Transductive Classification via Local Learning Regularization

Mingrui Wu, Bernhard Schölkopf
Max Planck Institute for Biological Cybernetics
72076 Tübingen, Germany
{mingrui.wu, bernhard.schoelkopf}@tuebingen.mpg.de

Abstract

The idea of local learning, classifying a particular point based on its neighbors, has been successfully applied to supervised learning problems. In this paper, we adapt it for *Transductive Classification* (TC) problems. Specifically, we formulate a *Local Learning Regularizer* (LL-Reg) which leads to a solution with the property that the label of each data point can be well predicted based on its neighbors and their labels. For model selection, an efficient way to compute the leave-one-out classification error is provided for the proposed and related algorithms. Experimental results using several benchmark datasets illustrate the effectiveness of the proposed approach.

1 Introduction

In this paper, we mainly consider the binary *Transductive Classification* (TC) problem: We are given l labeled data points $(\mathbf{x}_1, y_1)$, ..., $(\mathbf{x}_l, y_l)$, and u unlabeled points $\mathbf{x}_{l+1}$, ..., $\mathbf{x}_{l+u}$, where $\mathbf{x}_i \in \mathcal{X} \subseteq \mathbb{R}^d$, $1 \leq i \leq l+u$, is the input data, $\mathcal{X}$ is the input space, and $y_i \in \{-1, +1\}$, $1 \leq i \leq l$, is the class label. The goal is to predict the class labels of the *given* unlabeled data points, i.e. $\mathbf{x}_{l+1}$, ..., $\mathbf{x}_{l+u}$.[1]

In many TC algorithms [15, 13, 2, 5], a real valued f_i, $1 \leq i \leq l+u$, is assigned to each data point $\mathbf{x}_i$, based on which the final classification is performed as

$$y_j = \text{sign}(f_j) \qquad l+1 \leq j \leq l+u \tag{1}$$

In this case, how to compute f_i of each $\mathbf{x}_i$ is the main part of a TC algorithm. The key to this task is the prior assumption about the properties that f_i should have over the available data points. For example, graph Laplacian based regularization is a popular TC method [15, 13, 2]. It is based on the "cluster assumption" that if two data points $\mathbf{x}_i$ and $\mathbf{x}_j$ are on the same structure (manifold or cluster), then the values of f_i and f_j should be similar to each other.

In the present paper, we propose a local learning regularization for TC problems. In our approach, the desirable properties of f_i are described by a local learning regularizer, which constrains that the real valued solution f_i of each data point $\mathbf{x}_i$ should be well estimated based on its neighbors and their real value solutions.

The rest of this paper is organized as follows. In section 2, a general quadratic cost function and the Laplacian regularization for TC problems are briefly described. In section 3, the details of our local learning regularizer are presented. Comparison with related approaches is given in section 4. In section 5, we address the model selection issue. In particular, we derive a method to compute the *Leave-One-Out* (LOO) classification error efficiently for the proposed algorithm and some other related approaches. Experimental results are provided in section 6 and we conclude the paper in the last section.

2 A Quadratic Objective Function and Laplacian Regularization for TC

Several TC algorithms [15, 8, 13, 1] can be formulated as or are closely related to the following quadratic optimization problem,

$$\min_{\mathbf{f} \in \mathbb{R}^n} \mathbf{f}^\top \mathbf{R} \mathbf{f} + (\mathbf{f} - \mathbf{y})^\top \mathbf{C} (\mathbf{f} - \mathbf{y}) \tag{2}$$

where $\mathbf{R} \in \mathbb{R}^{n \times n}$ is the regularization matrix, $\mathbf{f} = [f_1, \ldots, f_n]^\top \in \mathbb{R}^n$ is the vector of real valued solution, $\mathbf{y} = [y_1, \ldots, y_l, 0, \ldots, 0]^\top \in \mathbb{R}^n$, and $\mathbf{C} \in \mathbb{R}^{n \times n}$ is a diagonal matrix, its i-th diagonal element c_i is computed as: $c_i = C_l > 0$ for $1 \leq i \leq l$, and $c_i = C_u \geq 0$

[1] Binary TC algorithms can be easily extended to the multi-class case by adopting the one-versus-all approach.

for $l+1 \leq i \leq n$, where C_l and C_u are two parameters.

It can be easily proved that the solution of (2) is

$$\mathbf{f} = (\mathbf{R} + \mathbf{C})^{-1}\mathbf{C}\mathbf{y} \tag{3}$$

In (2), the second term is just similar to the quadratic loss function used in supervised learning, which restricts that f_i should be close to y_i for $1 \leq i \leq l$.

The key part in (2) for solving the TC problem is the first term, the regularizer, which specifies the desirable properties of f_i. Currently the *Laplacian regularizer* (Lap-Reg) is very popular for TC problems. To compute its regularization matrix $\mathbf{L}$, we first build a weighted k-nearest neighbor graph of n nodes,[2] each node of which corresponds to a data point, and $\mathbf{L}$ is computed as [15]:

$$\mathbf{L} = \mathbf{D} - \mathbf{W} \tag{4}$$

where $\mathbf{W} = [w_{ij}] \in \mathbb{R}^{n \times n}$ is the adjacency matrix of this graph and it is calculated as:

$$w_{ij} = \exp(-\frac{1}{\gamma}\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2) \tag{5}$$

if $\mathbf{x}_i$ and $\mathbf{x}_j$ are connected, and 0 otherwise. In (4), $\mathbf{D} \in \mathbb{R}^{n \times n}$ is a diagonal matrix, its i-th diagonal element d_i is defined as $d_i = \sum_{j=1}^{n} w_{ij}$. When $\mathbf{R} = \mathbf{L}$, the first term of (2) can be written as:

$$\mathbf{f}^\top \mathbf{L}\mathbf{f} = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}(f_i - f_j)^2 \tag{6}$$

The *Normalized Laplacian Regularizer* (NLap-Reg) is also widely used in TC algorithms. Its regularization matrix $\mathbf{L_n}$ is calculated as [13]:

$$\mathbf{L_n} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}} \tag{7}$$

where $\mathbf{I}$ is the unit matrix, while $\mathbf{W}$ and $\mathbf{D}$ are the same as in (4). When $\mathbf{R} = \mathbf{L_n}$, the first term of (2) becomes:

$$\mathbf{f}^\top \mathbf{L_n}\mathbf{f} = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}(\frac{f_i}{\sqrt{d_i}} - \frac{f_j}{\sqrt{d_j}})^2 \tag{8}$$

According to (6) and (8), both Lap-Reg and NLap-Reg put a smoothness constraint on $\mathbf{f}$. Namely, the value of f_i should not change too much between nearby points.

The problem (2) represents several typical TC algorithms in literature. For example, in the approach of [15], the Laplacian matrix $\mathbf{L}$ is used for $\mathbf{R}$. And $C_l = \infty$, $C_u = 0$. That is, f_i must strictly equal to y_i for $1 \leq i \leq l$, while there is no constraints on f_i for $l+1 \leq i \leq n$. In [1], $\mathbf{R} = \mathbf{L}$, $C_u = 0$ but $0 < C_l < \infty$, hence the constraint on f_i ($1 \leq i \leq l$) is soft. In [13], the normalized Laplacian matrix $\mathbf{L_n}$ is adopted for $\mathbf{R}$ and $0 < C_l = C_u < \infty$. Here $C_u = C_l$ arises from the label propagation process defined in [13], and it has the effect of keeping f_i of unlabeled $\mathbf{x}_i$ within a reasonable range.

In this paper, we propose a new regularization matrix, and we will focus on problem (2) to solve TC problems, as it allows us to concentrate on the effectiveness of the regularization matrix $\mathbf{R}$.

[2]Other strategies of building the graph are also possible.

3 TC via Local Learning Regularization

In the following, $\mathcal{N}_i$ denotes the set of neighboring points of $\mathbf{x}_i$, *not* including $\mathbf{x}_i$. And n_i denotes $|\mathcal{N}_i|$, i.e. the number of points in $\mathcal{N}_i$. Here, "neighboring points" or "neighbors" of $\mathbf{x}_i$ simply means the nearest neighbors of $\mathbf{x}_i$ according to some distance metric, such as a the Euclidean distance.

3.1 Local Learning methods for Supervised Classification

In supervised classification algorithms, a classifier is trained with all the labeled training data and used to predict the class labels of unseen test data. These algorithms can be called global learning algorithms since the classifier is built based on the whole training dataset. In contrast, in local learning algorithms [3], for a given test data point, a classifier is trained only with its neighboring training data, and then the given test data point is classified by this locally learned model. It has been reported that local learning algorithms often outperform global ones [3] since the local models are trained only with the points that are related to the particular test data.

3.2 Basic Idea

Local learning algorithms described above can *not* be applied directly to TC problems, where there are usually many unlabeled points that do not have any labeled neighbors at all. However, we will show in the following that the idea of local learning can be adapted for TC problems.

The good performance of local learning methods indicates that *the label of a data point can be well estimated based on its neighbors.* Based on this, in order to find a good real valued solutions vector $\mathbf{f}$, we propose to replace the first term of problem (2) with the

following:

$$\sum_{i=1}^{n}(f_i - o_i(\mathbf{x}_i))^2 \tag{9}$$

where $o_i(\cdot)$ denotes the output function of a model, trained locally with some supervised learning algorithms, using the labeled data $\{(\mathbf{x}_j, f_j)\}_{\mathbf{x}_j \in \mathcal{N}_i}$. Details on how to compute $o_i(\mathbf{x}_i)$ will be given later. For the function $o_i(\cdot)$, the subscript i means the model is trained based on the neighbors of $\mathbf{x}_i$. Hence apart from $\mathbf{x}_i$, $\{(\mathbf{x}_j, f_j)\}_{\mathbf{x}_j \in \mathcal{N}_i}$ also influence the value of $o_i(\mathbf{x}_i)$.

A related idea is proposed in [6], where the local estimation of f_i for an unlabeled $\mathbf{x}_i$ is performed using only the labeled neighbors of $\mathbf{x}_i$. When the number of labeled neighbors of $\mathbf{x}_i$ is zero, $\mathbf{x}_i$ is ignored in the training stage. This is different from our approach, where $o_i(\cdot)$ is trained using all the neighboring points of $\mathbf{x}_i$, no matter whether they are labeled or not. Therefore in [6], the result of local estimation of f_i is a specific real number, while in our approach, as can be seen later, $o_i(\mathbf{x}_i)$ is expressed as an analytical equation of f_j for $\mathbf{x}_j \in \mathcal{N}_i$.

To explain the idea behind the (9), let us consider the following problem:

Problem 1. *For a data point $\mathbf{x}_i$, given the values of f_j at $\mathbf{x}_j \in \mathcal{N}_i$, what should be the proper value of f_i at $\mathbf{x}_i$?*

This problem is a typical learning problem and can be solved by some supervised learning algorithms. In particular, following the same approach as in [3], we can build a linear model with the training data $\{(\mathbf{x}_j, f_j)\}_{\mathbf{x}_j \in \mathcal{N}_i}$. As mentioned before, let $o_i(\cdot)$ denote the output function of this locally learned model, then the good performance of local learning methods mentioned above implies that $o_i(\mathbf{x}_i)$ is probably a good guess of f_i, or *the proper f_i should be similar to $o_i(\mathbf{x}_i)$*.

Therefore a good $\mathbf{f}$ should have the following property: *For any $\mathbf{x}_i$ $(1 \leq i \leq n)$, the value of f_i can be well estimated based on the neighbors of $\mathbf{x}_i$. That is, f_i should be similar to the output of the model that is trained locally with the data $\{(\mathbf{x}_j, f_j)\}_{\mathbf{x}_j \in \mathcal{N}_i}$.* This suggests that a good $\mathbf{f}$ will make the value of (9) as small as possible. We call this term the *Local Learning Regularizer* (LL-Reg), since it enforces that f_i should be well estimated by a model trained locally with the neighbors of $\mathbf{x}_i$. This regularizer is our main contribution in this paper.

Replacing the regularizer of (2) with (9) leads to the following optimization problem:

$$\min_{\mathbf{f} \in \mathbb{R}^n} \|\mathbf{f} - \mathbf{o}\|^2 + (\mathbf{f} - \mathbf{y})^\top \mathbf{C}(\mathbf{f} - \mathbf{y}) \tag{10}$$

where $\mathbf{o} = [o_1(\mathbf{x}_1), \ldots, o_n(\mathbf{x}_n)]^\top$.

3.3 Computing $o_i(\mathbf{x}_i)$

Having explained the basic idea, now we consider how to compute $o_i(\mathbf{x}_i)$ to make the regularizer (9) more specific. As stated above, following the approach presented in [3], for each $\mathbf{x}_i$, a linear model is built with the training data $\{(\mathbf{x}_j, f_j)\}_{\mathbf{x}_i \in \mathcal{N}_i}$. To obtain this model, we need to solve the following training problem:

$$\min_{\mathbf{w}_i \in \mathbb{R}^d, b_i \in \mathbb{R}} \lambda \|\mathbf{w}_i\|^2 + \sum_{\mathbf{x}_j \in \mathcal{N}_i} (\mathbf{w}_i^\top (\mathbf{x}_j - \mathbf{x}_i) + b_i - f_j)^2 \tag{11}$$

And the output function $o_i(\cdot)$ of this linear model is:

$$o_i(\mathbf{x}) = \mathbf{w}_i^\top (\mathbf{x} - \mathbf{x}_i) + b_i, \quad \forall \mathbf{x} \in \mathbb{R}^d \tag{12}$$

The reason for $\mathbf{x}_i$ being subtracted in (11) and (12) is as follows [3]: The number of data n_i used to train the linear model for $\mathbf{x}_i$ is usually small, therefore a regularization term, i.e. the first term in (11), is needed to control the capacity of this linear model. However, this term pulls the weight vector $\mathbf{w}_i$ toward some arbitrary origin. For isotropy reasons, the origin of the input space is translated on the testing pattern $\mathbf{x}_i$, by subtracting $\mathbf{x}_i$ from the training points $\mathbf{x}_j \in \mathcal{N}_i$.

According to (12), $o_i(\mathbf{x}_i)$ equals the bias b_i. So by solving the problem (11), $o_i(\mathbf{x}_i)$ can be calculated as:

$$o_i(\mathbf{x}_i) = \boldsymbol{\alpha}_i^\top \mathbf{f}_i \tag{13}$$

where $\mathbf{f}_i \in \mathbb{R}^{n_i}$ is the vector $[f_j]^\top$ for $\mathbf{x}_j \in \mathcal{N}_i$, and

$$\boldsymbol{\alpha_i}^\top = \frac{\mathbf{1}^\top - \mathbf{1}^\top \mathbf{X}_i^\top \mathbf{X}_i (\lambda \mathbf{I} + \mathbf{X}_i^\top \mathbf{X}_i)^{-1}}{n_i - \mathbf{1}^\top \mathbf{X}_i^\top \mathbf{X}_i (\lambda \mathbf{I} + \mathbf{X}_i^\top \mathbf{X}_i)^{-1} \mathbf{1}} \tag{14}$$

In the above equation, $\mathbf{1}$ is the vector of all 1's, $\mathbf{X}_i \in \mathbb{R}^{d \times n_i}$ denotes the matrix $[\mathbf{x}_j - \mathbf{x}_i]$ for $\mathbf{x}_j \in \mathcal{N}_i$.

It can be seen that $\boldsymbol{\alpha}_i$ is independent of $\mathbf{f}_i$ and it is different for different $\mathbf{x}_i$. Note that $\mathbf{f}_i$ is a sub-vector of $\mathbf{f}$, so equation (13) can be written in a compact form as:

$$\mathbf{o} = \mathbf{A}\mathbf{f} \tag{15}$$

where $\mathbf{o}$ is the same as in (10), while the matrix $\mathbf{A} = [a_{ij}] \in \mathbb{R}^{n \times n}$ is constructed as follows: $\forall \mathbf{x}_i$ and $\mathbf{x}_j$, $1 \leq i, j \leq n$, if $\mathbf{x}_j \in \mathcal{N}_i$, then a_{ij} equals the corresponding element of $\boldsymbol{\alpha}_i$ in (14), otherwise a_{ij} equals 0. Similar as $\boldsymbol{\alpha}_i$, the matrix $\mathbf{A}$ is also independent of $\mathbf{f}$.

Substituting (15) into the objective function (10) results in a quadratic optimization problem of the same form as (2), where the LL-Reg is transformed as $\mathbf{f}^\top \mathbf{R} \mathbf{f}$, and

$$\mathbf{R} = (\mathbf{I} - \mathbf{A})^\top (\mathbf{I} - \mathbf{A}) \tag{16}$$

where $\mathbf{A} \in \mathbb{R}^{n \times n}$ is the matrix contained in (15).

Computing the matrix $\mathbf{A}$ in (16) requires calculating $\boldsymbol{\alpha}_i$ in (14) for each $\mathbf{x}_i$. Equation (14) shows that $\boldsymbol{\alpha}_i$

can be computed with time complexity $O(n_i^3)$, therefore calculating $\mathbf{A}$ needs time complexity $O(\sum_{i=1}^n n_i^3)$. Usually the number of neighboring points n_i for each $\mathbf{x}_i$ is small, but sometimes we are also interested in a special case: $n_i = n-1$ for all $\mathbf{x}_i$, i.e. all the data are neighboring to each other. In this case, computing $\mathbf{A}$ as above is very time consuming. So *for this case*, we just build a single linear model for all $\mathbf{x}_i$ by solving the following training problem:

$$\min_{\mathbf{w}\in\mathbb{R}^d, b\in\mathbb{R}} \lambda \|\mathbf{w}\|^2 + \sum_{i=1}^n (\mathbf{w}^\top \mathbf{x}_i + b - f_i)^2 \qquad (17)$$

And $o_i(\mathbf{x}_i) = \mathbf{w}^\top \mathbf{x}_i + b$. Correspondingly, the matrix $\mathbf{A}$ in equation (15) and (16) should be changed as:

$$\mathbf{A} = \mathbf{P} - \mathbf{PQ} + \mathbf{Q} \qquad (18)$$

where

$$\begin{aligned} \mathbf{P} &= \mathbf{X}^\top \mathbf{X}(\lambda \mathbf{I} + \mathbf{X}^\top \mathbf{X})^{-1} \\ \mathbf{Q} &= \mathbf{1}\frac{\mathbf{1}^\top - \mathbf{1}^\top \mathbf{X}^\top \mathbf{X}(\lambda \mathbf{I} + \mathbf{X}^\top \mathbf{X})^{-1}}{n - \mathbf{1}^\top \mathbf{X}^\top \mathbf{X}(\lambda \mathbf{I} + \mathbf{X}^\top \mathbf{X})^{-1}\mathbf{1}} \end{aligned}$$

with $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{d\times n}$.

Thus the overall time complexity of our algorithm is $O(n^3)$, which is identical to many other TC algorithms.

4 Comparison with Related Approaches

4.1 Comparison with Laplacian Regularizers

By considering the problem 1 (cf. Section 3.2), we have derived the LL-Reg. It seeks a vector $\mathbf{f}$ with the property that for each point $\mathbf{x}_i$, the value of f_i is similar to the solution we would obtain by training a model locally if we had known the values of f_j for $\mathbf{x}_j \in \mathcal{N}_i$.

The popular Lap-Reg (6) and NLap-Reg (8) explicitly emphasize the pairwise similarities. They are usually regarded as putting smoothness constraints on the solution vector $\mathbf{f}$. Apart from this, they have also been explained with random walk [15, 14]. And the NLap-Reg is derived from the label propagation process in [13]. Here we show that these two regularizers can also be investigated from the local learning point of view.

In fact, the Lap-Reg (6) *implicitly* gives an answer to problem 1. By setting the gradient $\frac{\partial}{\partial \mathbf{f}} \mathbf{f}^\top \mathbf{L}\mathbf{f}$ to $\mathbf{0}$, it can be seen that the optimal $\mathbf{f}$ minimizing (6) must satisfy the following harmonic property [15]:

$$f_i = \frac{\sum_{\mathbf{x}_j\in\mathcal{N}_i} w_{ij} f_j}{\sum_{\mathbf{x}_j\in\mathcal{N}_i} w_{ij}} \qquad (19)$$

Equation (19) tells that ideally Lap-Reg expects that f_i equals the convex combination of f_j for $\mathbf{x}_j \in \mathcal{N}_i$, and the weight of f_j is proportional to w_{ij}, which measures the similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$. This is the answer we will obtain for problem 1 if we choose the classical Nadaraya-Watson algorithm [9] to estimate f_i based on $\{(\mathbf{x}_j, f_j)\}_{\mathbf{x}_j\in\mathcal{N}_i}$. Similarly, we can find out that the answer to problem 1 implicitly given by NLap-Reg (8) is:

$$f_i = \sum_{\mathbf{x}_j\in\mathcal{N}_i} \frac{w_{ij}}{\sqrt{d_i d_j}} f_j \qquad (20)$$

Therefore, from the local learning point of view, we have obtained some new insights for the popular Laplacian regularizers, based on which we can see that Lap-Reg, NLap-Reg and LL-Reg differ from each other mainly in their answers to the local learning problem 1. Clearly their final classification performance heavily relies on whether the corresponding local models can properly estimate f_i based on $\{(\mathbf{x}_j, f_j)\}_{\mathbf{x}_j\in\mathcal{N}_i}$. Note that problem 1 has two distinct characteristics: the number of training data n_i is usually small, and $\mathbf{x}_i$ is the only point whose label need to be predicted.[3] The approach (11) of [3] is derived from supervised learning algorithms, specially tuned for this local learning problem. Hence the local model (12) used in LL-Reg is probably a good choice for problem 1, and consequently we can expect LL-Reg can generally result in a good classification performance. We will see this later in the experimental results.

Just like no supervised learning algorithm is universally better than the others, approach (11) is by no means the unique answer to the local learning problem 1. In this paper, we simply follow the method in [3] to build a linear local model, but probably other supervised learning algorithms can also result in good performance. In fact, local learning regularization (9) is a flexible framework, which offers us a possible way to adapt various supervised learning techniques for TC problems, as long as the resulting local model can be expressed in the form of (13), where $\boldsymbol{\alpha}_i$ has an analytical expression. Note that to build the local model, some nonlinear learning algorithms, such as kernel ridge regression, also result in the same form as (13). Therefore equation (13) is quite general, not just restricted to linear local models.

Given a local model of the form (13) or equivalently (15), it may be also possible to build a regularizer using the label propagation approach proposed in [13]. For example, the NLap-Reg is obtained in [13] as a result of using the local model (20) for label propagation. It is an iterative method. In each iteration, $\mathbf{f}$ is replaced

[3] Interestingly, this problem itself is a transductive learning problem with labeled data $\{(\mathbf{x}_j, f_j)\}_{\mathbf{x}_j\in\mathcal{N}_i}$, and only one unlabeled point $\mathbf{x}_i$.

with $t\mathbf{Af} + (1-t)\mathbf{y}$, where $0 < t < 1$ is a parameter and $\mathbf{y}$ is the same as in (2). However, to ensure the convergence of iteration, the matrix $\mathbf{A}$ should satisfy some additional constraints. This indicates that deriving regularizers based on local learning has more flexibilities than based on label propagation.[4]

4.2 Comparison with Locally Linear Embedding

Another related approach is the *Locally Linear Embedding* (LLE) algorithm [10], which focuses on building a linear relationship among neighboring points. In LLE, the following optimization problem need to be solved for each $\mathbf{x}_i$:

$$\min_{\beta_{ij}} \quad ||\mathbf{x}_i - \sum_{\mathbf{x}_j \in \mathcal{N}_i} \beta_{ij}\mathbf{x}_j||^2 \tag{21}$$

$$\text{subject to} \quad \sum_{\mathbf{x}_j \in \mathcal{N}_i} \beta_{ij} = 1 \tag{22}$$

Then $\mathbf{x}_i$ is estimated by $\sum_{\mathbf{x}_j \in \mathcal{N}_i} \beta_{ij}\mathbf{x}_j$. We can see that in the LLE algorithm, no label information is used and the linear relationship is built in an unsupervised manner between the input data. While in our approach, we estimate the real valued solution f_i based on $\{(\mathbf{x}_j, f_j)\}_{\mathbf{x}_j \in \mathcal{N}_i}$, using supervised learning methods.

One might argue that despite the above conceptual difference, it is still possible to estimate f_i by $\sum_{\mathbf{x}_j \in \mathcal{N}_i} \beta_{ij} f_j$. To test this idea, and thanks to the flexibility of the local learning approach, we can build a *LLE regularizer* (LLE-Reg) by computing the $o_i(\mathbf{x}_i)$ in (9) and (13) as:

$$o_i(\mathbf{x}_i) = \sum_{\mathbf{x}_j \in \mathcal{N}_i} \beta_{ij} f_j \tag{23}$$

Then replacing equation (13) with (23), the same method in section 3.3 can be applied to compute the corresponding regularization matrix $\mathbf{R}$. A similar regularizer is proposed in [12]. However, the method presented in [12] is based on the label propagation approach, which is different from ours.

5 Computing the Leave-One-Out Classification Error

As mentioned before, several TC algorithms can be formulated as (2). In this section, we derive a method to compute the Leave-One-Out (LOO) classification error efficiently for these algorithms. This is useful for model selection. The content of this section is inspired by a similar result presented in [11], which is proposed for some supervised learning algorithms,

The LOO procedure of the TC problem consists of l iterations. In the i-th ($1 \le i \le l$) iteration, we *just omit the class label y_i and treat $\mathbf{x}_i$ as an unlabeled point.* Specifically, in the i-th iteration, we need to solve the TC problem where the labeled data are $(\mathbf{x}_1, y_1)$, ..., $(\mathbf{x}_{i-1}, y_{i-1})$, $(\mathbf{x}_{i+1}, y_{i+1})$, ..., $(\mathbf{x}_l, y_l)$, while the unlabeled points are $\mathbf{x}_i$, $\mathbf{x}_{l+1}$, ..., $\mathbf{x}_{l+u}$. This is different from the LOO procedure of the supervised learning problem, where there is no unlabeled data, and in the i-th iteration, we simply remove $(\mathbf{x}_i, y_i)$ from the training set, without keeping $\mathbf{x}_i$ for LOO error calculation.

Let $f_i^{(i)}$ denote the i-th element of $\mathbf{f}^{(i)}$, $1 \le i \le l$, which is the real valued solution vector of the i-th iteration in the LOO procedure. In order to compute the LOO classification error, we need to know whether $\mathbf{x}_i$ is correctly classified in the i-th iteration. Hence we need to calculate $f_i^{(i)}$. Instead of computing $f_i^{(i)}$ ($1 \le i \le l$) directly by following the LOO procedure described above, which requires solving l different TC problems, we derive the following lemma, based on which all the $f_i^{(i)}$ can be computed by solving only a single problem (2).

Lemma 1. *Let $\mathbf{M} = (\mathbf{R} + \mathbf{C})^{-1}$ (cf. equation (3)), and m_{ii} denote the i-th ($1 \le i \le l$) diagonal element of $\mathbf{M}$. Suppose $\mathbf{f} = [f_j]^\top$, $1 \le j \le n$, is the solution of problem (2), then $f_i^{(i)}$ can be computed as*

$$f_i^{(i)} = \frac{f_i - C_l y_i m_{ii}}{1 - (C_l - C_u)m_{ii}} \qquad 1 \le i \le l \tag{24}$$

Proof. To simplify the notation, in the proof, we use C_0 to denote $C_l - C_u$.

In the i-th iteration of the LOO procedure, we remove the label y_i and keep $\mathbf{x}_i$ as an unlabeled point. Therefore, in the i-th iteration, we need to solve the following optimization problem:

$$\min_{\mathbf{f}^{(i)} \in \mathcal{R}^n} (\mathbf{f}^{(i)})^\top \mathbf{R}\mathbf{f}^{(i)} + (\mathbf{f}^{(i)} - \mathbf{y}^{(i)})^\top \mathbf{C}^{(i)} (\mathbf{f}^{(i)} - \mathbf{y}^{(i)}) \tag{25}$$

where $\mathbf{f}^{(i)}$ is the solution of the i-th iteration in the LOO procedure, and according to the above description, $\mathbf{C}^{(i)}$ and $\mathbf{y}^{(i)}$ are computed as follows:

$$\mathbf{C}^{(i)} = \mathbf{C} - C_0 \mathbf{e}_i \mathbf{e}_i^\top \tag{26}$$

$$\mathbf{y}^{(i)} = \mathbf{y} - y_i \mathbf{e}_i \tag{27}$$

where $\mathbf{e}_i \in \mathcal{R}^n$ is the vector whose i-th element equals 1 and 0 otherwise. The solution of (25) is

$$\mathbf{f}^{(i)} = (\mathbf{R} + \mathbf{C}^{(i)})^{-1} \mathbf{C}^{(i)} \mathbf{y}^{(i)} \tag{28}$$

Substituting (26) and (27) into (28) leads to

$$\mathbf{f}^{(i)} = (\mathbf{R} + \mathbf{C} - C_0 \mathbf{e}_i \mathbf{e}_i^\top)^{-1} (\mathbf{C} - C_0 \mathbf{e}_i \mathbf{e}_i^\top)(\mathbf{y} - y_i \mathbf{e}_i) \tag{29}$$

[4] Actually the local model used in LL-Reg can not be used for label propagation since the convergence can not be guaranteed.

We now evaluate the first term on the right hand side of the above equation. To this end, we consider the matrix identity [7]

$$(\mathbf{S}+\mathbf{UV})^{-1} = \mathbf{S}^{-1} - \mathbf{S}^{-1}\mathbf{U}(\mathbf{I}+\mathbf{VS}^{-1}\mathbf{U})^{-1}\mathbf{VS}^{-1} \quad (30)$$

where $\mathbf{I}$ is the unit matrix, $\mathbf{S} \in \mathcal{R}^{p\times p}$, $\mathbf{U} \in \mathcal{R}^{p\times q}$ and $\mathbf{V} \in \mathcal{R}^{q\times p}$, where p, q are arbitrary positive integers. If we now identify $\mathbf{R}+\mathbf{C}$ with $\mathbf{S}$, $-C_0\mathbf{e}_i$ with $\mathbf{U}$, and $\mathbf{e}_i^\top$ with $\mathbf{V}$, then $\mathbf{M} = \mathbf{S}^{-1}$ and we can apply (30) to the first term of the right hand side of (29) to obtain

$$(\mathbf{R}+\mathbf{C}-C_0\mathbf{e}_i\mathbf{e}_i^\top)^{-1} = \mathbf{M} + \frac{C_0\mathbf{M}\mathbf{e}_i\mathbf{e}_i^\top\mathbf{M}}{1-C_0\mathbf{e}_i^\top\mathbf{M}\mathbf{e}_i} \quad (31)$$

Using $\mathbf{m}_i$ to denote the i-th column of $\mathbf{M}$ and recall that m_{ii} is the i-th diagonal element of $\mathbf{M}$, we have

$$\begin{aligned} \mathbf{m}_i &= \mathbf{M}\mathbf{e}_i & (32)\\ m_{ii} &= \mathbf{e}_i^\top\mathbf{m}_i = \mathbf{m}_i^\top\mathbf{e}_i & (33)\end{aligned}$$

Then based on (32) and (33), together with the fact that $\mathbf{M}$ is symmetric, equation (31) can be simplified as

$$(\mathbf{R}+\mathbf{C}-C_0\mathbf{e}_i\mathbf{e}_i^\top)^{-1} = \mathbf{M} + \frac{C_0\mathbf{m}_i\mathbf{m}_i^\top}{1-C_0m_{ii}} \quad (34)$$

Now we turn to calculate the product of the second and the third term on the right hand side of (29).

$$\begin{aligned} & (\mathbf{C}-C_0\mathbf{e}_i\mathbf{e}_i^\top)(\mathbf{y}-y_i\mathbf{e}_i) & (35)\\ =\ & \mathbf{C}\mathbf{y} - \mathbf{C}y_i\mathbf{e}_i - C_0\mathbf{e}_i\mathbf{e}_i^\top\mathbf{y} + C_0y_i\mathbf{e}_i\mathbf{e}_i^\top\mathbf{e}_i \\ =\ & \mathbf{C}\mathbf{y} - \mathbf{C}y_i\mathbf{e}_i - C_0y_i\mathbf{e}_i + C_0y_i\mathbf{e}_i \\ =\ & \mathbf{C}\mathbf{y} - C_ly_i\mathbf{e}_i \end{aligned}$$

where the third line uses $\mathbf{e}_i^\top\mathbf{y} = y_i$ and $\mathbf{e}_i^\top\mathbf{e}_i = 1$, and the last line uses $\mathbf{C}\mathbf{e}_i = C_l\mathbf{e}_i$ for $1 \le i \le l$.

Substituting (34) and (35) into (29), we have

$$\begin{aligned} \mathbf{f}^{(i)} &= (\mathbf{M} + \frac{C_0\mathbf{m}_i\mathbf{m}_i^\top}{1-C_0m_{ii}})(\mathbf{C}\mathbf{y} - C_ly_i\mathbf{e}_i) & (36)\\ &= \mathbf{M}\mathbf{C}\mathbf{y} - C_ly_i\mathbf{m}_i + \frac{C_0\mathbf{m}_i\mathbf{m}_i^\top}{1-C_0m_{ii}}(\mathbf{C}\mathbf{y} - C_ly_i\mathbf{e}_i) \end{aligned}$$

where we used (32) in the second line. The sum of the last two terms of the last equation can be computed as

$$\begin{aligned} & -C_ly_i\mathbf{m}_i + \frac{C_0\mathbf{m}_i\mathbf{m}_i^\top}{1-C_0m_{ii}}(\mathbf{C}\mathbf{y} - C_ly_i\mathbf{e}_i) & (37)\\ =\ & -C_ly_i\mathbf{m}_i + \frac{C_0\mathbf{m}_i\mathbf{m}_i^\top\mathbf{C}\mathbf{y} - C_lC_0y_im_{ii}\mathbf{m}_i}{1-C_0m_{ii}} \\ =\ & \frac{C_0\mathbf{m}_i^\top\mathbf{C}\mathbf{y} - C_ly_i}{1-C_0m_{ii}}\mathbf{m}_i \end{aligned}$$

where we used $m_{ii} = \mathbf{m}_i^\top\mathbf{e}_i$ (cf. (33)) in the second line.

Combining (36) and (37) leads to

$$\mathbf{f}^{(i)} = \mathbf{M}\mathbf{C}\mathbf{y} + \frac{C_0\mathbf{m}_i^\top\mathbf{C}\mathbf{y} - C_ly_i}{1-C_0m_{ii}}\mathbf{m}_i \quad (38)$$

Recall that $\mathbf{M} = (\mathbf{R}+\mathbf{C})^{-1}$ and according to (3)

$$\mathbf{f} = (\mathbf{R}+\mathbf{C})^{-1}\mathbf{C}\mathbf{y} = \mathbf{M}\mathbf{C}\mathbf{y} \quad (39)$$

As $\mathbf{M}$ is symmetric and $\mathbf{m}_i$ is the i-th column of $\mathbf{M}$, we have

$$f_i = \mathbf{m}_i^\top\mathbf{C}\mathbf{y} \quad (40)$$

Based on (39) and (40), (38) can be re-written as

$$\mathbf{f}^{(i)} = \mathbf{f} + \frac{C_0f_i - C_ly_i}{1-C_0m_{ii}}\mathbf{m}_i \quad (41)$$

According to (41), we can compute $f_i^{(i)}$, the i-th element of $\mathbf{f}^{(i)}$, as the following

$$f_i^{(i)} = f_i + \frac{C_0f_i - C_ly_i}{1-C_0m_{ii}}m_{ii} = \frac{f_i - C_ly_im_{ii}}{1-C_0m_{ii}}$$

The lemma is proven. □

Note that the proof of lemma 1 is based only on the general form of (2), therefore it can be applied to any TC algorithms that can be formulated as (2). This is the second contribution of this paper.

Hence in order to compute the LOO classification error, first $\mathbf{f}$ is computed with (3), where the matrix $\mathbf{M} = (\mathbf{R}+\mathbf{C})^{-1}$ is also computed at the same time, then $f_i^{(i)}$ can be calculated with (24) for $1 \le i \le l$, based on which the LOO classification error can be easily obtained, since the classification result for $\mathbf{x}_i$ is totally determined by $f_i^{(i)}$ in the i-th iteration of the LOO procedure.[5]

6 Experimental Results

In this section, we empirically compare the following regularizers for the TC problem: Lap-Reg, NLap-Reg, LLE-Reg (cf. Section 4.2) and LL-Reg.

6.1 Datasets and Experimental Settings

Two groups of datasets are adopted in the experiments. The first group is composed of the seven

[5]Note that for $f_i^{(i)}$, the same symbol i is needed for both the superscript and subscript, since to compute the LOO error, we need to know the real valued output of $\mathbf{x}_i$ in the i-th iteration, $1 \le i \le l$.

datasets provided in [4]: g241c, g241d, Digit1, USPS, COIL, BCI and Text.[6] For each one of them, 20 labeled/unlabeled splits are randomly created. The number of labeled point is 100 in each splits for all the datasets.[7]

To further investigate the performance of each method, we construct a second group of data, which contains eight datasets, which are derived from eight classification benchmarks: Banana, Diabetis, German, Image, Ringnorm, Splice, Twonorm and Waveform.[8] The whole set of Diabetis and German are taken directly for the experiments, which contain 768 and 1000 data points respectively. For the remaining six datasets, following the scheme in [4], 1500 data points are randomly sampled from each of them. Similarly as in the first group, 20 labeled/unlabeled splits are randomly generated for each of these eight datasets. For the Diabetis, which contains 768 points, 10% data are randomly selected as labeled points, while for the other seven datasets, the number of labeled points equals 100.

On each dataset, the regularization matrix of each regularizer is calculated and substituted into the matrix $\mathbf{R}$ of problem (2), based on which the corresponding real valued solution vector $\mathbf{f}$, and consequently the final classification result can be obtained. The average classification error on unlabeled data over the 20 labeled/unlabeled splits is used to evaluate the classification performance. And the t-test is conducted to examine whether the performance difference between different approaches are statistically significant.

6.2 Parameter Selection

All the regularizers studied here need to define the neighbors for each point. In the experiments, to well preserve the local structure, we adopt the k-mutual neighbors, i.e. $\mathbf{x}_j$ is defined as a neighbor of $\mathbf{x}_i$ only if $\mathbf{x}_i$ is also one of the k-nearest neighbors of $\mathbf{x}_j$.

In the experiments, a single value k is used for all n_i, $1 \leq i \leq n$. The parameters of each method are searched over some grids by computing the LOO error based on (24).

For LL-Reg, the number of neighbors k of each point, the regularization parameter λ (cf. (11)) are searched over the following grids: $k \in \{5,\ 10,\ 20,\ 50,\ n-1\}$, $\lambda \in \{0.1,\ 1,\ 10\}$. The loss parameter C_l (cf. (2)) does not affect the classification performance too much, hence it is fixed at 10 for LL-Reg.

For Lap-Reg and NLap-Reg, a weighted k-mutual nearest neighbor graph need to be built, whose adjacency matrix is computed based on equation (5). The number of neighbors k, the parameter γ (cf. (5)) are searched from: $k \in \{5,\ 10,\ 20,\ 50,\ n-1\}$, $\gamma \in \{\sigma_0^2/64,\ \sigma_0^2/16,\ \sigma_0^2/4,\ \sigma_0^2,\ 4\sigma_0^2,\ 16\sigma_0^2,\ 64\sigma_0^2\}$, where in a c-class problem, σ_0 is the $1/c^2$ quantile of the pairwise distance of all the data. In addition, we also search the parameter C_l in: $C_l \in \{0.1,\ 1,\ 10,\ 100\}$.

For LLE-Reg, the parameter k and C_l are searched over the following grids: $k \in \{5,\ 10,\ 20,\ 50\}$, $C_l \in \{0.1,\ 1,\ 10,\ 100\}$. For LLE-Reg, we do not consider the case where $k = n-1$, because that requires solving problem (21)–(22) for each point with $n_i = n-1$, which is infeasible.

For simplicity, we would fix $C_u = 0$. However setting C_u to 0 can sometimes cause $\mathbf{R} + \mathbf{C}$ to be ill-conditioned, in which case computing (3) is problematic, therefore in the experiments, we set C_u to a fixed small value, 10^{-6}, for all regularization methods.

6.3 Numerical Results

Numerical results are summarized in Table 1. It can be seen that LL-Reg compares favorably to the other regularizers. Actually it obtains the best performance on most datasets. On the other hand, the LLE algorithm is known to be capable of keeping the manifold structure in the data. Correspondingly LLE-Reg has good performance on Digit1, where the data are close to a low-dimensional manifold embedded into a high-dimensional space [4]. These observations corroborate the statement in section 4 that the property of the local model used by each regularizer can strongly affect the final classification performance.

7 Conclusions

We have proposed a local learning regularization approach for the transductive classification problem. This approach requires that the label of each data point should be well estimated based on its neighbors. Comparison with related approaches illustrates that local learning regularization is a flexible framework, under which we can examine some current regularizers for transductive classification. It also allows us to adapt various learning algorithms for transduction. We have also derived an efficient way to compute the leave-one-out classification error for transductive classification algorithms that can be formulated as (2). Finally, experimental results have been provided to validate the effectiveness of our approach.

[6] See http://www.kyb.tuebingen.mpg.de/ssl-book for detailed descriptions of these datasets.

[7] For these seven datasets, 12 labeled/unlabeled splits are already provided in [4], so we just randomly generate additional 8 splits.

[8] From http://ida.first.fraunhofer.de/projects/bench.

Table 1: Average test error rates (%) and the standard deviations (%) on the fifteen datasets. For each dataset, the results shown in boldface are significantly better than the others, judged by t-test, with a significance level of 0.01. The multiple results in boldface on the same row are not significantly different.

Dataset	Lap-Reg	NLap-Reg	LLE-Reg	LL-Reg
g241c	39.00±2.23	45.00±3.92	41.46±4.51	**21.36±3.67**
g241d	36.12±1.50	43.31±3.30	40.15±4.05	**22.51±1.79**
Digit1	**3.02±0.84**	**2.91±0.59**	**2.54±0.72**	**2.63±0.66**
USPS	7.09±3.37	4.60±2.04	4.70±1.86	**3.67±1.24**
COIL	**11.11±2.72**	**10.71±3.29**	13.61±4.01	12.04±2.30
BCI	47.60±2.29	47.22±2.31	44.23±3.74	**31.15±5.02**
Text	29.29±2.36	**23.55±3.55**	50.11±0.37	**24.23±3.28**
Banana	14.26±1.69	14.15±1.96	17.09±2.48	**12.75±1.70**
Diabetis	32.80±2.54	31.93±2.71	33.31±2.51	**27.63±2.40**
German	31.76±2.51	**30.82±1.40**	33.69±2.95	**29.95±2.49**
Image	14.39±1.63	14.28±1.88	18.67±2.44	**12.08±2.07**
Ringnorm	19.06±2.17	**9.66±0.86**	11.85±1.48	10.28±0.38
Splice	37.48±3.75	36.01±8.50	39.36±2.31	**27.27±5.05**
Twonorm	**4.17±1.30**	**4.05±1.29**	7.54±1.33	**3.35±1.02**
Waveform	17.09±3.27	16.88±3.28	19.02±1.80	**13.50±1.94**

References

[1] M. Belkin, I. Matveeva, and P. Niyogi. Regularization and semi-supervised learning on large graphs. In J. Shawe-Taylor and Y. Singer, editors, *Proc. 17th Annual Conference on Learning Theory*, pages 624–638. Springer, 2004.

[2] M. Belkin, P. Niyogi, and V. Sindhwani. On manifold regularization. In R. G. Cowell and Z. Ghahramani, editors, *AISTATS05*, pages 17–24. Society for Artificial Intelligence and Statistics, 2005.

[3] L. Bottou and V. Vapnik. Local learning algorithms. *Neural Computation*, 4:888–900, 1992.

[4] O. Chapelle, B. Schölkopf, and A. Zien. *Semi-Supervised Learning*. The MIT Press, Cambridge, Massachusetts, 2006.

[5] O. Chapelle and A. Zien. Semi-supervised classification by low density separation. In R. G. Cowell and Z. Ghahramani, editors, *AISTATS05*, pages 57–64. Society for Artificial Intelligence and Statistics, 2005.

[6] C. Cortes and M. Mohri. On transductive regression. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*. MIT Press, Cambridge, MA, 2007.

[7] G. H. Golub and C. F. V. Loan. *Matrix Computations, 2nd edition*. The John Hopkins University Press, 1989.

[8] T. Joachims. Transductive learning via spectral graph partitioning. In T. Fawcett and N. Mishra, editors, *Proc. 20th International Conference on Machine Learning*, pages 290–297. AAAI Press, 2003.

[9] E. A. Nadaraya. *Nonparametric Estimation of Probabiliry Densities and Regression Curves*. Kluwer Academic, Dordrecht, 1989.

[10] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, 2000.

[11] H. Wahba. *Spline Model for Observational Data*. Society for Industrial and Applied Mathematics, Philadelphia and Pennsylvania, 1990.

[12] F. Wang and C. Zhang. Label propagation through linear neighborhoods. In *Proc. 23th International Conference on Machine Learning*, 2006.

[13] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, Cambridge, MA, 2004. MIT Press.

[14] D. Zhou and B. Schölkopf. Learning from labeled and unlabeled data using random walks. In C. E. Rasmussen, H. H. Bülthoff, M. A. Giese, and B. Schölkopf, editors, *Proceedings of the 26th DAGM Symposium*, Berlin, 2004. Springer.

[15] X. Zhu, Z. Ghaharmani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In T. Fawcett and N. Mishra, editors, *Proc. 20th International Conference on Machine Learning*, pages 912–919. AAAI Press, 2003.