{-1},4^0,4^1,4^2,4^3\}$ and we set $\lambda + (\gamma_I/n^2) = 0$, the variance of the Gaussian kernel is set by the method introduced in [31].
- *Local learning regularization* (*LLReg*). The implementation of this algorithm is the same as in [27], in which we also adopt the mutual neighborhood with its size search from {5,10,50}. The regularization parameter of the local classifier is searched from $\{4^{-3},4^{-2},4^{-1}, 1,4^1,4^2,4^3\}$, and the tradeoff parameter between the loss and local regularization term is also searched from $\{4^{-3},4^{-2},4^{-1}, 4^0,4^1,4^2,4^3\}$.
- *Laplacian regularized least squares* (*LapRLS*). The implementation of the algorithm is the same as in [4], in which the variance of the Gaussian kernel is also set by the method in [31], and the extrinsic and intrinsic regularization parameters are searched from $\{4^{-3},4^{-2},4^{-1}, 4^0,4^1,4^2,4^3\}$. The linear kernel is adopted.
- *Learning with local and global consistency* (*LLGC*). The implementation of the algorithm is the same as in [30], in which the variance of the Gaussian kernel is also by the method in [31], and the regularization parameter is searched from $\{4^{-3},4^{-2}, 4^{-1},4^0,4^1, 4^2,4^3\}$.
- *Gaussian random fields* (*GRF*). The implementation of the algorithm is the same as in [31].
- *Transductive support vectpr machine* (*TSVM*). The implementation is based on [11] using SVMlight.[6] The cost parameter is searched from $\{10^{-4},10^{-3},10^{-2}, 10^{-1},10^0,10^1,10^2,10^3,10^4\}$. Gaussian kernel is adopted with the width searched from $\{2^{-5},2^{-4},\ldots,2^0, 2^1,\ldots,2^5\}$.

[2] http://www.kernel-machines.org/data.html.
[3] http://www1.cs.columbia.edu/CAVE/software/softlib/coil-20.php
[4] http://www.cs.cmu.edu/~WebKB/.
[5] http://www.ics.uci.edu/mlearn/MLRepository.html.
[6] http://svmlight.joachims.org/

- *Support vector machine* (*SVM*). We use *libSVM* [8] to implement the SVM algorithm. The cost parameter is searched from $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3}, 10^{4}\}$. Gaussian kernel is adopted with the width searched from $\{2^{-5}, 2^{-4}, \ldots, 2^{0}, 2^{1}, \ldots, 2^{5}\}$.

4.3. *Experimental results*

The experimental results are shown in Fig. 4. In all the figures, the *x*-axis represents the percentage of randomly labeled points, the *y*-axis is the average classification accuracy over 50 independent runs. From the figures we can observe that

- The *LapRLS* algorithm works very well on the toy and text data sets, but not very well on the image and UCI data sets.
- The *LLGC* and *GRF* algorithm work well on the image data sets, but not very well on other data sets.
- The *LLReg* algorithm works well on the image and text data sets, but not very well on the BCI and toy data sets.
- *SVM* works well when the data sets are not well structured, e.g. the toy, UCI and BCI data sets.
- *LGReg* works very well on almost all the data sets, except for the toy data sets.

To better illustrate the experimental results, we also provide the numerical results of those algorithms on all the data sets with 10% of the points randomly labeled, and the values in Table 4 are the mean classification accuracies and standard deviations of 50 independent runs, from which we can also see the superiority of the *LGReg* algorithm.

Moreover, we also list the computational time of *LGReg* with some other competitors, the results are shown in Table 5, where the time units are seconds averaged over 50 independent runs. All experiments are conducted on a Windows machine with 2.2 GHz Pentium IV processor and 512 MB main memory. From the table we can see *LGReg* is a little more time consuming.

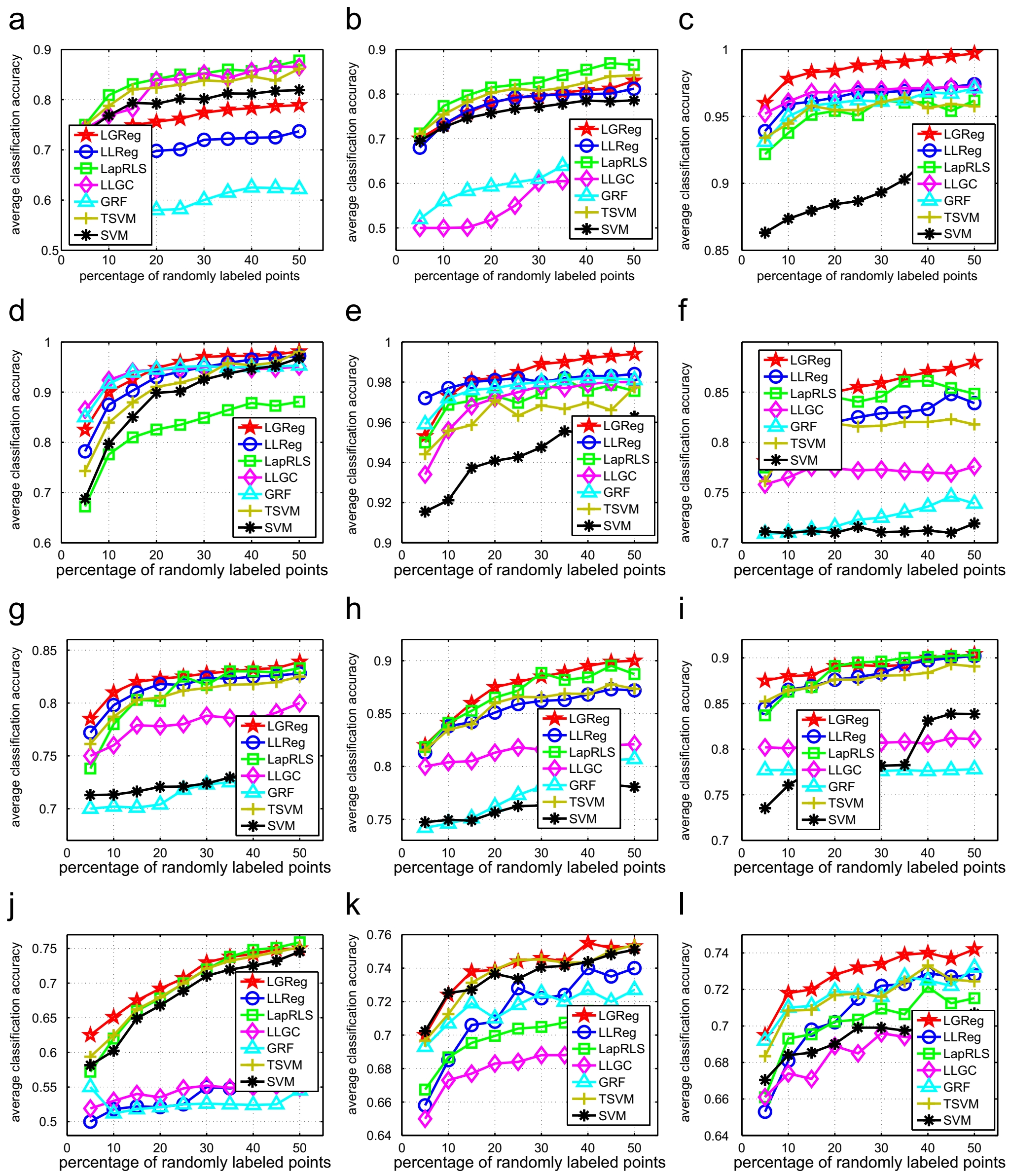

Fig. 4. Experimental results of different algorithms. (a) g241c; (b) g241n; (c) USPS; (d) COIL; (e) digit1; (f) cornell; (g) texas; (h) wisconsin; (i) washington; (j) BCI; (k) diabetes; (l) ionosphere.

Table 4
Experimental results with 10% of the data points randomly labeled.

	SVM	*GRF*	*LLGC*	*LLReg*	*LapRLS*	*LGReg*
g241c	76.03 ± 1.2056	56.34 ± 2.1665	77.13 ± 2.5871	65.31 ± 2.1220	$\mathbf{80.39\pm1.0678}$	72.29 ± 0.1347
g241n	73.67 ± 1.7044	55.06 ± 1.9519	49.75 ± 0.2570	73.25 ± 0.2466	$\mathbf{78.70\pm1.1279}$	73.20 ± 0.5983
USPS	88.56 ± 1.0653	94.87 ± 1.7490	96.19 ± 0.7588	95.79 ± 0.6804	93.25 ± 1.2516	$\mathbf{99.21\pm1.1290}$
COIL	79.98 ± 1.7864	91.23 ± 1.8321	$\mathbf{92.04\pm1.9170}$	86.86 ± 2.2190	78.69 ± 1.2337	89.61 ± 1.2197
digit1	92.04 ± 1.5013	96.95 ± 0.9601	95.49 ± 0.5638	$\mathbf{97.64\pm0.6636}$	96.89 ± 1.0874	97.10 ± 1.0982
cornell	71.13 ± 0.4629	71.43 ± 0.8564	76.30 ± 2.5865	79.46 ± 1.6336	80.34 ± 1.4553	$\mathbf{81.39\pm0.8968}$
texas	71.33 ± 0.5433	70.03 ± 0.8371	75.93 ± 3.6708	79.44 ± 1.7638	77.94 ± 1.3980	$\mathbf{80.75\pm1.2513}$
wisconsin	75.06 ± 0.3786	74.65 ± 0.4979	80.57 ± 1.9062	83.62 ± 1.5191	$\mathbf{84.33\pm1.0113}$	84.05 ± 0.5421
washington	76.15 ± 0.4601	78.26 ± 0.4053	80.23 ± 1.3997	86.37 ± 1.5516	86.20 ± 1.3784	$\mathbf{88.01\pm1.1369}$
BCI	60.04 ± 3.9207	50.49 ± 1.9392	53.07 ± 2.9037	51.56 ± 2.8277	61.52 ± 2.5076	$\mathbf{65.31\pm2.5354}$
diabetes	73.18 ± 1.5671	70.69 ± 2.6321	67.15 ± 1.9766	68.38 ± 2.1772	69.02 ± 1.2036	$\mathbf{72.36\pm1.3223}$
ionosphere	68.87 ± 1.2509	70.21 ± 2.2778	67.31 ± 2.6155	68.15 ± 2.3018	69.28 ± 0.7885	$\mathbf{84.05\pm0.5421}$

Table 5
Computational time comparison.

	SVM	GRF	LLGC	LLReg	LGReg
g241c	2.35	5.63	6.08	9.34	10.87
COIL	4.36	6.05	6.97	9.98	12.05
cornell	7.03	9.54	10.12	12.45	14.16
wisconsin	8.75	10.02	11.30	13.09	15.76
BCI	1.02	1.87	2.15	3.06	5.07
diabetes	1.21	1.90	2.09	3.17	5.20
ionosphere	0.75	1.28	1.96	2.69	4.18

5. Conclusions

In this paper we proposed a general learning framework based on local and global regularization. We showed that many existing learning algorithms can be derived from our framework. Finally experiments are conducted to demonstrate the effectiveness of our method.

Appendix A. Proof of Lemma 1

To prove $\tilde{\mathbf{G}}_i$ is an orthogonal projection matrix, we need to justify that it satisfies the following three properties:

(1) $\tilde{\mathbf{G}}$ is idempotent;
(2) $\tilde{\mathbf{G}}$ is Hermitian;
(3) $\tilde{\mathbf{G}}$ is orthogonal to $\mathbf{I}-\tilde{\mathbf{G}}$.

Clearly,

(1) $\tilde{\mathbf{G}}\tilde{\mathbf{G}}=(\mathbf{I}-\mathbf{G}_i(\mathbf{G}_i^T\mathbf{G}_i)^{-1}\mathbf{G}_i^T)=\tilde{\mathbf{G}}$,
(2) $(\tilde{\mathbf{G}})^T=\tilde{\mathbf{G}}$,
(3) $\tilde{\mathbf{G}}^T(\mathbf{I}-\tilde{\mathbf{G}})=\tilde{\mathbf{G}}-\tilde{\mathbf{G}}\tilde{\mathbf{G}}=\tilde{\mathbf{G}}-\tilde{\mathbf{G}}=\mathbf{0}$.

Therefore $\tilde{\mathbf{G}}$ is an orthogonal projection matrix. $\square$

Appendix B. Proof of Theorem 1

Let $\mathbf{X}_i=[\mathbf{x}_{i_1}-\mathbf{x}_i,\mathbf{x}_{i_2}-\mathbf{x}_i,\ldots,\mathbf{x}_{i_{n_i}}-\mathbf{x}_i]$, $\mathbf{1}=[1,1,\ldots,1]^T\in\mathbb{R}^{n_i\times1}$, then

$$\mathbf{G}_i=\begin{bmatrix}\mathbf{X}_i^T & \mathbf{1}\\ \frac{\sqrt{\gamma_A}}{n_i}\mathbf{I}_d & \mathbf{0}\end{bmatrix}.$$

Then

$$(\mathbf{G}_i^T\mathbf{G}_i)^{-1}=\begin{bmatrix}\mathbf{X}_i\mathbf{X}_i^T+\frac{\gamma_A}{n_i^2}\mathbf{I}_d & \mathbf{X}_i\mathbf{1}\\ \mathbf{1}^T\mathbf{X}_i^T & n_i\end{bmatrix}^{-1}$$
$$=\begin{bmatrix}\mathbf{H}_i^{-1}+\frac{\mathbf{H}_i^{-1}\mathbf{X}_i\mathbf{1}\mathbf{1}^T\mathbf{X}_i^T\mathbf{H}_i^{-1}}{n_i-c} & -\frac{\mathbf{H}_i^{-1}\mathbf{X}_i\mathbf{1}}{n_i-c}\\ -\frac{\mathbf{1}^T\mathbf{X}_i^T\mathbf{H}_i^{-1}}{n_i-c} & \frac{1}{n_i-c}\end{bmatrix},$$

where $\mathbf{H}_i=\mathbf{X}_i\mathbf{X}_i^T+\frac{\gamma_A}{n_i^2}\mathbf{I}_d, c=\mathbf{1}^T\mathbf{X}_i^T\mathbf{H}_i^{-1}\mathbf{X}_i\mathbf{1}$. Then

$$\mathbf{G}_i(\mathbf{G}_i^T\mathbf{G}_i)^{-1}\mathbf{G}_i^T=\begin{bmatrix}\mathbf{X}_i & \frac{\sqrt{\gamma_A}}{n_i}\mathbf{I}_d\\ \mathbf{1}^T & \mathbf{0}^T\end{bmatrix}(\mathbf{G}_i^T\mathbf{G}_i)^{-1}\begin{bmatrix}\mathbf{X}_i & \frac{\sqrt{\gamma_A}}{n_i}\mathbf{I}_d\\ \mathbf{1}^T & \mathbf{0}^T\end{bmatrix},$$

and we can derive that

$$\mathbf{A}_i=\mathbf{I}_{n_i}-\left(\mathbf{X}_i^T\mathbf{H}_i^{-1}\mathbf{X}_i+\frac{\mathbf{X}_i^T\mathbf{H}_i^{-1}\mathbf{X}_i\mathbf{1}\mathbf{1}^T\mathbf{X}_i^T\mathbf{H}_i^{-1}\mathbf{X}_i}{n_i-c}-\frac{\mathbf{X}_i^T\mathbf{H}_i^{-1}\mathbf{X}_i\mathbf{1}\mathbf{1}^T}{n_i-c}-\frac{\mathbf{1}\mathbf{1}^T\mathbf{X}_i^T\mathbf{H}_i^{-1}\mathbf{X}_i}{n_i-c}+\frac{\mathbf{1}\mathbf{1}^T}{n_i-c}\right),$$

where $\mathbf{I}_{n_i}$ is the $n_i\times n_i$ identity matrix. So

$$\mathbf{A}_i\mathbf{1}=\mathbf{1}-\left(\mathbf{X}_i^T\mathbf{H}_i^{-1}\mathbf{X}_i\mathbf{1}+\frac{\mathbf{X}_i^T\mathbf{H}_i^{-1}\mathbf{X}_i\mathbf{1}c}{n_i-c}-\frac{\mathbf{X}_i^T\mathbf{H}_i^{-1}\mathbf{X}_i\mathbf{1}n_i}{n_i-c}-\frac{\mathbf{1}c}{n_i-c}+\frac{\mathbf{1}n_i}{n_i-c}\right)=\mathbf{0}$$

which proves the lemma. $\square$

Appendix C. Proof of Theorem 2

According to their definitions, let

$$\mathbf{p}=\mathbf{H}_i^{-1}\mathbf{X}_i\mathbf{r}=\left(\mathbf{X}_i\mathbf{X}_i^T+\frac{\gamma_A}{n_i^2}\mathbf{I}_d\right)^{-1}\mathbf{X}_i\mathbf{r}.$$

Then

$$\left(\mathbf{X}_i\mathbf{X}_i^T+\frac{\gamma_A}{n_i^2}\mathbf{I}_d\right)\mathbf{p}=\mathbf{X}_i\mathbf{X}_i^T\mathbf{p}+\frac{\gamma_A}{n_i^2}\mathbf{p}=\mathbf{X}_i\mathbf{r}$$
$$\Longrightarrow\mathbf{p}=\frac{n_i^2}{\gamma_A}\mathbf{X}_i(\mathbf{r}-\mathbf{X}_i^T\mathbf{p}).$$

Let $\mathbf{q} = \frac{n_i^2}{\gamma_A}(\mathbf{r} - \mathbf{X}_i^T\mathbf{p})$, then

$$\begin{aligned}
&\mathbf{p} = \mathbf{X}_i\mathbf{q}\\
&\quad\Longrightarrow \frac{\gamma_A}{n_i^2}\mathbf{q} = \mathbf{r} - \mathbf{X}_i^T\mathbf{p} = \mathbf{r} - \mathbf{X}_i^T\mathbf{X}_i\mathbf{q}\\
&\quad\Longrightarrow \mathbf{r} = \left(\mathbf{X}_i^T\mathbf{X}_i + \frac{\gamma_A}{n_i^2}\mathbf{I}_{n_i}\right)\mathbf{q}\\
&\quad\Longrightarrow \mathbf{q} = \left(\mathbf{X}_i^T\mathbf{X}_i + \frac{\gamma_A}{n_i^2}\mathbf{I}_{n_i}\right)^{-1}\mathbf{r}\\
&\quad\Longrightarrow \mathbf{p} = \mathbf{X}_i\mathbf{q} = \mathbf{X}_i\left(\mathbf{X}_i^T\mathbf{X}_i + \frac{\gamma_A}{n_i^2}\mathbf{I}_{n_i}\right)^{-1}\mathbf{r}.
\end{aligned}$$

That is $\mathbf{H}_i^{-1}\mathbf{X}_i\mathbf{r} = \mathbf{X}_i\overline{\mathbf{H}}_i^{-1}\mathbf{r}$. □

References

[1] M. Belkin, I. Matveeva, P. Niyogi, Regularization and semi-supervised learning on large graphs, in: Proceedings of the 17th Conference on Learning Theory (COLT), 2004, pp. 624–638.

[2] M. Belkin, P. Niyogi, Towards a theoretical foundation for Laplacian-based manifold methods, Journal of Computer and System Sciences 74 (8) (2008) 1289–1308.

[3] M. Belkin, P. Niyogi, Laplacian eigenmaps for dimensionality reduction and data representation, Neural Computation 15 (6) (2003) 1373–1396.

[4] M. Belkin, P. Niyogi, V. Sindhwani, Manifold regularization: a geometric framework for learning from labeled and unlabeled examples, Journal of Machine Learning Research 7 (November) (2006) 2399–2434.

[5] L. Bottou, V. Vapnik, Local learning algorithms, Neural Computation 4 (1992) 888–900.

[6] O. Chapelle, M. Chi, A. Zien, A continuation method for semi-supervised SVMs, in: Proceedings of the 23rd International Conference on Machine Learning, 2006, pp. 185–192.

[7] O. Chapelle, B. Schölkopf, A. Zien, Semi-Supervised Learning, MIT Press, Cambridge, MA, USA, 2006, p. 508.

[8] R.-E. Fan, P.-H. Chen, C.-J. Lin, Working set selection using second order information for training SVM, Journal of Machine Learning Research 6 (2005) 1889–1918.

[9] G.H. Gloub, C.F. Vanloan, Matrix Computations, Johns Hopking UP, Baltimore, 1983.

[10] M. Hein, J.Y. Audibert, von U. Luxburg, From graphs to manifolds-weak and strong pointwise consistency of graph Laplacians, in: Proceedings of the 18th Annual Conference on Learning Theory (COLT), 2005, pp. 470–485.

[11] T. Joachims, Transductive inference for text classification using support vector machines, in: Proceedings of the International Conference on Machine Learning (ICML), 1999, pp. 200–209.

[12] I.T. Jolliffe, Principal Component Analysis, second ed., Springer, NY, 2002.

[13] A. Kapoor, Y. Qi, H. Ahn, R.W. Picard, Hyperparameter and kernel learning for graph based semi-supervised classification, In Advances in Neural Information Processing Systems (NIPS) 18 (2006) 627–634.

[14] T.N. Lal, M. Schröder, T. Hinterberger, J. Weston, M. Bogdan, N. Birbaumer, B. Schölkopf, Support vector channel selection in BCI, IEEE Transactions on Biomedical Engineering 51 (6) (2004) 1003–1010.

[15] A. Levin, D. Lischinski, Y. Weiss, A closed form solution to natural image matting, IEEE Transactions Pattern Analysis and Machine Intelligence 30 (2) (2007) 228–242.

[16] W. Liu, D. Tao, J. Liu, Transductive component analysis, in: Proceedings of the Eighth IEEE International Conference on Data Mining, 2008, pp. 433–442.

[17] S. Roweis, S. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (5500) (2000) 2323–2326.

[18] B. Schölkopf, A. Smola, Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond, The MIT Press, Cambridge, MA, 2002.

[19] A.J. Smola, P.L. Bartlett, B. Scholkopf, D. Schuurmans, Advances in Large Margin Classifiers, The MIT Press, Cambridge, MA, 2000.

[20] C.R. Rao, S.K. Mitra, Generalized Inverse of Matrices, Wiley, New York, 1971.

[21] V.N. Vapnik, The Nature of Statistical Learning Theory, Springer-Verlag, Berlin, 1995.

[22] F. Wang, C. Zhang, T. Li, Regularized clustering for documents, in: Proceedings of The 30th International ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR), Amsterdam, The Netherlands. 2007, pp. 95–102 .

[23] F. Wang, C. Zhang, T. Li, Clustering with local and global regularization, in: Proceedings of the 22nd AAAI Conference on Artificial Intelligence (AAAI), Vancouver, Canada, 2007, pp. 657–662.

[24] F. Wang, S. Wang, C. Zhang, O. Winther, Semi-supervised mean fields, in: Proceedings of the 11th International Conference on Artificial Intelligence and Statistics (AISTATS), 2007, pp. 596–603.

[25] F. Wang, T. Li, C. Zhang, Semi-supervised clustering via matrix factorization. in: Proceedings of the 8th SIAM Conference on Data Mining (SDM), Hyatt Regency Hotel, Atlanta, Georgia, 2008, pp. 1–12.

[26] M. Wu, B. Schölkopf, A local learning approach for clustering, Advances in Neural Information Processing Systems, 2006, pp. 1–8.

[27] M. Wu, B. Schölkopf, Transductive classification via local learning regularization. in: Proceedings of the 11th International Conference on Artificial Intelligence and Statistics (AISTATS), 2007, pp. 624–631.

[28] D. Zhang, F. Wang, C. Zhang, T. Li, Multi-view local learning, in: Proceedings of The 23rd AAAI Conference on Artificial Intelligence (AAAI), Chicago, Illinois, USA, July 13–17, 2008, pp. 752–757.

[29] T. Zhang, D. Tao, X. Li, J. Yang, Patch alignment for dimensionality reduction, IEEE Transactions on Knowledge and Data Engineering 21 (9) (2009) 1299–1313.

[30] D. Zhou, O. Bousquet, T.N. Lal, J. Weston, B. Schölkopf, Learning with local and global consistency, Advances in Neural Information Processing Systems 16 (2004) 321–328.

[31] X. Zhu, Z. Ghahramani, Z. Lafferty, Semi-supervised learning using Gaussian fields and harmonic functions, in: Proceedings of the 20th International Conference on Machine Learning (ICML), 2003, pp. 912–919.

[32] X. Zhu, A. Goldberg, Kernel regression with order preferences, in: Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence (AAAI), 2007.

About the Author—FEI WANG received his Ph.D. degree from Tsinghua University, Beijing, China in 2008. After that, he came to Florida International University as a postdoctorate research associate till August, 2009. He will join Department of Statistics, Cornell University as a postdoc in September 2009.