Transductive Learning on Adaptive Graphs

Yan-Ming Zhang,† Yu Zhang,‡ Dit-Yan Yeung,‡ Cheng-Lin Liu,† Xinwen Hou†
† National Laboratory of Pattern Recognition,
Institute of Automation, Chinese Academy of Sciences, Beijing 100090, China
{ymzhang, liucl,xwhou} @nlpr.ia.ac.cn
‡Department of Computer Science and Engineering,
Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong
{zhangyu,dyyeung} @cse.ust.hk

Abstract

Graph-based semi-supervised learning methods are based on some smoothness assumption about the data. As a discrete approximation of the data manifold, the graph plays a crucial role in the success of such graph-based methods. In most existing methods, graph construction makes use of a predefined weighting function without utilizing label information even when it is available. In this work, by incorporating label information, we seek to enhance the performance of graph-based semi-supervised learning by learning the graph and label inference simultaneously. In particular, we consider a particular setting of semi-supervised learning called transductive learning. Using the LogDet divergence to define the objective function, we propose an iterative algorithm to solve the optimization problem which has closed-form solution in each step. We perform experiments on both synthetic and real data to demonstrate improvement in the graph and in terms of classification accuracy.

Introduction

In many machine learning applications, labeled data are scarce because labeling data is both time-consuming and expensive. However, unlabeled data are very easy to collect in many applications such as text categorization and image classification. This has motivated machine learning researchers to develop learning methods that can exploit both labeled and unlabeled data during model learning. Such a learning paradigm developed over the past decade or so is referred to as semi-supervised learning (Chapelle, Schölkopf, and Zien 2006; Zhu 2008).

In this paper, we consider a particular setting of semi-supervised learning called transductive learning in which the unlabeled test data are also known and available for use before model training begins. Specifically, we are given l labeled data points $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l)$ and u unlabeled data points $\mathbf{x}_{l+1}, \ldots, \mathbf{x}_{l+u}$, where $\mathbf{x}_i \in \mathbb{R}^d$, $1 \leq i \leq l+u$, is the input of a labeled or unlabeled data point and $y_i \in \{-1, 1\}$, $1 \leq i \leq l$, is the class label of a labeled data point. The goal is to predict the labels of unlabeled data.

Among the most popular transductive learning methods are graph-based methods (Zhu, Ghahramani, and Lafferty 2003; Zhou et al. 2004; Belkin, Niyogi, and Sindhwani 2006).[1] They are based on some smoothness assumption (cluster assumption or manifold assumption) which implies that data points residing in the same high-density region should have the same label. By using a graph as a discrete approximation of the data manifold, graph-based transductive learning methods learn a classification of the data that satisfies two criteria. First, it should have low classification error on the labeled data. Second, it should be smooth with respect to the graph. We will give a brief review of these methods in the next section.

As reported by many researchers (Wang and Zhang 2008; Jebara, Wang, and Chang 2009; Maier, Von Luxburg, and Hein 2009), graph construction plays a crucial role in the success of graph-based transductive learning methods. Typically, to construct a graph, we define neighborhood based on either the k nearest neighbors (kNN) or the ϵ-ball method to determine the connectivity of the graph. In general, the edges are associated with weights determined by some weighting function. For a comprehensive review of graph construction methods, readers are referred to (Jebara, Wang, and Chang 2009). These methods typically have some hyperparameters whose values have to be chosen in advance and fixed throughout the entire learning process. Moreover, the same hyperparameter value is used for all data points in the data set. As a result, some undesirable edges and weights are unavoidable.

Some researchers have proposed methods to alleviate this problem. For example, given a connectivity graph, Wang and Zhang (2008) proposed a method to learn the weights of the edges instead of using a predefined weighting function to compute the weights. Jebara, Wang, and Chang (2009) proposed a method for constructing a connectivity graph in which each node has exactly k edges. They showed that such graphs give better performance than kNN graphs. Daitch, Kelner, and Spielman (2009) proposed a method to simultaneously learn the connectivity graph and its weights using a quadratic loss function. However, to the best of our knowl-

[1] The methods proposed by Belkin, Niyogi, and Sindhwani (2006) can also give prediction on unseen data and hence are more general than the other methods.

edge, all these methods perform graph construction and label inference in two separate stages. Moreover, their graph construction procedures are unsupervised in nature in the sense that label information is not utilized even when it is available.

In this work, we seek to enhance the performance of graph-based transductive learning by using an adaptive graph that is learned automatically while learning to make label inference for the nodes corresponding to unlabeled data. The motivation behind our work is to improve the graph by incorporating label information from the labeled data such that the points with the same label become more similar in the new graph and those with different labels become less similar. Although this does not necessarily make the new graph a better discrete approximation of the data manifold, it will likely improve the subsequent classification accuracy.

Specifically, we propose a novel method, called *transductive learning on adaptive graphs* (TLAG), which simultaneously constructs the graph and infers the labels of the unlabeled data points. An iterative algorithm is proposed to solve the optimization problem. In each iteration, the following two operations are performed. First, based on the current graph and the predicted values of the data points, a new graph is learned by incorporating the label information. Next, based on the new graph, the predicted values are updated using an ordinary graph-based tranductive learning method. Using a positive semidefinite matrix to represent each graph, we use the LogDet divergence (Bregman 1967) as a measure of the dissimilarity between two graphs.[2] With this formulation, we can obtain closed-form solutions for both operations performed in each iteration of the algorithm, making the algorithm very easy to implement. As we will see later in the experiments, this iterative algorithm which learns the graph and label inference simultaneously often leads to improvement in the graph and in terms of classification accuracy.

The rest of this paper is organized as follows. In section 2, we give a brief review of graph-based transductive learning methods. In section 3, we present the formulation of our TLAG method and the iterative algorithm for solving the optimization problem. Some experimental results are presented in section 4, followed by conclusion of the paper in section 5.

Notation

We first introduce some symbols and notations that will be used subsequently in the paper. We use boldface uppercase letters such as $\mathbf{A}$ to denote matrices, boldface lowercase letters such as $\mathbf{a}$ to denote vectors, and normal lowercase letters such as a to denote scalars. We use l and u to denote the numbers of labeled and unlabeled data points, respectively, and hence $n = l + u$ is the size of the whole data set. Also, $\mathbf{I}$ denotes the identity matrix, $\mathbf{A}_{ij}$ refers to the (i, j)th element of matrix $\mathbf{A}$, $\det(\mathbf{A})$ denotes the determinant of $\mathbf{A}$, $\mathrm{tr}(\mathbf{A})$ denotes the trace of $\mathbf{A}$, and $\mathbf{A} \succeq 0$ means that $\mathbf{A}$ is a positive semidefinite matrix.

[2]The LogDet divergence is a type of matrix divergence which measures the dissimilarity between two matrices.

A Brief Review of Graph-based Methods

Several existing graph-based transductive learning algorithms can be viewed as solving a quadratic optimization problem. We adopt the problem formulation in (Wu and Schölkopf 2007):

$$\min_{\mathbf{f}\in\mathbb{R}^n} \mathbf{f}^T\mathbf{L}\mathbf{f} + (\mathbf{f}-\mathbf{y})^T\mathbf{C}(\mathbf{f}-\mathbf{y}), \tag{1}$$

where the vector $\mathbf{f} = (f_1, \ldots, f_n)^T \in \mathbb{R}^n$ is the prediction of labels, $\mathbf{y} = (y_1, \ldots, y_l, 0, \ldots, 0)^T \in \mathbb{R}^n$, $\mathbf{L} \in \mathbb{R}^{n\times n}$ is the regularization matrix, and $\mathbf{C} \in \mathbb{R}^{n\times n}$ is a diagonal matrix with the ith diagonal element c_i set as: $c_i = \alpha_l > 0$ for $1 \le i \le l$ and $c_i = \alpha_u \ge 0$ for $l+1 \le i \le l+u$, where α_l and α_u are two parameters.

The second term of the objective function in (1) uses a quadratic loss function to measure the fitting error of the prediction $\mathbf{f}$. It constrains $\mathbf{f}$ to be close to $\mathbf{y}$. The first term of the objective function evaluates how smooth the prediction $\mathbf{f}$ is with respect to the data manifold which is represented by the regularization matrix $\mathbf{L}$. It reflects the prior knowledge that a good prediction should have low variance along the data manifold.

A very popular choice for $\mathbf{L}$ is the so-called graph Laplacian (Chung 1997) which is defined as

$$\mathbf{L} = \mathbf{D} - \mathbf{W}, \tag{2}$$

where $\mathbf{W} \in \mathbb{R}^{n\times n}$ is the adjacency matrix of the graph and its entries are calculated as

$$w_{ij} = \exp\Big(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\Big), \tag{3}$$

where σ is the kernel width parameter. The matrix $\mathbf{D} \in \mathbb{R}^{n\times n}$ is diagonal with the ith diagonal element d_i defined as $d_i = \sum_{j=1}^{n} w_{ij}$. Another widely used regularization matrix is the normalized graph Laplacian which is defined as

$$\mathbf{L} = \mathbf{D}^{-\frac{1}{2}}(\mathbf{D}-\mathbf{W})\mathbf{D}^{-\frac{1}{2}} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}. \tag{4}$$

It is easy to show that problem (1) has the following closed-form solution:

$$\mathbf{f} = (\mathbf{L}+\mathbf{C})^{-1}\mathbf{C}\mathbf{y}. \tag{5}$$

Classification can be performed by considering the signs of the predicted values of the unlabeled data points:

$$y_i = \mathrm{sgn}(f_i) \quad l+1 \le i \le l+u. \tag{6}$$

Note that problem (1) represents several famous graph-based transductive learning methods. Gaussian random field (GRF) (Zhu, Ghahramani, and Lafferty 2003) uses the graph Laplacian as regularization matrix and sets $\alpha_l = \infty$ and $\alpha_u = 0$. That is, f_i must be strictly equal to y_i for $1 \le i \le l$ and there is no constraint on the unlabeled data. Laplacian regularized least squares (LapRLS) (Belkin, Niyogi, and Sindhwani 2006) also uses the graph Laplacian as regularization matrix and sets α_l to a finite number and $\alpha_u = 0$. This imposes a soft constraint on the labeled data

but no constraint on the unlabeled data. Learning with local and global consistency (LLGC) (Zhou et al. 2004) uses the normalized graph Laplacian and sets $0 < \alpha_l = \alpha_u < \infty$. It imposes a soft constraint on the labeled data and also requires the predicted values of the unlabeled data points to be close to zero.

Our TLAG Method

Objective Function

As discussed above, our approach seeks to learn the graph and label inference simultaneously. The learning problem can be formulated as the following optimization problem:

$$\min_{\mathbf{f}\in\mathbb{R}^n, \tilde{\mathbf{L}}\succeq 0} h(\mathbf{f}, \tilde{\mathbf{L}}) = \mathbf{f}^T\tilde{\mathbf{L}}\mathbf{f} + (\mathbf{f}-\mathbf{y})^T\mathbf{C}(\mathbf{f}-\mathbf{y}) + \beta D(\tilde{\mathbf{L}}, \mathbf{L}), \quad (7)$$

where $\mathbf{f}$, $\mathbf{y}$, $\mathbf{L}$ and $\mathbf{C}$ are the same as those defined in (1), $\tilde{\mathbf{L}}$ denotes the (adaptive) graph we want to learn, $D(\mathbf{A}, \mathbf{B})$ is a measure of the dissimilarity between two matrices, and β is a regularization parameter. The first two terms of the objective function are similar to those in (1), except that instead of the original graph Laplacian $\mathbf{L}$ we use a learned matrix $\tilde{\mathbf{L}}$ to infer the labels of the data points. The last term constrains the new graph to be similar to the original graph. We only require the new graph $\tilde{\mathbf{L}}$ to be positive semidefinite, but there is no guarantee that it is a graph Laplacian. We still call it a graph in the sense that $-\tilde{\mathbf{L}}_{ij}$ can be viewed as the weight on the edge between points i and j.

In particular, we use the LogDet divergence (Bregman 1967; Kulis, Sustik, and Dhillon 2009) as the dissimilarity measure between two matrices. The LogDet divergence can be defined as follows:

$$D_{ld}(\mathbf{A}, \mathbf{B}) = \mathrm{tr}(\mathbf{A}\mathbf{B}^{-1}) - \log\det(\mathbf{A}\mathbf{B}^{-1}) - n. \quad (8)$$

There are two main reasons for using the LogDet divergence in this work. First, as we will show in the next section, this measure leads to a closed-form solution for $\tilde{\mathbf{L}}$ that naturally meets the positive semidefinite constraint. By employing the Woodbury identity, the computational requirement for updating $\tilde{\mathbf{L}}$ is rather low.

Second, the LogDet divergence is one type of Bregman divergence which is defined as:

$$D_\phi(\mathbf{A}, \mathbf{B}) = \phi(\mathbf{A}) - \phi(\mathbf{B}) - \mathrm{tr}((\bigtriangledown\phi(\mathbf{B}))^T(\mathbf{A}-\mathbf{B})), \quad (9)$$

where $\phi : \mathbb{R}^{n\times n} \to \mathbb{R}$ is a strictly convex, differentiable function. For the LogDet divergence $D_{ld}(\mathbf{A}, \mathbf{B})$, the Burg entropy of the eigenvalues is used as ϕ, that is, $\phi(\mathbf{A}) = \sum_i \log\frac{1}{\lambda_i}$ where λ_i is an eigenvalue of $\mathbf{A}$. Given matrix $\mathbf{B}$, to keep $D_{ld}(\mathbf{A}, \mathbf{B})$ small, $\mathbf{A}$ tends to copy those small eigenvalues of $\mathbf{B}$ because even a small difference in those λ_i will result in a big divergence $D_\phi(\mathbf{A}, \mathbf{B})$. This is a desirable property for graph learning because we are more concerned about those eigenvectors corresponding to small eigenvalues since they are smooth with respect to the data manifold (Von Luxburg 2007).

As in LapRLS, we impose a soft constraint on the labeled data but no constraint on the unlabeled data. Thus, the objective function can be written as:

$$h(\mathbf{f}, \tilde{\mathbf{L}}) = \begin{bmatrix}\mathbf{f}_l^T & \mathbf{f}_u^T\end{bmatrix}\begin{bmatrix}\tilde{\mathbf{L}}_{ll} & \tilde{\mathbf{L}}_{lu}^T \\ \tilde{\mathbf{L}}_{lu} & \tilde{\mathbf{L}}_{uu}\end{bmatrix}\begin{bmatrix}\mathbf{f}_l \\ \mathbf{f}_u\end{bmatrix} + \alpha\|\mathbf{f}_l - \mathbf{y}_l\|^2 + \beta D_{ld}(\tilde{\mathbf{L}}, \mathbf{L}), \quad (10)$$

where $\mathbf{f} = \begin{bmatrix}\mathbf{f}_l \\ \mathbf{f}_u\end{bmatrix}$, $\mathbf{f}_l$ is the prediction on the labeled data, $\mathbf{f}_u$ is the prediction on the unlabeled data, and $\begin{bmatrix}\tilde{\mathbf{L}}_{ll} & \tilde{\mathbf{L}}_{lu}^T \\ \tilde{\mathbf{L}}_{lu} & \tilde{\mathbf{L}}_{uu}\end{bmatrix}$ is the block matrix or partitioned matrix of $\tilde{\mathbf{L}}$.

Convexity of $h(\mathbf{f}, \tilde{\mathbf{L}})$

In the following, we show that the objective function $h(\mathbf{f}, \tilde{\mathbf{L}})$ in (7) is convex with respect to (w.r.t.) $\mathbf{f}$ and is also convex w.r.t. $\tilde{\mathbf{L}}$.

Proposition: $h(\mathbf{f}, \tilde{\mathbf{L}})$ in (7) is convex w.r.t. $\mathbf{f}$.

Proof: Because $\frac{\partial^2 h(\mathbf{f}, \tilde{\mathbf{L}})}{\partial\mathbf{f}\partial\mathbf{f}^T} = 2(\tilde{\mathbf{L}} + \mathbf{C})$ and $\tilde{\mathbf{L}} \succeq 0, \mathbf{C} \succeq 0$, so $\frac{\partial^2 h(\mathbf{f}, \tilde{\mathbf{L}})}{\partial\mathbf{f}\partial\mathbf{f}^T} \succeq 0$. ■

Proposition: $h(\mathbf{f}, \tilde{\mathbf{L}})$ in (7) is convex w.r.t. $\tilde{\mathbf{L}} \succeq 0$.

Proof: Let $g(\tilde{\mathbf{L}}) = h(\mathbf{f}, \tilde{\mathbf{L}})$, and we have $g(\tilde{\mathbf{L}}) = \mathrm{tr}(\tilde{\mathbf{L}}\mathbf{f}\mathbf{f}^T) + \beta\mathrm{tr}(\tilde{\mathbf{L}}\mathbf{L}^{-1}) - \beta\log\det(\tilde{\mathbf{L}}) + M$ where M is a constant independent of $\tilde{\mathbf{L}}$. Obviously, the first two terms of $g(\tilde{\mathbf{L}})$ are convex w.r.t. $\tilde{\mathbf{L}}$. One can check the convexity of $-\log\det(\tilde{\mathbf{L}})$ by checking the convexity of the single-variable function $f(t) = -\log\det(\tilde{\mathbf{L}} + t\mathbf{A})$. We note that $f(t) = -\log\det(\tilde{\mathbf{L}}) - \log\det(\mathbf{I} + t\tilde{\mathbf{L}}^{-\frac{1}{2}}\mathbf{A}\tilde{\mathbf{L}}^{-\frac{1}{2}}) = -\log\det(\tilde{\mathbf{L}}) - \sum_{i=1}^m \log(1 + t\lambda_i)$ where λ_i are the eigenvalues of $\tilde{\mathbf{L}}^{-\frac{1}{2}}\mathbf{A}\tilde{\mathbf{L}}^{-\frac{1}{2}}$. Since $f(t)$ is convex for any choice of $\tilde{\mathbf{L}} \succeq 0$ and $\mathbf{A}$, so $-\log\det(\tilde{\mathbf{L}})$ is convex w.r.t. $\tilde{\mathbf{L}} \succeq 0$. Thus, $g(\tilde{\mathbf{L}})$ is convex. ■

Extension for Multi-Class Problems

So far we have only considered binary classification problems for simplicity of the problem formulation. However, it is easy to extend TLAG for multi-class classification applications. For a k-class problem, one can use the one-in-k representation to represent the labels of the data points where the label $\mathbf{y} = (y_1, \ldots, y_k)^T \in \mathbb{R}^k$ of $\mathbf{x}$ is an indicator vector with $y_j = 1$ if $\mathbf{x}$ belongs to the jth class and $y_j = 0$ otherwise. The corresponding optimization problem can be formulated as:

$$\min_{\mathbf{F}\in\mathbb{R}^{n\times k}, \tilde{\mathbf{L}}\succeq 0} \mathrm{tr}(\mathbf{F}^T\tilde{\mathbf{L}}\mathbf{F}) + \mathrm{tr}((\mathbf{F}-\mathbf{Y})^T\mathbf{C}(\mathbf{F}-\mathbf{Y})) + \beta D(\tilde{\mathbf{L}}, \mathbf{L}),$$

where $\mathbf{F} \in \mathbb{R}^{n\times k}$ is the prediction matrix we need to learn and the label matrix $\mathbf{Y} \in \mathbb{R}^{n\times k}$ is such that the ith row of $\mathbf{Y}$ is the one-in-k representation of the label for point $\mathbf{x}_i$ if $1 \le i \le l$ and otherwise it is a zero vector. After obtaining the prediction matrix $\mathbf{F}$, we classify each $\mathbf{x}_i$ to the jth class if $\mathbf{F}_{ij}$ is the largest entry in the ith row of $\mathbf{F}$.

The multi-class objective function corresponding to (10)

is:

$$h(\mathbf{F},\tilde{\mathbf{L}}) = \mathrm{tr}(\begin{bmatrix}\mathbf{F}_l^T\mathbf{F}_u^T\end{bmatrix}\begin{bmatrix}\tilde{\mathbf{L}}_{ll} & \tilde{\mathbf{L}}_{lu}^T\\ \tilde{\mathbf{L}}_{lu} & \tilde{\mathbf{L}}_{uu}\end{bmatrix}\begin{bmatrix}\mathbf{F}_l\\ \mathbf{F}_u\end{bmatrix}) + \alpha\mathrm{tr}((\mathbf{F}_l-\mathbf{Y}_l)(\mathbf{F}_l-\mathbf{Y}_l)^T) + \beta D_{ld}(\tilde{\mathbf{L}},\mathbf{L}), \quad (11)$$

where $\mathbf{F} = \begin{bmatrix}\mathbf{F}_l\\ \mathbf{F}_u\end{bmatrix}$, $\mathbf{F}_l$ is the prediction on the labeled data, $\mathbf{F}_u$ is the prediction on the unlabeled data, and $\mathbf{Y}_l$ is the first l rows of $\mathbf{Y}$.

Algorithm

We propose an iterative algorithm to minimize the objective function in (10). First, we initialize $\mathbf{f}$ by setting $\mathbf{f}_i = y_i$ for $i = 1,\ldots,l$ and $\mathbf{f}_i = 0$ for $i = l+1,\ldots,n$. Then, in each iteration, we first fix the prediction $\mathbf{f}$ and optimize w.r.t. the graph $\tilde{\mathbf{L}}$, then fix $\tilde{\mathbf{L}}$ and optimize w.r.t. $\mathbf{f}$. As we will show below, both steps have closed-form solutions and so the algorithm is very easy to implement. Since the value of the objective function decreases in each step, the algorithm is guaranteed to converge to a local minimum.

For fixed $\mathbf{f}$ We compute the gradient of the objective function in (10) w.r.t. $\tilde{\mathbf{L}}$ and set it to zero:

$$\frac{\partial h}{\partial \tilde{\mathbf{L}}} = \mathbf{f}\mathbf{f}^T + \beta(\mathbf{L}^{-1} - \tilde{\mathbf{L}}^{-1}) = 0. \quad (12)$$

It follows from the facts that $\frac{\partial \mathrm{tr}(\mathbf{AB})}{\partial \mathbf{A}} = \mathbf{B}^T$ and $\frac{\partial \log|\det(\mathbf{A})|}{\partial \mathbf{A}} = (\mathbf{A}^{-1})^T$.

From (12), we get

$$\tilde{\mathbf{L}} = \big(\frac{1}{\beta}\mathbf{f}\mathbf{f}^T + \mathbf{L}^{-1}\big)^{-1}. \quad (13)$$

Obviously, $\tilde{\mathbf{L}}$ is a positive semidefinite matrix. This update can be understood from the view of graph kernels. According to (Smola and Kondor 2003), the inverse of a graph Laplacian $\mathbf{L}^{-1}$ is known as a graph kernel. The term $\mathbf{f}\mathbf{f}^T$ can be viewed as an output kernel which is similar to a target kernel defined as the outer product of the label vector (Cristianini, Kandola, and Elissee 2001). Thus, the updated kernel $\tilde{\mathbf{L}}^{-1}$ is a linear combination of the graph kernel and the output kernel. Since $\tilde{\mathbf{L}}_{ij}^{-1} = \frac{1}{\beta}\mathbf{f}_i\mathbf{f}_j + \mathbf{L}_{ij}^{-1}$, compared to the original graph, the points i and j in the new graph become more similar if they belong to same class and become less similar if they belong to different classes.

To accelerate the computation, we apply the Woodbury identity to (13) to give the following equation which does not require performing matrix inversion:

$$\tilde{\mathbf{L}} = \mathbf{L} - \frac{1}{\frac{1}{\beta} + \mathbf{f}^T\mathbf{L}\mathbf{f}}\mathbf{L}\mathbf{f}(\mathbf{L}\mathbf{f})^T. \quad (14)$$

For fixed $\tilde{\mathbf{L}}$ We now compute the gradient of the objective function w.r.t. $\mathbf{f}$ and set it to zero:

$$\frac{\partial h}{\partial \mathbf{f}_l} = 2(\tilde{\mathbf{L}}_{ll}\mathbf{f}_l + \tilde{\mathbf{L}}_{lu}\mathbf{f}_u + \alpha(\mathbf{f}_l - \mathbf{y}_l)) = 0 \quad (15)$$

$$\frac{\partial h}{\partial \mathbf{f}_u} = 2(\tilde{\mathbf{L}}_{uu}\mathbf{f}_u + \tilde{\mathbf{L}}_{lu}^T\mathbf{f}_l) = 0. \quad (16)$$

Solving these equations, we get:

$$\mathbf{f}_l = \alpha(\tilde{\mathbf{L}}_{ll} + \alpha\mathbf{I} - \tilde{\mathbf{L}}_{lu}\tilde{\mathbf{L}}_{uu}^{-1}\tilde{\mathbf{L}}_{lu}^T)^{-1}\mathbf{y}_l \quad (17)$$

$$\mathbf{f}_u = -\tilde{\mathbf{L}}_{uu}^{-1}\tilde{\mathbf{L}}_{lu}^T\mathbf{f}_l. \quad (18)$$

With no surprise, the prediction for unlabeled data in (18) is very similar to that for the GRF method in which the prediction is calculated as $\mathbf{f}_u = -\mathbf{L}_{uu}^{-1}\mathbf{L}_{lu}^T\mathbf{y}_l$.

Extending this algorithm for multi-class problems only requires very minimal changes. To compute the prediction matrix $\mathbf{F}$, we simply replace the vector $\mathbf{y}_l$ in (17) by the label matrix $\mathbf{Y}_l$. To update $\tilde{\mathbf{L}}$, one can directly use (13) with $\mathbf{f}\mathbf{f}^T$ replaced by $\mathbf{F}\mathbf{F}^T$, or apply the Woodbury identity to obtain a more efficient equation:

$$\tilde{\mathbf{L}} = \mathbf{L} - \mathbf{L}\mathbf{F}(\beta\mathbf{I} + \mathbf{F}^T\mathbf{L}\mathbf{F})^{-1}(\mathbf{L}\mathbf{F})^T.$$

As the size of $\beta\mathbf{I} + \mathbf{F}^T\mathbf{L}\mathbf{F}$ is just $k \times k$ ($k \ll n$), the complexity of the above step is $\mathcal{O}(kn^2)$ while that of (13) is $\mathcal{O}(n^3)$.

Computational Complexity

Assuming that $l \ll u$, the computational complexity required for updating $\tilde{\mathbf{L}}$ using (14) is $\mathcal{O}(n^2)$ while that for updating $\mathbf{f}$ using (17) and (18) is $\mathcal{O}(u^3)$. Thus, the total complexity of our algorithm is about $\mathcal{O}(u^3)$ which is in the same order as conventional graph-based transductive learning methods. In practice, the running time of our method is about T times that of GRF where T is the number of iterations in our algorithm.

Experiments

In this section, we present some experimental results to demonstrate the effectiveness of TLAG. In all the experiments, we simply fix α to 1, β to 0.01 and stop the algorithm after 30 iterations for the TLAG method as the result obtained is similar to that obtained by cross-validation.

Graph Learning on Two-Moons Data Set

In this experiment, we show that learning the graph by incorporating label information can improve the performance by making the points with the same label more similar.

The experiment is performed on the two-moons data set in which each moon corresponds to one class. Given a training data set, we construct an adjacency matrix $\mathbf{W}$ as in (3). The parameter σ is set to $2^{-2}\sigma_0$ where σ_0 is the average distance between points. Then, equation (4) is applied to construct the normalized graph Laplacian $\mathbf{L}$ which is used as the original graph.

Based on the original graph $\mathbf{L}$, we run the proposed TLAG algorithm to learn the new graph $\tilde{\mathbf{L}}$. After obtaining $\tilde{\mathbf{L}}$, we treat it as a graph Laplacian by regarding $-\tilde{\mathbf{L}}_{ij}$ as the weight between data points i and j for $i \neq j$. Similarly, in the original graph, we take $-\mathbf{L}_{ij}$ as the weight between data points i and j.

We first sample 20 data points from each moon and randomly select five of them as labeled points. In Figure 1, subfigure (a) shows the data set and (b) and (c) show the original

graph $\mathbf{L}$ and the learned graph $\tilde{\mathbf{L}}$ after removing those edges whose weights are less than 0.1. We also try another setting with 40 data points from each moon, of which five are labeled points. Subfigures (d)–(f) of Figure 1 show the data set, the original graph $\mathbf{L}$ and the learned graph $\tilde{\mathbf{L}}$ after removing those edges whose weights are less than 0.05.[3] Note that those edges with weights $-\tilde{\mathbf{L}}_{ij} < 0$ are also removed.

As we can see, due to sparsity of the data points, the upper moon is broken into two separate components in the original graph and hence the similarity between the two components is not characterized appropriately. In the language of random walk (Zhu, Ghahramani, and Lafferty 2003), this makes it difficult to spread the label information through the manifold. On the other hand, the two components are connected in the learned graph and hence they are more similar. In fact, in this new graph, most pairs of labeled data points belonging to the same class are connected by an edge.

Classification on Real Data Sets

We next perform experiments on seven real data sets and compare TLAG with several popular graph-based semi-supervised learning methods and a baseline supervised learning method.

The data sets are from those in (Chapelle, Schölkopf, and Zien 2006) and the UCI data sets (Asuncion and Newman 2007). They are from different application areas including text and image classification. The UCI data sets have been preprocessed such that each feature has zero mean and unit standard deviation. A brief description of the data sets is shown in Table 1.

Table 1: Brief description of the data sets.

Data set	Size	No. of classes	No. of dimensions
Digit1	1500	2	241
USPS	1500	2	241
COIL	1500	6	241
Text	1500	2	11,960
Glass	214	6	10
Ionosphere	351	2	34
Sonar	208	2	60

The experimental setting is as follows. For each data set, we first randomly split it into the labeled and unlabeled sets. Then all the algorithms are trained on the whole data set including the labels of the labeled data. Finally, the classification accuracy on the unlabeled data is recorded. This procedure is repeated 10 times and the average accuracy and standard deviation over these 10 runs are reported.

We compare TLAG with three existing graph-based semi-supervised learning methods, namely, GRF, LLGC and LapRLS, as well as support vector machine (SVM) which serves as a baseline supervised learning method. We apply 5-fold cross-validation on the labeled data to choose the hyperparameters for these methods, with the details given below for each method:

[3]We use different thresholds because of the use of a fully connected normalized graph Laplacian in which the weight between two points becomes smaller as the data set becomes larger.

- GRF: (3) is used as the weighting function to construct the graph, and the hyperparameter σ is chosen from $\sigma_0 \cdot \{2^{-4}, 2^{-3}, 2^{-2}, 2^{-1}, 1, 2\}$ where σ_0 is the average distance between two points in the data set.
- LLGC: σ is chosen as above for GRF. Following the setting in (Zhou et al. 2004), the regularization parameter α is set to 0.99.
- LapRLS: σ is also chosen as above for both GRF and LLGC. The parameters γ_A and γ_I are chosen from $\{10^{-4}, 10^{-2}, 1, 10^2, 10^4\}$. The RBF kernel like (3) is also used as the kernel function with the kernel width parameter σ_k chosen from $\sigma_0 \cdot \{2^{-4}, 2^{-3}, 2^{-2}, 2^{-1}, 1, 2\}$.
- TLAG: σ is also chosen as above. Normalized graph Laplacian is used as the original graph $\mathbf{L}$. The regularization parameters α and β are set to 1 and 0.01, respectively.
- SVM: The linear kernel is used and the regularization parameter C is chosen from $\{10^{-4}, 10^{-2}, 1, 10^2, 10^4\}$.

The classification results for $l = 50$ are reported in Table 2. We can see that TLAG is better than or comparable with other methods on most of the data sets. We note that TLAG does not give very good result on the Text data set when the normalized graph Laplacian is used. When the un-normalized graph Laplacian is used instead, its result is comparable to that of GRF.

Conclusion

In this paper, we have proposed a novel graph-based transductive learning method that not only learns to infer the labels of the unlabeled data points but also learns the graph simultaneously by incorporating label information. We use the LogDet divergence to formulate the optimization problem and propose an iterative algorithm to solve the problem. An attractive feature of the iterative algorithm is that there is closed-form solution in each step for updating the graph or the labels. This frees us from having to use complex optimization methods such as semi-definite programming.

In this work, the learned graph is represented by a positive semidefinite matrix which may lead to negative weights on some edges. In the future, we will consider the possibility of learning a graph Laplacian directly. Moreover, we will also consider various ways to speed up our algorithm.

Acknowledgements

This work is supported by the NSFC Outstanding Youth Fund (No. 60825301) and the General Research Fund (No. 621407) from the Research Grants Council of the Hong Kong Special Administrative Region, China.

References

Asuncion, A., and Newman, D. 2007. UCI machine learning repository.

Belkin, M.; Niyogi, P.; and Sindhwani, V. 2006. Manifold regularization: a geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research* 7:2399–2434.

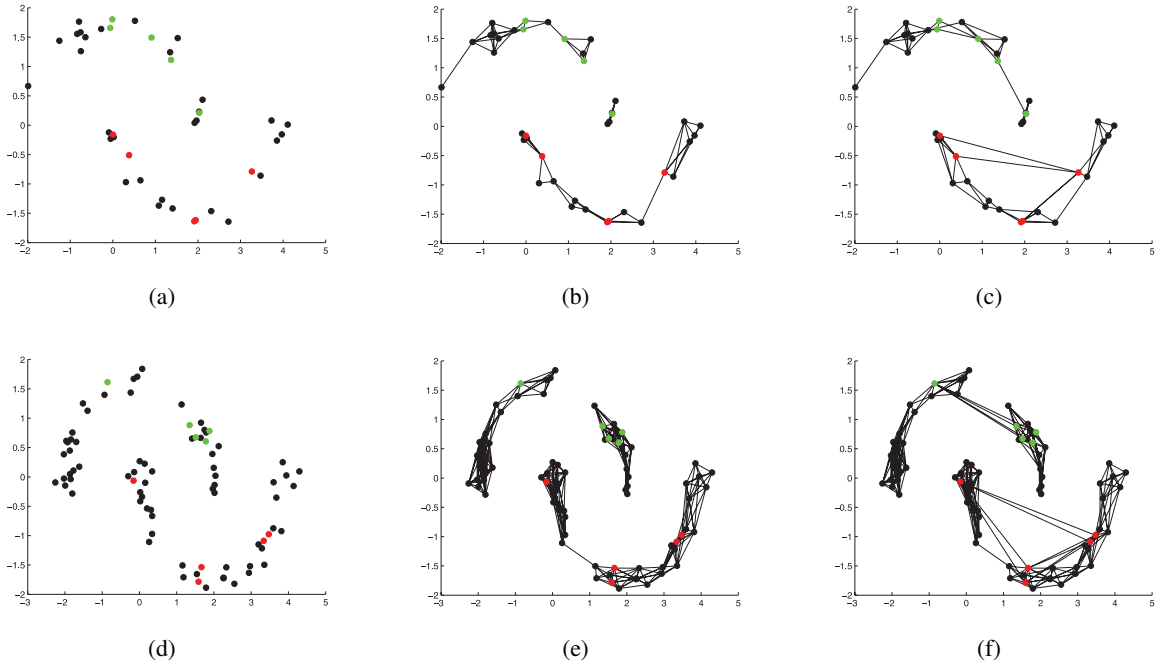

Figure 1: Graph learning: (a) & (d) data set; (b) & (e) original graph $\mathbf{L}$; (c) & (f) learned graph $\tilde{\mathbf{L}}$.

Table 2: Results in average classification rate on the test data and standard deviation for 50 labeled points.

Data set	SVM	GRF	LLGC	LapRLS	TLAG
Digit1	0.8928± 0.0107	0.9620± 0.0116	0.9585± 0.0103	0.9131± 0.0203	**0.9641± 0.0120**
USPS	0.8277± 0.0366	0.9057± 0.0357	0.9429± 0.0272	0.8799± 0.0219	**0.9494± 0.0304**
COIL	0.6310± 0.0423	0.7870± 0.0230	**0.8295± 0.0337**	0.7293± 0.0466	0.8270± 0.0329
Text	0.6742± 0.0520	**0.7047± 0.0820**	0.6397± 0.0306	0.6729± 0.0690	0.6557± 0.0421
Glass	0.5951± 0.0417	0.5676± 0.0799	0.6067± 0.0698	0.6207± 0.0621	**0.6354± 0.0681**
Ionosphere	0.8378± 0.0184	0.8475± 0.0358	0.8814± 0.0253	0.8451± 0.0342	**0.8864± 0.0303**
Sonar	0.7189± 0.0535	0.7494± 0.0483	0.7709± 0.0470	0.7348± 0.0424	**0.7816± 0.0279**

Bregman, L. 1967. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Mathematical Physics* 7(3):200–217.

Chapelle, O.; Schölkopf, B.; and Zien, A. 2006. *Semi-Supervised Learning*. MIT Press.

Chung, F. 1997. *Spectral Graph Theory*. American Mathematical Society.

Cristianini, N.; Kandola, J.; and Elissee, A. 2001. On kernel target alignment. *Advances in Neural Information Processing Systems 14*.

Daitch, S.; Kelner, J.; and Spielman, D. 2009. Fitting a graph to vector data. In *Proceedings of the 26th International Conference on Machine Learning*. ACM New York, NY, USA.

Jebara, T.; Wang, J.; and Chang, S. 2009. Graph construction and b-matching for semi-supervised learning. In *International Conference on Machine Learning*.

Kulis, B.; Sustik, M.; and Dhillon, I. 2009. Low-rank kernel learning with Bregman matrix divergences. *Journal of Machine Learning Research* 10:341–376.

Maier, M.; Von Luxburg, U.; and Hein, M. 2009. Influence of graph construction on graph-based clustering measures. In *Advances in Neural Information Processing Systems 22*.

Smola, A., and Kondor, R. 2003. Kernels and regularization on graphs. In *Learning Theory and Kernel Machines: 16th Annual Conference on Learning Theory and 7th Kernel Workshop*, 144. Springer Verlag.

Von Luxburg, U. 2007. A tutorial on spectral clustering. *Statistics and Computing* 17(4):395–416.

Wang, F., and Zhang, C. 2008. Label propagation through linear neighborhoods. *Transactions on Knowledge and Data Engineering* vol.20, no.1,:55–67.

Wu, M., and Schölkopf, B. 2007. Transductive classification via local learning regularization. In *International Conference on Artificial Intelligence and Statistics*, 381–400.

Zhou, D.; Bousquet, O.; Lal, T.; Weston, J.; and Scholkopf, B. 2004. Learning with local and global consistency. In *Advances in Neural Information Processing Systems 16*, 321–328.

Zhu, X.; Ghahramani, Z.; and Lafferty, J. 2003. Semi-supervised learning using Gaussian fields and harmonic functions. In *Proceedings of the International Conference on Machine Learning*.

Zhu, X. 2008. Semi-supervised learning literature survey. Technical report, Computer Science, University of Wisconsin-Madison.