Semisupervised Classification With Cluster Regularization

Rodrigo G. F. Soares, Huanhuan Chen, *Member, IEEE*, and Xin Yao, *Fellow, IEEE*

***Abstract*— Semisupervised classification (SSC) learns, from cheap unlabeled data and labeled data, to predict the labels of test instances. In order to make use of the information from unlabeled data, there should be an assumed relationship between the true class structure and the data distribution. One assumption is that data points clustered together are likely to have the same class label. In this paper, we propose a new algorithm, namely, cluster-based regularization (ClusterReg) for SSC, that takes the partition given by a clustering algorithm as a regularization term in the loss function of an SSC classifier. ClusterReg makes predictions according to the cluster structure together with limited labeled data. The experiments confirmed that ClusterReg has a good generalization ability for real-world problems. Its performance is excellent when data follows this cluster assumption. Even when these clusters have misleading overlaps, it still outperforms other state-of-the-art algorithms.**

***Index Terms*— Clustering, machine learning, regularization, semisupervised learning.**

I. Introduction

TRADITIONAL machine learning techniques use only labeled instances (that is, pairs of features and labels) to perform training processes. However, labeled data is usually expensive and time consuming to obtain. For instance, one learning task requires expensive sensors and human experts to gather and label all the data. On the other hand, it might be convenient to collect plenty of unlabeled data, which are typically cheap and abundant. Therefore, it is natural to employ such unlabeled data to improve performance. Semisupervised learning (SSL) aims to use large amounts of unlabeled data along with labeled data to build better learning machines. As SSL requires less human effort and delivers potentially higher accuracy, it became popular in the machine learning community, in both theory and practice [1]. In this paper, we will focus on semisupervised classification (SSC).

SSL can be either transductive or inductive. A classifier is transductive when it cannot generalize its predictions to unseen data. In this situation, test data is regarded as unlabeled data, while inductive learners can generalize their predictions to unseen data.

Manuscript received March 1, 2012; revised July 30, 2012; accepted August 3, 2012. Date of publication October 2, 2012; date of current version October 15, 2012. This work was supported in part by The Capes Foundation, Ministry of Education of Brazil, Brazil, and the European Union Seventh Framework Programme under Grant 270428.

The authors are with the Centre of Excellence for Research in Computational Intelligence and Applications, University of Birmingham, Edgbaston B15 2TT, U.K. (e-mail: rgfsoares@gmail.com; h.chen@cs.bham.ac.uk; x.yao@cs.bham.ac.uk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2012.2214488

One assumption in SSL is that the true class distribution is somehow related to the distribution of the data. In the literature, there are three most-often used assumptions for semisupervised methods [2]. Of them, the semisupervised smoothness assumption assumes that if two instances are close to each other in a high-density region, they are likely to share the same label, also known as the "consistency assumption." The second assumption is the "cluster assumption," which states that classes are often separated by a low-density region, that is, if two data points are in the same cluster they are likely to share the same label (also known as "low-density separation assumption" [2]). The third assumption, i.e., the "manifold assumption," assumes that the true structure of the data lies in a low-dimensional manifold embedded in the high-dimensional data space. Such manifold would deliver better estimates and similarity measures about the data. In this paper, we will focus on the cluster-based SSC methods.

Among cluster-based approaches, most attempt to find a low-density region to separate classes, avoiding placing the decision boundary inside clusters (cutting through high-density regions). Transductive support vector machine (TSVM) [3] is a typical example.

Most of the existing cluster-based SSC approaches do not work well when classes are overlapping, i.e., the decision boundary should be in high-density regions, especially when there is limited labeled data.

However, some clustering algorithms can often easily achieve better performance with overlapping classes when compared to the mentioned margin-based methods,[1] as demonstrated by simple synthetic examples in Figs. 1–4.

The first dataset (two half-moons), in Fig. 1(a), has two labeled points (denoted as dark diamonds) and each moon-shaped cluster corresponds to one class. Both TSVM (Fig. 1) and ClusterReg [Fig. 1(c)] are able to deliver a good decision boundary. The second dataset [Fig. 2(a)] is a different version of the first with one inverted class, which makes such dataset more challenging. As TSVM is sensitive to the position of the single labeled points in each cluster [Fig. 2(b)], it could not find a proper decision boundary. While ClusterReg, taking advantage of self-tuning spectral clustering (STSC) [4], was able to regularize the algorithm to fit the moon-shaped

[1] Intuitively, as seen in Fig. 4(a), in some cases where there is no clear gap between clusters, discovering high-density regions is an easier task than finding low-density gaps between these regions. And clustering algorithms are specifically designed to search for high-density regions. Therefore, in some cases, clustering algorithms can deliver more accurate estimates over the data distribution than methods that seek the largest margin.

2162–237X/$31.00 © 2012 IEEE

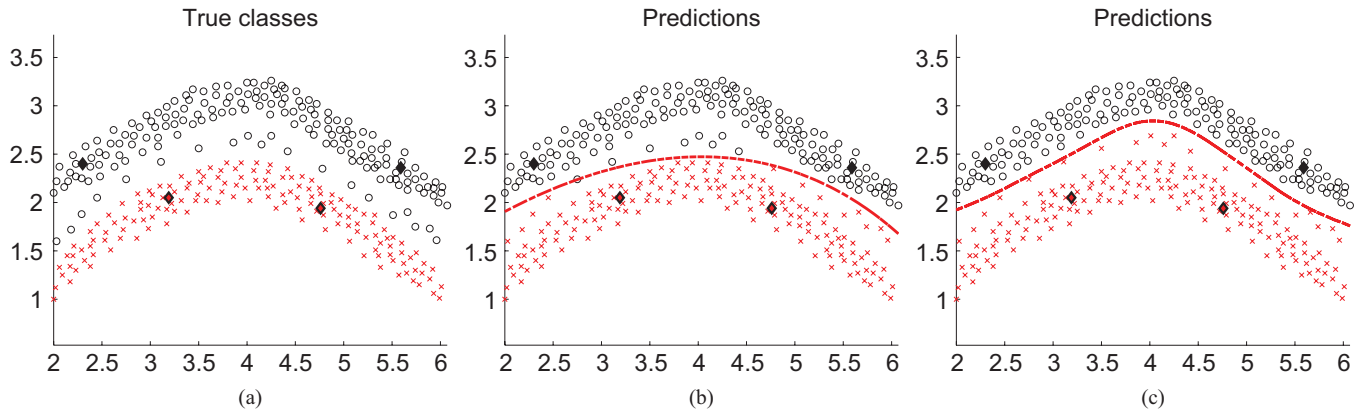

Fig. 1. Synthetic two half-moons dataset. Each half-moon corresponds to one class. (a) True classes. (d) Predictions of TSVM. (c) Predictions of ClusterReg.

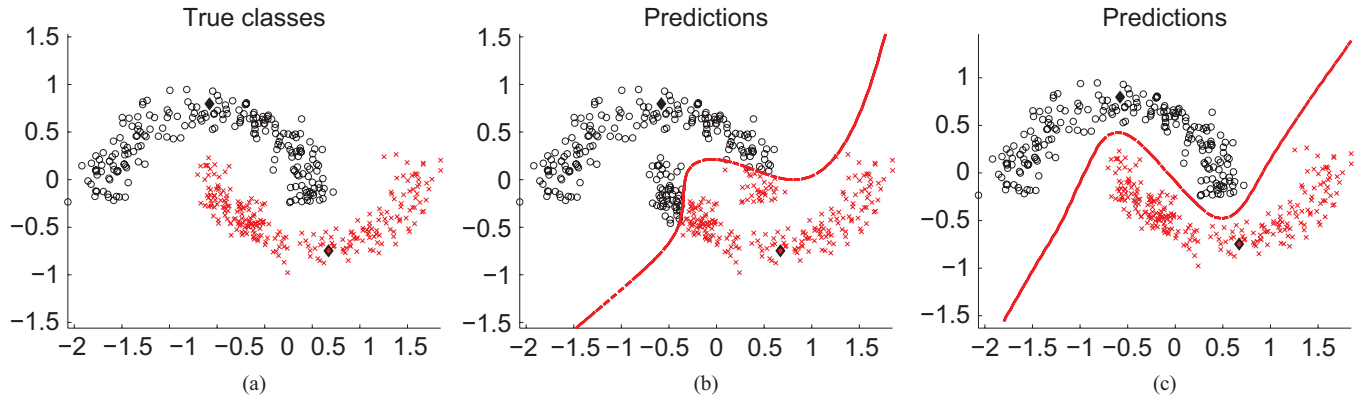

Fig. 2. Two inverted half-moons. (a) True classes. (b) Predictions of TSVM. (c) Predictions of ClusterReg.

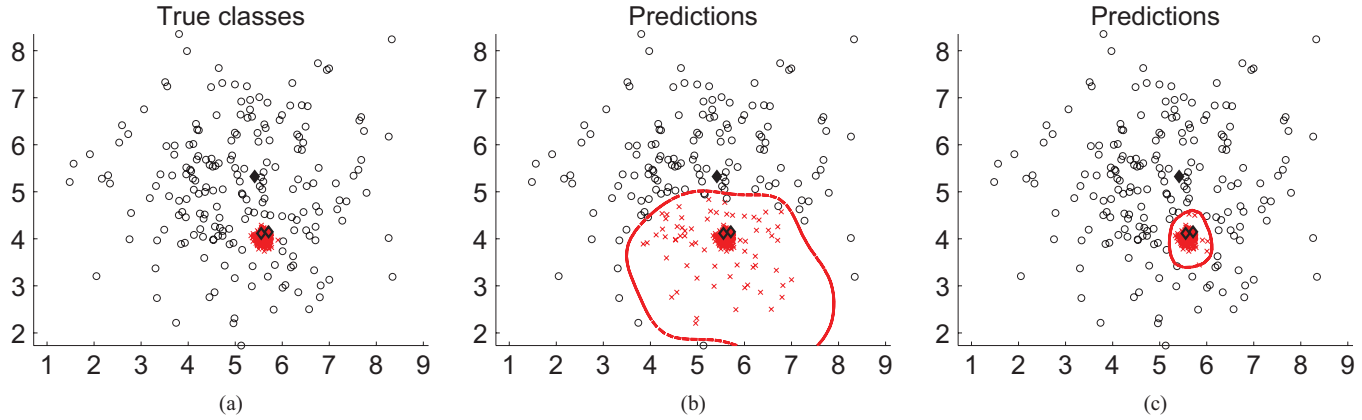

Fig. 3. Dataset with one sparse and one dense class corresponding to clusters. The denser cluster is placed in the sparser cluster. The labeled instances are arbitrarily chosen to mislead the classifiers. They would tend to classify the instances on the bottom of the sparse class as belonging to the tighter class. TSVM is sensitive to the position of the instances in the clusters; therefore it might not find the correct decision boundary. ClusterReg, as STSC can deal with clusters of arbitrary shapes, can take such a partition into account and properly generate a decision boundary. (a) True classes. (b) Predictions of TSVM. (c) Predictions of ClusterReg.

clusters, delivering a smooth decision boundary between classes [Fig. 2(c)]. The third dataset [Fig. 3(a)] has three labeled instances and two classes. One class is sparsely distributed while the second corresponds to a denser cluster inside the other class. The labeled points are arbitrarily placed to mislead classifiers. That is, the instances in the bottom of the sparse class are prone to be classified as belonging to the dense class. As expected, in Fig. 3(b), TSVM is not able to correctly predict the labels of the instances in the bottom of the sparse class, since there is no labeled instance in that region. However, ClusterReg incorporates the partition information from STSC to avoid cutting though the dense and sparse cluster, which makes it more robust to the position of labeled instances, as shown in Fig. 3(c).

In Fig. 4(a), a 2-D dataset with six Gaussians corresponds to the true data distribution with six classes. The classes

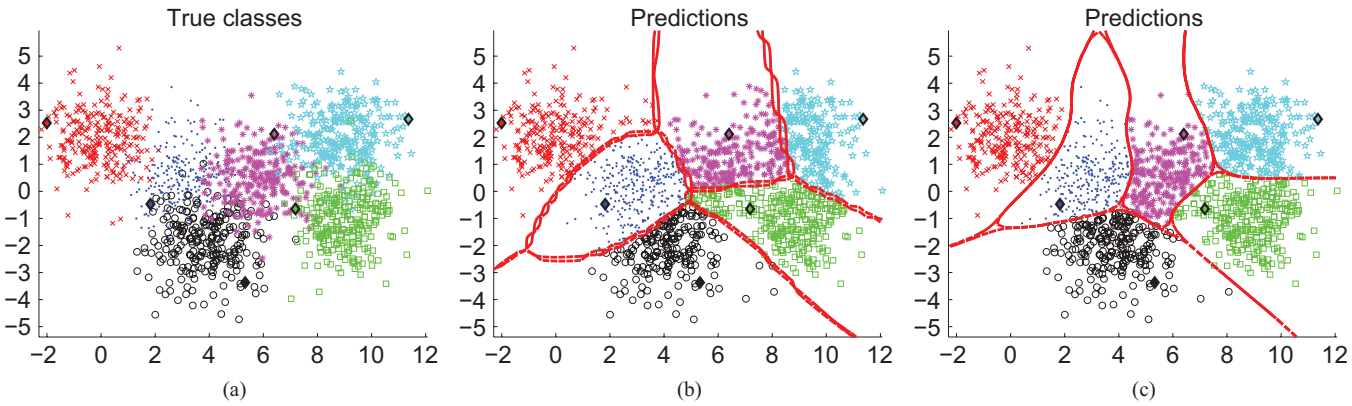

Fig. 4. Dataset with six overlapping classes drawn from unit-variance isotropic Gaussians ($\mathcal{N}(\mu, \mathbf{I})$) and translated. Because of the overlapping clusters, TSVM cannot find the appropriate decision boundary. ClusterReg, by considering the partition of a clustering algorithm, is able to find a better decision boundary. (a) True classes. (b) Predictions of TSVM. (c) Predictions of ClusterReg.

were designed to overlap, while still keeping the cluster structure. The labeled instances, denoted by black diamonds, were chosen to lie roughly on the borders of the classes.

Such data is challenging for existing algorithms, since these clusters do not have a clear gap between them. The decision boundaries of these algorithms are mainly determined by the distribution of these limited labeled data, which is not robust with few labeled data.

As an example, for a multiclass case, shown in Fig. 4(b), TSVM could not find the appropriate gap, making the decision boundary cut through the clusters. However, clustering algorithms (such as Gaussian mixture models (GMMs) or K-means [5] as verified in this case) may identify those six clusters.[2] When we apply the cluster structure to our method, it becomes more robust to the position of the few labels in the clusters. Therefore, clustering algorithm can be properly wrapped into SSC to improve its performance on this kind of data.

In this paper, we propose to incorporate clustering algorithms in ClusterReg to overcome the issues mentioned above. In this algorithm, we also take into account the probability of each instance belonging to each cluster to regularize the loss function. As shown in Fig. 4(c), unlike other cluster-based methods, our method can easily establish a decision boundary between clusters without being misled by either the overlapping classes or the few labels. ClusterReg achieves such robustness by incorporating clustering algorithm into its mechanism. This simple case confirmed the benefits of our algorithm for overlapping clustering data.

ClusterReg regards the structure arising from the clustering algorithm as a soft partition. That is, each instance is assigned a probability of belonging to a given cluster, unlike hard partition where clusters are strictly disjoint. By using soft partitions (also known as "soft clustering"), we can address uncertain instances (likely in low-density region, that is, in the border of clusters) differently from the more confident ones (likely in the densest region of clusters). Soft clustering helps the algorithm regard uncertain instances (with low probabilities for all clusters) as instances lying in gaps, thereby helping the classifier to generate the decision boundary in such low-density (according to the clustering algorithm) region.

The contribution of this paper is to propose an algorithm that employs any clustering algorithm[3] into SSC to regularize the prediction. The proposed algorithm: 1) is robust under the presence of fewer labeled points; 2) is robust to the position of labeled data in clusters by considering the strength of clustering algorithms in a natural way; and 3) is able to improve the performance of a given classifier when the classes or clusters overlap, compared to other cluster-based algorithms.

ClusterReg can handle any clustering algorithm with a proper processing of its output. It can employ any classifier that is able to use the proposed loss function. Therefore, ClusterReg can be seen as a framework for SSC methods.

The remainder of this paper is organized as follows. The next section presents a review of existing methods. Section III introduces the proposed algorithm in details. Then, we present the experimental results and we discuss our algorithm in Section IV. Finally, Section V discusses our contributions and Section VI presents the conclusions.

II. Related Work

In this section, we review most relevant SSC methods, pointing out their advantages and drawbacks.

A. Co-Training

In the co-training algorithm [7], the features in the training set will be divided into two different sets (views). Such splitting can be achieved naturally if the data have intrinsically two possible features sets, or by applying some artificial method, such as randomly selecting features. The algorithm also assumes that both feature subsets are good enough to train a classifier and they are conditionally independent given the class. The algorithm consists of two classifiers, each one

[2]The example shown in Fig. 4 is suitable for K-means and GMMs, because the clusters are spherical. There are other situations where ClusterReg can take advantage of other clustering methods, such as spectral-based clustering algorithms, to estimate the cluster structure with arbitrary shape [4], which is the case of the datasets in Figs. 1(a), 2(a), and 3(a).

[3]In this paper, we use tering algorithm, K-means [5], STSC, GMM, and fuzzy GK clustering [6], to evaluate ClusterReg.

assigned to one different view. Initially, each one is trained with the labeled data from the view it is associated to. Then, each classifier labels the unlabeled data of its own view (pseudo-labeling) and puts the most confident ones with their predicted labels on the training set of the other classifier. Afterwards, they are then retrained with the newly obtained instances. In fact, both classifiers teach each other, and tend to agree in the labeled and also in the unlabeled instances. Co-training makes strong assumptions on the splitting of features. In order to relax such assumptions, in [8] the authors propose the tri-training algorithm, which uses three learners. For training one classifier, it uses the agreement between the other two classifiers to label one given unlabeled instance. Tri-training and co-training may end up overfitting the most confident instances, leading to no gain in the classification accuracy. This fact arises when the splitting of the dataset is not straightforward. The method depends on the quality of such splitting of features.

B. Methods Based on Manifold Assumption

Most manifold-based methods assume that there is a low-dimensional manifold structure embedded in the data space. Typically, these algorithms build graphs to represent all instances [1]. The nodes represent the instances (labeled and unlabeled) and the edges denote the similarity between the instances. In order to predict labels in the graph, these methods usually assume label smoothness among these instances.

Spectral graph transducer (SGT) [9] can be seen as a semisupervised version of the K nearest neighbor classifier. This method uses unlabeled instances to build a graph. The nature of its manifold assumption is in the fact that predictions are based on the neighborhood of an instance within the graph. This method is proposed for binary classification, which could be a shortcoming since it depends on the decomposition of multiclass datasets into a set of different binary tasks, leading to problems of imbalanced classification and different output scales of binary classifiers [10]. Moreover, graph-based approaches, such as SGT, often leave the graph construction, an important part of its learning, out of the training algorithm.

Most of the graph-based approaches only focus on the optimization functions, leaving the graph construction, which is an important part of the learning procedure, out of the framework. In consequence, the issue of graph construction has not been studied extensively yet [1].

Besides the graph, these procedures usually cannot deal with unseen (test) data, that is, they are inherently transductive. This can prevent the application of graph-based methods in problems requiring fully inductive classifiers.

C. Methods Based on Cluster Assumption

Among the methods based on cluster assumption, we can highlight the TSVM algorithm [3]. It is an extension of the SVM method (also known as semisupervised SVM, S3VM). TSVM uses unlabeled data to find the decision boundary with the largest margin between classes. Unlike SVM, TSVM tries to maximize the margin with a linear boundary by considering both labeled and unlabeled instances, which might deliver lower generalization error [11]. The unlabeled data drive the decision boundary away from dense regions [1]. However, if the dense regions are overlapping, TSVM might not find the correct decision boundary between such regions (clusters). And, in this case, this algorithm might be sensitive to the few labeled points in the dense regions. Moreover, for multiclass classification problems, this method has the same drawback, mentioned before, as other binary SSC algorithms: it depends on the decomposition of the dataset into a number of independent binary classification problems.

In [12], the authors introduced the Bayesian semisupervised SVM (SemiBSVM) model for binary classification. TSVM and SemiBSVM aim to find the largest margin in both labeled and unlabeled data space. The loss function was specially designed with a penalty term with the likelihood part constructed from the unlabeled data. SemiBSVM was successfully compared with some supervised classification methods when the unlabeled data was very informative, especially in the cases where the amount of unlabeled data was much larger than the labeled data. However, similar to TSVM, SemiBSVM is a binary classifier that is sensitive to overlapping high-density regions with few labeled data.

Recently, in [13], the authors introduced the SSC based on class membership (SSCCM) algorithm. It employs a loss function that uses the concept of "label membership" to weight the pertinence of a given instance to each class. And, in order to have more reliable labels, such function also regards each instance as a weighted average of its neighbors. As TSVM, this algorithm seeks the largest margin separator. Experiments showed that it outperformed other methods with hard labels. However, unlike ClusterReg, the method equally considers instances in low- and high-density regions; that is, an unreliable (uncertain with respect to its membership) instance lying in the border of a cluster (or class, if cluster assumption holds) has the same impact in the training process as any other instance, whereas the intuition behind the cluster assumption suggests that the sharing of labels should be more reliable in high density regions.

As mentioned before, these cluster-based methods try to find the largest margin between high-density regions (clusters). When overlapping clusters are present, with sparse labeled instances on their borders, these classifiers may not produce good predictions, although these inherent clusters might be easily identified.

Our proposed algorithm, ClusterReg, is a cluster-based method. Unlike other cluster-based algorithms, it does not depend on gaps between potential clusters, but captures the partition information from a clustering algorithm in order to improve the decision boundary.

D. Ensemble Methods in SSC

Some semisupervised approaches try to make use of an ensemble of base classifiers to perform the SSC task. Some of these methods use an ensemble of supervised algorithms, while others use an ensemble of semisupervised methods.

The SemiBoost algorithm [14] is a meta SSL algorithm. The method combines the similarity information among the

instances with the classifier predictions to obtain more reliable pseudo-labels. It is a graph-based ensemble approach. Its objective function has the smoothness, manifold, and cluster assumptions. Such algorithm can be used to improve the classification accuracy of any supervised learning algorithm using unlabeled instances.

SemiBoost is a binary class algorithm. Later, a similar boosting approach, multiclass semisupervised boosting (MCSSB), was proposed by [10] in order to solve multiclass tasks. In [15], the authors extended the information regularization framework to semisupervised boosting. The authors proposed sequential gradient descent optimization algorithms to optimize the loss function. Such loss function incorporates all three SSC assumptions.

In [16], the authors proposed a tree-structured ensemble approach where a complex multiclass problem is decomposed into a set of binary subproblems. Each subproblem (a binary classification) is represented as an internal node in a tree. The leaf nodes represent the classes. In each internal node, the algorithm performs a co-training procedure using RBF networks as base classifiers. The authors also compared another approach in which the co-training classifiers are the ensemble trees. The authors demonstrated that the combination of tree-structured ensemble and co-training is especially useful for classification tasks that involve a large number of classes and a small amount of labeled data. However, the tree-structured ensemble, similar to co-training, may classify unlabeled instances incorrectly and when such instances are used to train other classifiers, errors may be reinforced.

RegBoost [17] employs three semisupervised assumptions in its boosting learning style algorithm. It uses a kernel density estimation approach which penalizes the classifier if it does not assign the same label to a pair of neighbor instances in a high-density region, to implement cluster assumption. However, as mentioned before, if overlapping high-density regions are present, RegBoost might not establish a good separation between these regions. Moreover, this algorithm is designed only for binary classification.

As mentioned before, a decomposition technique, such as one-against-the-rest [10], can be employed to extend the algorithm to multiclass problems. However, as expected, our experiments showed that RegBoost delivers inferior results when applied to multiclass real-world datasets.

In ensemble-based SSC approaches, ensemble pruning techniques [18], [19] can be employed for a compact yet powerful ensemble classifier.

III. ClusterReg Algorithm

In this section, we present the proposed multiclass semisupervised algorithm, ClusterReg. First, we introduce the new loss function with a regularization term based on clustering algorithm. And we show an example using multilayer perceptron (MLP) in this algorithm.

The general architecture of ClusterReg is presented in Fig. 5. The first step of the method is executing a given clustering algorithm on the dataset and extracting the probabilities of each instance belonging to each cluster. The initialization

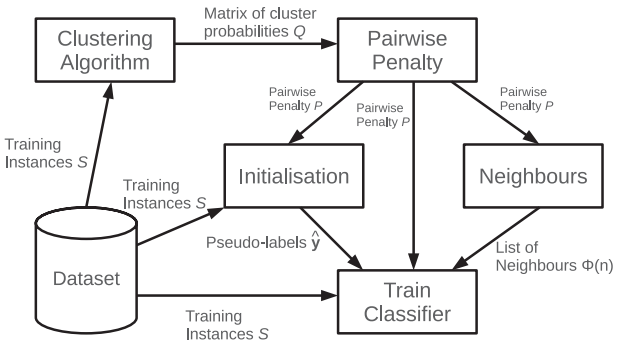

Fig. 5. ClusterReg's architecture.

procedure assigns the initial pseudo-labels to the unlabeled instances according to the labeled instances in the clusters they belong to. Afterwards, ClusterReg computes the pairwise penalty according to the output of the clustering algorithm. The penalty values are the metric employed to find the nearest neighbors of each instance. The neighborhood of a given instance is defined as those instances with the highest penalty values relative to that instance. Then, with the initial pseudo-labels, penalty values, and nearest neighbors at hand, the classifier is trained by a method that minimizes the proposed loss function. We will show the details of these steps in the following sections.

A. Semisupervised Loss Function

Formally, in SSC, the training set $S = L \cup U$ is composed of l labeled instances $L = \{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$ and u unlabeled instances $U = \{(\mathbf{x}_i)\}_{i=l+1}^{u}$ and $N = l + u$, often $u \gg l$. Basically, the aim of SSC is to improve the classifier f in comparison to using the labeled data L alone.

In this paper, we propose a new algorithm in order to include the clustering information in the SSC algorithm. The main idea is to use the output of clustering algorithms to regularize the loss function of the SSC algorithm. Its first term is fully supervised, using only the labeled instances to measure the difference between the classifier output and the true labels. The second term represents the semisupervised regularization procedure. The proposed loss function is in (1).

The output of a clustering algorithm is a partition $Q = [q_{ij}]_{N \times K}$ with K clusters and N instances, where the row vector $\mathbf{q}_n$ contains the probabilities of instance n belonging to each one of the K clusters. For example, $\mathbf{q}_n = (0.3, 0.1, 0.6)$ denotes that n has a 30% of chance of belonging to the first cluster, and so on. Consequently, the vector sums to 1. And n belongs to the third cluster as it holds the highest probability

$$E = \frac{1}{N} \sum_{n=1}^{N} \sum_{j=1}^{G} \left\{ I_{nL} C[y_{nj}, f_j(n)] + I_{nU} \frac{\lambda \max(\mathbf{q}_n)}{|\phi(n)|} \times \sum_{k \in \phi(n)} P(\mathbf{q}_n, \mathbf{q}_k) C[\hat{y}_{kj}, f_j(n)] \right\} \quad (1)$$

where $I_{wA} = 1$ if $w \in A$ and 0 otherwise. $f_j(n)$ denotes the output of the classifier for class j and instance n. $C[y_j, f_j(n)]$ can be any monotonically decreasing loss function, for example, mean squared error or cross entropy. $P(\mathbf{q}_n, \mathbf{q}_k)$ is the

penalty assigned to instance n and k. The parameter λ denotes the tradeoff between the supervised loss and semisupervised regularization. G is the number of classes. $\max(\mathbf{q}_n)$ returns the maximum value in the vector $\mathbf{q}_n$ to indicate the most probable cluster that instance n belongs to. $|\phi(n)|$ is the cardinality of the set of neighbors of instance n, i.e., the number of neighboring points for instance n. $\hat{y}_{kj}$ is an estimate of the desired output for instance n regarding its neighbor k. $\hat{y}_{kj}$ can be either the true label y_{kj} if k is a labeled instance, or the output $f_j(k)$ if k is unlabeled. When k is unlabeled, $\hat{y}_{kj}$ is also known as pseudo-label of k.

The penalty function, presented in (2), measures the similarity between vectors $\mathbf{q}_n$ and $\mathbf{q}_k$. By doing so, we consider the similarity as a direct outcome from the clustering algorithm. Such penalty function uses similarity measures, r [in (3)] and s [in (4)], and map them into a penalty factor in the regularization term. The product $r(\mathbf{q}_n, \mathbf{q}_k) * s(\mathbf{q}_n, \mathbf{q}_k)$ is normalized in $[0, 1]$

$$P(\mathbf{q}_n, \mathbf{q}_k) = \sin\left(\frac{\pi}{2}\left(r(\mathbf{q}_n, \mathbf{q}_k) * s(\mathbf{q}_n, \mathbf{q}_k)\right)^{\kappa}\right). \tag{2}$$

The parameter κ controls the steepness of the mapping from similarity to penalization. This value regulates the degree of decision boundary, cutting through clusters. If we increase κ, we relax the cluster assumption by letting the classifier split a high-density region. On the other hand, decreasing the parameter forces the classifier to avoid placing the decision boundary inside clusters. This form of penalty is flexible to allow different levels of penalization for highly similar instances, while assigning low penalty to instances with low similarity, according to a proper value of κ [17].[4]

There are many ways to measure the similarity between vectors. In this paper, we focus on the correlation coefficient and the Euclidean distance (transformed into similarity) between the probability vectors $\mathbf{q}_n$ and $\mathbf{q}_k$. Using Euclidean distance alone may not capture all the information between two vectors. Suppose we have the probability vectors $\mathbf{u} = (0.8, 0, 0.2)$, $\mathbf{v} = (0.5, 0.2, 0.3)$, and $\mathbf{w} = (0.8, 0.2, 0)$; we can notice that the instances they represent belong to the same cluster, which is the one with the highest probability. The Euclidean distance between $\mathbf{u}$ and $\mathbf{v}$ is $||\mathbf{u} - \mathbf{v}|| = 0.37$ and $||\mathbf{u} - \mathbf{w}|| = 0.28$. However, $\mathbf{v}$ has higher chance of belonging to the third cluster than the second, which is also the case for $\mathbf{u}$, whereas for $\mathbf{w}$, the second highest probability is for the second cluster. In this sense, $\mathbf{v}$ should be the point more similar to $\mathbf{u}$, instead of $\mathbf{w}$. Therefore, although all the corresponding instances belong to the same cluster, the correlation between their cluster probability distribution should be taken into account. Then, we use Pearson's correlation coefficient along with the Euclidean distance to calculate the penalization for a pair of points.

Formally, (3) shows the similarity concerning the correlation between two probability vectors

$$r(\mathbf{q}_n, \mathbf{q}_k) = \frac{\sum_{i=1}^{K}(q_{ni} - \bar{\mathbf{q}}_n)(q_{ki} - \bar{\mathbf{q}}_k)}{\sqrt{\sum_{i=1}^{K}(q_{ni} - \bar{\mathbf{q}}_n)^2}\sqrt{\sum_{i=1}^{K}(q_{ki} - \bar{\mathbf{q}}_k)^2}} \tag{3}$$

[4] κ is chosen as a positive value in [1, 12]. Lower values lead to no difference of penalty for dissimilar instances, and higher values do not penalize even the most similar instances.

where $\bar{\mathbf{q}}_n$ is the mean of the vector $\mathbf{q}_n$. For the second similarity measure, we compute all the pairwise Euclidean distances between the probability vectors and normalize them in $[0, 1]$. Then, we transform the Euclidean distance into similarity as shown in (4). Therefore, similar instances should be close to each other and highly correlated

$$s(\mathbf{q}_k, \mathbf{q}_n) = 1 - \frac{d(\mathbf{q}_k, \mathbf{q}_n) - d_{\min}}{d_{\max} - d_{\min}} \tag{4}$$

where $d(\mathbf{q}_k, \mathbf{q}_n) = ||\mathbf{q}_k - \mathbf{q}_n||$. And $d_{\max}$ and $d_{\min}$ are the maximum and minimum Euclidean distances, respectively.

As we intend to use the structure arising from the clustering algorithm to calculate the similarity in the regularization term, we also employ this information to find the nearest neighbors $\phi(n)$. Then, the nearest neighbors of n are the V instances[5] with the highest $P(\mathbf{q}_k, \mathbf{q}_n)$.

Following the smoothness assumption, the regularization term in (1) penalizes the classifier if it assigns different labels to similar instances. This is achieved by the product $P(\mathbf{q}_k, \mathbf{q}_n)C[\hat{y}_{kj}, f_j(n)]$. That is, if the classifier gives different outputs for two similar instances, the loss and penalty will be high, causing a large regularization to the training. On the other hand, if the penalty is low (the instances are not similar according to the clustering algorithm), it does not matter if the classifier assigns distinct labels to the couple of instances.

Regarding the cluster assumption, we use the density information in Q to regularize the classifier, following the cluster structure given by some clustering algorithm. In order to complete the cluster assumption, we also add the maximum value in the probability vector $\max(\mathbf{q}_n)$ as a factor in the second term of the loss function. It weights the importance of instance n as an estimate of the density in its region. The higher this value, the higher is the density. So, we penalize the training if the classifier assign two different labels to the instance to be learned n and its neighbor k; and the penalty will be even higher if n is in a high-density region, according to the clustering algorithm. Therefore, the classifier will avoid delivering a decision boundary that crosses through clusters.

B. Initialization Procedure

In the beginning, ClusterReg does not have the estimated labels (pseudo-labels) of the neighbors of a given instance n to perform the regularization for that point. We applied the initialization procedure described in [17] to assign pseudo-labels to the unlabeled instances. The output of cluster algorithm is employed to set the pseudo-labels $\hat{\mathbf{y}}$. Therefore, for each position j of each unlabeled instance n in cluster Ψ, we have

$$\hat{y}_{nj} = \text{softmax}\left(\sum_{k \in \Psi} I_{kL} * P(\mathbf{q}_n, \mathbf{q}_k) * y_{kj}\right) \tag{5}$$

the softmax function is defined in (6).

The pretraining, with the pseudo-label values assigned to unlabeled instances, is performed for a certain number of iterations. Throughout this paper, we use 10 iterations of pretraining, as different numbers did not improve performance in preliminary experiments.

[5] V is the number of neighbors.

Algorithm 1 ClusterReg Algorithm With MLP and Cross Entropy Loss Function

Input: Training set $S = L \cup U$
Output: Trained MLP.

$Q \leftarrow$ cluster(S) {cluster is a given clustering algorithm.}
for $n = 1$ to N **do**
  for $k = 1$ to N **do**
    $P(\mathbf{q}_n, \mathbf{q}_k) \leftarrow \sin\left(\frac{\pi}{2}\left(r(\mathbf{q}_n, \mathbf{q}_k) * s(\mathbf{q}_n, \mathbf{q}_k)\right)^{\kappa}\right)$. {Computing the pairwise penalty.}
  end for
end for
for all clusters Ψ **do**
  $\hat{y}_{nj} = \text{softmax}\left(\sum_{k\in\Psi} I_{kL} * P(\mathbf{q}_n, \mathbf{q}_k) * y_{kj}\right)$ {Initialization.}
end for
for all instances n **do**
  $\phi(n) \leftarrow$ the V instances with highest $P(\mathbf{q}_n, \mathbf{q}_k)$ {Finding the nearest neighbors.}
end for
for all instances n, nodes j and weights w_{ji} with inputs x_i **do**
  Perform a desired number of updates on w_{ji} according to the delta rule:

$$\Delta w_{ji} = \left\{ I_{nL}\frac{y_{nj}}{f_j(n)} + I_{nU}\frac{\lambda \max(\mathbf{q}_n)}{|\phi(n)|}\sum_{k\in\phi(n)} P(\mathbf{q}_k, \mathbf{q}_n)\frac{\hat{y}_{kj}}{f_j(n)} \right\} * g'(h_j) * x_i$$

end for

The initialization procedure of neural network has a great impact in the outcome of the training. This procedure ensures that, at the first iterations, the classifier has more reliable estimates over the labels of unlabeled instances. These estimates are weighted pairwise penalty values within each cluster. Without such technique, the error might degrade, compromising the final generalization ability of the method.

C. ClusterReg by MLP

In this paper, we apply the proposed loss function to feedforward MLP networks with one hidden layer. We chose to use the cross entropy as the loss function and softmax as the output activation function [20] since they form a natural pairing that leads to more accurate results [21]. Additionally, cross entropy may be robust in maintaining its performance advantage for problems with limited amounts of data [22]. Algorithm 1 describes the ClusterReg method.

The softmax activation function employed in the output nodes of a MLP is as follows:

$$f_j(n) = \text{softmax}(z_{nj}) = \frac{\exp(z_{nj})}{\sum_{i=1}^{G}\exp(z_{ni})} \tag{6}$$

where z_{nj} is the linear combination of weights and inputs of the node j for instance n.

The cross entropy function is presented in

$$\text{XEnt}_j(n) = -y_{nj} * \ln\left(\frac{f_j(n)}{y_{nj}}\right). \tag{7}$$

Then, instantiating the loss function with the cross entropy, the loss function becomes

$$E = -\frac{1}{N}\sum_{n=1}^{N}\sum_{j=1}^{G}\left\{ I_{nL}\, y_{nj} \ln\left(\frac{f_j(n)}{y_{nj}}\right) + I_{nU}\frac{\lambda \max(\mathbf{q}_n)}{|\phi(n)|} \times \sum_{k\in\phi(n)} P(\mathbf{q}_k, \mathbf{q}_n)\hat{y}_{kj} \ln\left(\frac{f_j(n)}{\hat{y}_{kj}}\right)\right\}. \tag{8}$$

We used the scaled conjugate gradient (SCG) algorithm to train our neural networks since it does not depend on user parameters [23]. In order to apply our loss function to the SCG algorithm, we present the derivative of (8) with respect to the weight w_{ji} of node j in

$$\frac{\partial e_{nj}}{\partial w_{ji}} = \frac{\partial e_{nj}}{\partial f_j} * \frac{\partial f_j}{\partial h_j} * \frac{\partial h_j}{\partial w_{ji}} = -\left\{ I_{nL}\frac{y_{nj}}{f_j(n)} + I_{nU}\frac{\lambda \max(\mathbf{q}_n)}{|\phi(n)|} \times \sum_{k\in\phi(n)} P(\mathbf{q}_k, \mathbf{q}_n)\frac{\hat{y}_{kj}}{f_j(n)}\right\} * g'(h_j) * x_i \tag{9}$$

where $g'(h_j)$ is the derivative of the activation function $g(h_j)$ of neuron j with respect to its total input $h_j = \sum_i w_{ji} * x_i$.

In this method, we can apply any clustering algorithm. Four algorithms from various clustering approaches, namely K-means, STSC, GMM, and fuzzy GK clustering, are employed in this paper.[6] Since the first two clustering algorithms do not output probabilities, we employed a simple procedure to transform the original output g_{ni} for instance n and cluster i into the probability q_{ni}. For K-means, where g_{ni} is the distance from instance n to the cluster centroid i, we have

$$q_{ni} = \frac{1 - \left(\frac{g_{ni}}{\sum_{k=1}^{K} g_{nk}}\right)}{K - 1}. \tag{10}$$

STSC outputs a matrix of K eigenvectors with N dimensions, g_{ni} is the nth position of the ith eigenvector, and we have

$$q_{ni} = \frac{|g_{ni}|}{\sum_{k=1}^{K}|g_{nk}|}. \tag{11}$$

IV. Experiments

A. Transductive Settings

In the transductive learning settings, the test instances are used as unlabeled data during the training phase of a classifier—the generalization error is the training error on unlabeled data. Several benchmarks have been designed and used for this setting in [2]. We selected three artificial datasets—g241c, g241d, and Digit1—and three real world datasets—USPS, COIL, and BCI—to evaluate the proposed algorithm and other state-of-the-art methods.

Among the artificial datasets, g241c was specifically generated such that the classes correspond to clusters, so the cluster assumption holds. g241d was specially built so that the cluster assumption is misleading and the manifold assumption does not hold. And Digit1 was designed to have a low-dimensional manifold embedded into a high-dimensional space and does not show cluster structure.

All six datasets have 2 classes [equally balanced, as shown in Fig. 6(a) and (c)], 1500 instances, and 241 dimensions; except for BCI, which has 400 instances and 114 dimensions, and COIL with 6 classes.

Each dataset has 12 subsets of 10 and 100 labeled instances, and the algorithms are run 12 times with 10 and 100 labels

[6] K-means and GMM are sensitive to the initialization of centroids and components, respectively. We run these algorithms five times and choose the result with the least intracluster variance.

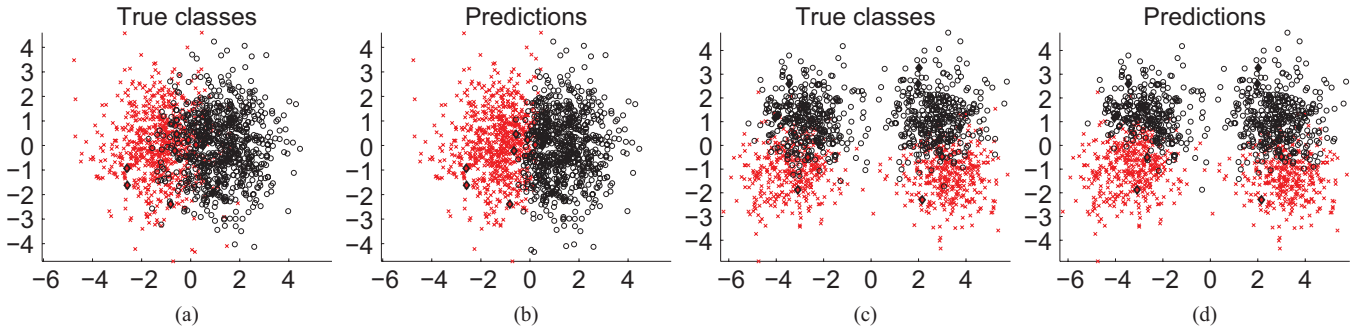

Fig. 6. 2-D projections of true classes and predictions from ClusterReg for g241c and g241d with 10 labeled instances, denoted by dark diamonds. (a) True classes of g241c. (b) Predictions of ClusterReg with K-means, $K = 2$, $\kappa = 5$, $V = 20$, for g241c. (c) True classes of g241d. (d) Predictions of ClusterReg with K-means, $K = 4$, $\kappa = 5$, $V = 20$, for g241d.

TABLE I

AVERAGE OF ERRORS (%) OF THE RUNS WITH 12 SUBSETS OF LABELED DATA. FOR ALL THE ALGORITHMS, THE TEST SETS ARE FIXED. THE TABLE REPORTS ONLY THE MEAN OF THE RESULTS, AS IN [2], OF MANIFOLD-BASED ALGORITHMS. BOLD FACE DENOTES THE BEST RESULT

	10 labels						100 labels					
Algorithm	g241c	g241d	Digit1	USPS	COIL	BCI	g241c	g241d	Digit1	USPS	COIL	BCI
1NN	44.05	43.22	23.47	19.82	65.91	48.74	40.28	37.49	6.12	7.64	23.27	44.83
SVM	47.32	46.66	30.6	20.03	68.36	49.85	23.11	24.64	5.53	9.75	22.93	34.31
MVU + 1NN	48.68	47.28	11.92	14.88	65.72	50.24	44.05	43.21	3.99	6.09	32.27	47.42
LEM + 1NN	47.47	45.34	12.04	19.14	67.96	49.94	42.14	39.43	**2.52**	6.09	36.49	48.64
QC + CMN	39.96	46.55	9.8	**13.61**	59.63	50.36	22.05	28.2	3.15	6.36	10.03	46.22
Discrete Reg.	49.59	49.05	12.64	16.07	63.38	49.51	43.65	41.65	2.77	**4.68**	**9.61**	47.67
SGT	22.76	**18.64**	8.92	25.36	n/a	49.59	17.41	9.11	2.61	6.8	n/a	45.03
Laplacian RLS	43.95	45.68	**5.44**	18.99	**54.54**	48.97	24.36	26.46	2.92	**4.68**	11.92	**31.36**
CHM (normed)	39.03	43.01	14.86	20.53	n/a	46.9	24.82	25.67	3.79	7.65	n/a	36.03
ClusterReg	**16.90**	40.82	12.06	19.42	65.51	**45.36**	**13.38**	**4.36**	3.45	5.25	24.73	33.92

TABLE II

AVERAGE OF ERRORS (%) OF THE RUNS WITH 12 SUBSETS OF LABELED DATA. FOR ALL THE ALGORITHMS, THE TEST SETS ARE FIXED. THE TABLE REPORTS ONLY THE MEAN OF THE RESULTS, AS IN [2], OF CLUSTER-BASED ALGORITHMS. BOLD FACE DENOTES THE BEST RESULT

	10 labels						100 labels					
Algorithm	g241c	g241d	Digit1	USPS	COIL	BCI	g241c	g241d	Digit1	USPS	COIL	BCI
1NN	44.05	43.22	23.47	19.82	65.91	48.74	40.28	37.49	6.12	7.64	23.27	44.83
SVM	47.32	46.66	30.6	20.03	68.36	49.85	23.11	24.64	5.53	9.75	22.93	34.31
TSVM	24.71	50.08	17.77	25.2	67.5	49.15	18.46	22.42	6.15	9.77	25.8	**33.25**
Cluster-Kernel	48.28	42.05	18.73	19.41	67.32	48.31	13.49	4.95	3.79	9.68	21.99	35.17
Data-Rep. Reg.	41.25	45.89	12.49	17.96	63.65	50.21	20.31	32.82	**2.44**	5.1	**11.46**	47.47
LDS	28.85	50.63	15.63	**15.57**	**61.9**	49.27	18.04	28.74	3.46	**4.96**	13.72	43.97
ClusterReg	**16.90**	**40.82**	**12.06**	19.42	65.51	**45.36**	**13.38**	**4.36**	3.45	5.25	24.73	33.92

and the mean error is reported. As the test sets are fixed, we directly compare the generalization error, as done in [2]. The details of the generation procedure of these datasets and the experimental setup for all the other algorithms mentioned here can be found in [2, Ch. 21].

For ClusterReg, we performed a grid search for the best combination of the parameters K, V, κ, and the clustering algorithm as described later in Section IV-C. The other parameters are fixed for all datasets: $\lambda = 0.2$, 10 hidden neurons, and 50 epochs. The predictions of ClusterReg for the first subset of 100 labeled points of each dataset are presented in Fig. 6(b) and (d).

The results are grouped in cluster-based and manifold-based methods, so that we can analyze ClusterReg among algorithms working under the same assumption. Table I presents the results of manifold-based classifiers and ClusterReg, and Table II gives the methods based on the cluster assumption. Both tables report the results when there are very few (10) and more (100) labeled points.

B. Inductive Settings

We selected 14 datasets from the UCI machine learning repository [24]. Table III summarizes each dataset employed. Here we present an inductive setup of the experiments, which means, in contrast to transductive learning, that the classifiers are able to predict the labels of unseen instances.

As the amount of labeled instances plays an important role in the performance of the classifiers, in the experimental setup

TABLE III
SUMMARY OF DATASETS

Datasets	No. of instances	No. of attributes	No. of classes
BUPA	345	6	2
German credit	1000	24	2
Harberman	306	3	2
Horse colic	368	27	2
House Votes	435	16	2
Ionosphere	351	34	2
Pima Indians	768	8	2
WDBC	569	30	2
Contraceptive	1473	9	3
Heart-cleveland	303	13	5
SPECT	267	22	2
Statlog	846	18	4
Transfusion	748	4	2
Yeast	1479	8	9

we analyze three different scenarios: when the proportion (l/N) is 5%, 10%, and 20%. We transformed these problems into semisupervised tasks by randomly selecting the proper amount of labeled instances for each dataset. The labeled instances of each dataset are different for each ratio, so that each dataset presents itself as a different problem for each ratio (l/N).

We performed 10-fold cross-validation for all datasets. As we are dealing with real-world datasets, we do not know the true class structure and the corresponding assumption in SSC. Intuitively, if the dataset has a manifold-like structure, it is expected algorithms that use the correct assumptions to deliver a better performance when compared to other SSC algorithms [2]. We also expect that ensemble-based algorithms that run with more than two assumptions may yield higher average performance throughout the datasets [17].

We compare our method to algorithms with different assumptions (all methods employ the smoothness assumption): one classifier based on the manifold assumption, i.e., SGT; one based on the cluster assumption, i.e., TSVM; and two ensemble classifiers that work under the three SSC assumptions, i.e., MCSSB and RegBoost.

C. Parameter Tuning

For SGT algorithm, we used all the parameter combinations from the following setup, making a broader search than in [9]: the number of neighbors $k \in (10, 50, 100)$; the number of first eigenvectors is $d \in (10, 40, 80, 100)$; and the error parameter $c \in (10^0, 10^2, 10^3, 10^4)$. Although, in [9], the parameter c was set between 3200 and 12 800, our preliminary experiments presented better results within our setup. Then, we chose the combinations of parameters with the best result for each dataset.

TSVM follows the same assumption as our method. Therefore, if the dataset presents a real cluster structure, we expect ClusterReg to deliver more accurate results. In the case where there is no such characteristic, both algorithms may have similar performance. For TSVMs settings, following the setup in [2], we used a radial basis function (RBF) kernel. The kernel's width parameter was chosen as the median of the pairwise distances of the instances [2]. Unlike in [2], we decided to make a broader search with $C \in (10^0, 10^1, 10^2, 10^3)$. In preliminary experiments, lower values of C made the algorithm too slow and less accurate, and higher values did not deliver any better results. We performed a grid search with all combinations of parameters and picked the ones with the best result for each dataset.

TABLE IV
SUMMARY OF PARAMETER SETTING

Clustering algorithm	Grid search with K-means, GMM, STSC, or Fuzzy GK
λ	0.2
V	Grid search in (5, 10, 30)
K	Grid search in 1, 2, and 3 times the number of classes
κ	Grid search in (2, 5, 9, 12)

The third method is MCSSB ensemble. It uses all three SSC assumptions, so we expect our algorithm to outperform MCSSB only on some datasets where there is, in fact, a meaningful cluster structure. As base classifier, we chose the decision tree, as indicated in [10]. We fixed the parameter[7] $C = 10\ 000$. The percentage of the range of distances used for kernel construction was searched in $\sigma \in (0.01, 0.05, 0.1, 0.15, 0.2, 0.25, 0.5, 0.8, 1)$. We also adjusted the sample size as $s \in (0.1, 0.5, 0.8, 1)$. The number of weak learners was 20 and 50. All the parameter combinations were tested, and the best result for each dataset is reported.

For the parameters in RegBoost, we followed the guidelines in [17] to perform grid search for the best combination of parameters. The number of iterations was 20 and 50. The number of neighbors was in $(3, 4, 5, 6)$. The resampling rate in the first iteration was fixed to 0.1. And the resampling rate in the rest of iterations was in $(0.1, 0.25, 0.5)$. According to our preliminary experiments, we chose SVM as the base classifier.

For ClusterReg, the parameter λ controls the amount of regularization in the algorithm. As in [17], we set this trade-off parameter to one-fifth, i.e., $\lambda = 0.2$, of the importance of supervised error, as we do not know whether the data hold the cluster assumption. It is advisable to set this value between 0 and 1.

Our preliminary experiments showed that the number of neighbors V can be set to 30 for most datasets used in this paper. For datasets not larger than 1500 instances, this number might represent a comprehensive search for labels in the neighborhood of an instance. For a small number of neighbors, ClusterReg may not capture the correct label structure of the neighborhood. For datasets larger than 1500 instances, V could be set to around 2% of the number of instances.

We employed four clustering methods from different clustering approaches: K-means, STSC, GMM, and fuzzy GK. We selected this clustering algorithm by cross-validation, since the

[7] As demonstrated in [10] and confirmed by our preliminary experiments, this value should be set to 10 000. Lower and higher values did not improve the performance.

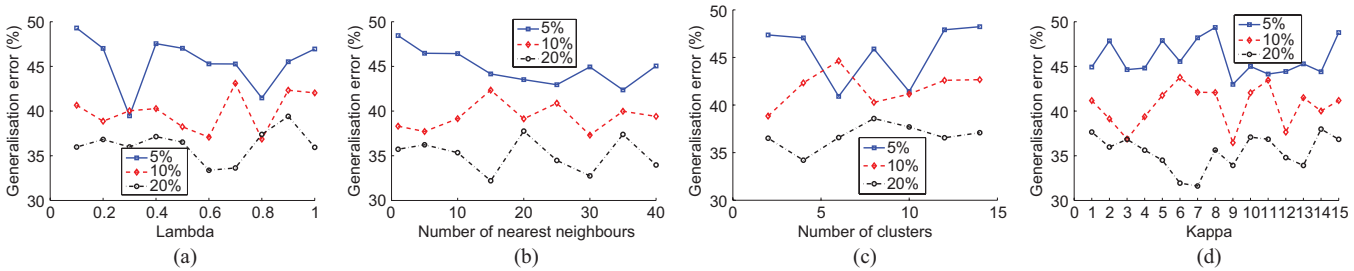

Fig. 7. Generalization error from 10-fold cross-validation with different values of λ, V, K, and κ for BUPA dataset. (a) λ. (b) Number of nearest neighbours V. (c) Number of Clusters K. (d) κ.

TABLE V

MEAN AND STANDARD DEVIATION (%) OF 10-FOLD CROSS-VALIDATION ERROR AT 5% OF LABELED DATA. •/◦ INDICATES WHETHER CLUSTERREG IS STATISTICALLY SUPERIOR/INFERIOR TO THE COMPARED METHOD, ACCORDING TO PAIRWISE T-TEST AT 95% SIGNIFICANCE LEVEL. WIN/TIE/LOSS DENOTES THE NUMBER OF DATASETS WHERE CLUSTERREG IS SIGNIFICANTLY SUPERIOR/EQUAL/INFERIOR TO THE COMPARED ALGORITHM

Datasets	SGT	TSVM	MCSSB	RegBoost	ClusterReg
BUPA	44.95 ± 7.89•	44.60 ± 6.21•	42.04 ± 8.70	49.36 ± 9.99•	37.70 ± 6.76
German credit	31.50 ± 2.68	31.30 ± 3.92	33.70 ± 7.24	29.90 ± 3.11	30.00 ± 4.27
Haberman	34.34 ± 7.67	40.20 ± 7.37•	26.17 ± 7.75◦	31.37 ± 7.29	33.32 ± 10.11
Horse colic	37.21 ± 6.62	42.93 ± 7.37•	33.14 ± 6.76	33.43 ± 4.47	33.97 ± 7.47
House votes	8.04 ± 5.80	9.41 ± 4.71	19.52 ± 8.14•	10.81 ± 4.47	8.74 ± 5.37
Ionosphere	35.34 ± 9.77•	25.06 ± 9.76•	33.32 ± 8.11•	33.58 ± 15.26•	11.71 ± 6.10
Pima Indians	32.28 ± 5.13	36.33 ± 5.12	34.89 ± 5.98	32.53 ± 7.06	35.40 ± 5.67
WDBC	8.44 ± 2.96•	11.07 ± 5.22•	32.13 ± 9.81•	6.15 ± 4.47	4.57 ± 3.90
Contraceptive	57.65 ± 4.93•	52.15 ± 5.67	62.25 ± 6.02•	57.31 ± 3.84•	52.15 ± 5.02
Heart-Cleveland	44.89 ± 9.96	45.23 ± 8.50	45.20 ± 9.09	45.86 ± 7.74	42.58 ± 9.07
SPECT	28.86 ± 5.07•	20.17 ± 6.96	79.49 ± 6.86•	39.70 ± 8.59•	15.31 ± 7.28
Statlog	42.32 ± 2.81•	40.30 ± 5.45•	74.59 ± 4.34•	74.00 ± 3.93•	36.64 ± 4.79
Transfusion	29.81 ± 19.24	29.16 ± 6.48•	23.81 ± 6.10	24.61 ± 6.46	22.87 ± 4.70
Yeast	49.63 ± 3.31•	44.96 ± 4.85	69.10 ± 3.35•	69.04 ± 3.21•	45.98 ± 5.64
Win/Tie/Loss	7/7/0	7/7/0	7/6/1	6/8/0	/

performance of these algorithms varies depending on the real underlying structure of the dataset and the type of partition that these methods attempt to find.[8] However, our experiments demonstrated that the STSC algorithm usually obtains good results for most datasets. This might indicate that most of these datasets have clusters with arbitrary shapes that the other algorithms cannot find. Therefore, we suggest the use of STSC as the clustering algorithm for ClusterReg.

For the number of clusters K, we recommend to set it, at least, to the number of classes. We intend to generate clusters that are as compact as possible. If the class structure is not captured by the clustering algorithm, we can increase the number of clusters, so that one class is composed of multiple clusters. The algorithm will avoid splitting these clusters and, therefore, may be able to place the decision boundary outside the class. According to our preliminary experiments, we recommend, in general, to set K to two times the number of classes.

The parameter κ controls the importance of each neighbor according to their similarity (conforming to the clustering algorithm) to an instance. With a larger κ, we relax the cluster assumption by allowing the decision boundary to cut through relatively distant neighbors. It regulates the size of the portion of a cluster that we allow the decision boundary to cut through. According to our preliminary experiments, it should be set between 1 and 12—values in the middle of this range often deliver good performance. ClusterReg's performance degrades, for all datasets, with values outside this range. According to our preliminary experiments, one could use $\kappa = 5$. In Table IV, we summarize the tuning of each parameter in ClusterReg.

In Fig. 7, we show the behavior of the generalization error for different values of λ, V, K, and κ. We selected only a subset of these values that roughly yielded good performance to be set in our experiments. Then, we tuned four parameters for the proposed method: 1) the clustering algorithm was chosen from K-means, GMM, STSC, and fuzzy GK; 2) the number of cluster was set as 1, 2, or 3 times the number of classes; 3) the number of neighbors was picked from $(5, 10, 30)$; and 4) and $\kappa = (2, 5, 9, 12)$. The parameter λ was fixed at 0.2, and the number of hidden nodes and epochs were 15 and 50, respectively.

D. Inductive Results

Tables V–VII show the mean and standard deviation of generalization error of all algorithms for all datasets with 5%, 10%, and 20% of labeled data, respectively. We employ a pairwise t-test with 95% of significance level to compare the algorithms to ClusterReg, as shown in these tables.

[8] K-means tends to generate hyperspherical clusters [5]. GMM and fuzzy GK are able to obtain elliptical clusters, whereas STSC is capable of finding clusters with arbitrary shapes.

TABLE VI

MEAN AND STANDARD DEVIATION (%) OF 10-FOLD CROSS-VALIDATION ERROR AT 10% OF LABELED DATA. •/◦ INDICATES WHETHER CLUSTERREG IS STATISTICALLY SUPERIOR/INFERIOR TO THE COMPARED METHOD, ACCORDING TO PAIRWISE T-TEST AT 95% SIGNIFICANCE LEVEL. WIN/TIE/LOSS DENOTES THE NUMBER OF DATASETS WHERE CLUSTERREG IS SIGNIFICANTLY SUPERIOR/EQUAL/INFERIOR TO THE COMPARED ALGORITHM

Datasets	SGT	TSVM	MCSSB	RegBoost	ClusterReg
BUPA	34.43 ± 8.73	40.83 ± 6.96•	42.32 ± 9.65•	40.62 ± 8.91•	33.30 ± 5.24
German credit	28.10 ± 6.05	34.70 ± 6.31	30.70 ± 4.35	29.50 ± 4.25	31.90 ± 6.19
Haberman	32.39 ± 11.42	37.62 ± 9.99•	26.48 ± 7.02	26.47 ± 7.63	29.44 ± 7.96
Horse colic	32.36 ± 6.59	35.02 ± 6.10•	33.69 ± 5.05	35.04 ± 5.65•	29.61 ± 7.54
House votes	6.19 ± 3.04	9.40 ± 4.69	15.84 ± 5.55•	7.56 ± 3.86	8.26 ± 3.40
Ionosphere	24.75 ± 8.14•	19.10 ± 7.29•	35.92 ± 10.60•	29.04 ± 16.92•	8.27 ± 5.63
Pima indians	31.38 ± 5.23•	25.65 ± 4.41	34.90 ± 7.54•	27.73 ± 6.83	24.35 ± 3.38
WDBC	8.97 ± 3.19•	6.33 ± 3.83	25.32 ± 9.80•	5.98 ± 3.33	4.39 ± 3.72
Contraceptive	55.06 ± 2.73•	52.21 ± 3.41	72.37 ± 4.51•	57.29 ± 4.05•	50.58 ± 3.47
Heart-Cleveland	38.92 ± 3.77◦	50.44 ± 6.44	62.72 ± 13.36•	45.84 ± 7.08	47.24 ± 6.38
SPECT	21.35 ± 8.95	19.10 ± 7.91	79.39 ± 5.72•	50.51 ± 7.33•	15.34 ± 7.40
Statlog	40.08 ± 5.18•	32.86 ± 4.66	72.32 ± 8.88•	74.94 ± 4.76•	31.21 ± 5.99
Transfusion	20.98 ± 4.58	29.55 ± 5.45•	23.78 ± 4.37	23.78 ± 4.37	21.78 ± 5.03
Yeast	40.57 ± 3.46	43.48 ± 5.12	67.95 ± 3.83•	68.69 ± 4.21•	42.94 ± 4.09
Win/Tie/Loss	5/8/1	5/9/0	10/4/0	7/7/0	/

TABLE VII

MEAN AND STANDARD DEVIATION (%) OF 10-FOLD CROSS-VALIDATION ERROR AT 20% OF LABELED DATA. •/◦ INDICATES WHETHER CLUSTERREG IS STATISTICALLY SUPERIOR/INFERIOR TO THE COMPARED METHOD, ACCORDING TO PAIRWISE T-TEST AT 95% SIGNIFICANCE LEVEL. WIN/TIE/LOSS DENOTES THE NUMBER OF DATASETS WHERE CLUSTERREG IS SIGNIFICANTLY SUPERIOR/EQUAL/INFERIOR TO THE COMPARED ALGORITHM

Datasets	SGT	TSVM	MCSSB	RegBoost	ClusterReg
BUPA	33.03 ± 9.26	35.66 ± 7.53	44.37 ± 7.87•	42.60 ± 7.58•	31.03 ± 6.69
German credit	26.00 ± 5.42	30.20 ± 4.64	29.80 ± 4.42	26.80 ± 5.55	28.40 ± 3.47
Haberman	24.91 ± 11.19	30.10 ± 8.24	48.88 ± 24.45•	27.49 ± 7.56	26.22 ± 11.79
Horse colic	27.99 ± 5.12	33.94 ± 8.40	34.01 ± 10.76	33.46 ± 10.40	29.33 ± 6.68
House votes	6.21 ± 4.18	5.98 ± 3.11	33.56 ± 13.18•	4.82 ± 3.32	5.29 ± 3.77
Ionosphere	16.54 ± 6.02•	13.95 ± 5.43	34.75 ± 7.92•	11.10 ± 5.92	10.53 ± 4.64
Pima Indians	29.18 ± 7.15•	25.13 ± 5.75	34.90 ± 3.39•	26.18 ± 7.76	22.91 ± 4.63
WDBC	9.31 ± 4.21•	5.45 ± 3.03•	29.69 ± 11.62•	5.28 ± 2.49•	2.82 ± 1.90
Contraceptive	50.38 ± 2.95•	51.39 ± 3.87•	62.66 ± 4.84•	57.30 ± 4.04•	47.05 ± 3.69
Heart-Cleveland	37.31 ± 6.91	46.87 ± 11.75	45.53 ± 7.84	45.86 ± 9.12	43.55 ± 9.65
SPECT	16.89 ± 7.23	18.75 ± 5.98	79.42 ± 3.06•	32.98 ± 3.58•	18.75 ± 6.78
Statlog	31.91 ± 4.50•	23.88 ± 4.62	70.92 ± 6.16•	74.95 ± 3.00•	22.33 ± 3.08
Transfusion	20.32 ± 4.41	25.94 ± 4.05•	23.80 ± 5.43	35.43 ± 5.52•	21.13 ± 4.56
Yeast	38.95 ± 4.00	42.12 ± 2.16	70.65 ± 5.85•	68.70 ± 4.16•	40.84 ± 3.80
Win/Tie/Loss	5/9/0	3/11/0	10/4/0	7/7/0	/

In order to provide more statistical evidences, we also use the Bonferroni–Dunn statistical test [25]. Such a test compares, with a high confidence level, multiple algorithms to a control classifier, i.e., ClusterReg, on multiple datasets. The performance of a classifier is significantly different from the control classifier if their average ranks[9] over all datasets differ by at least the critical difference (CD). According to [26], a threshold of 0.10 is sufficient to perform the statistical comparison. With 5 classifiers, 14 datasets, and a threshold of 0.10, the critical difference is 1.34. Table VIII presents the average ranks of each algorithm over all datasets and Table IX shows the differences of average ranks of each classifier to ClusterReg.

[9]The rank of each algorithm is based on its generalization error in each dataset. Average ranks are assigned in case of ties. The average ranks for each algorithm are given in Table VIII.

TABLE VIII

AVERAGE RANKS OF THESE ALGORITHMS WITH DIFFERENT AMOUNTS OF LABELED DATA

% of labeled data	SGT	TSVM	MCSSB	RegBoost	ClusterReg
5	3.29	3.14	3.57	3.21	1.79
10	2.36	3.36	4.21	3.21	1.79
20	2.14	3.32	4.57	3.29	1.68

E. Computation Time

We also measure the computation time of ClusterReg. In Fig. 8, we plot the CPU time of each method for 5%, 10%, and 20% of labeled data, so that we can compare the behavior of each method under different amounts of labels for each dataset. Each computation time reported is the average time and its standard deviation of the 10-fold cross-validation executions

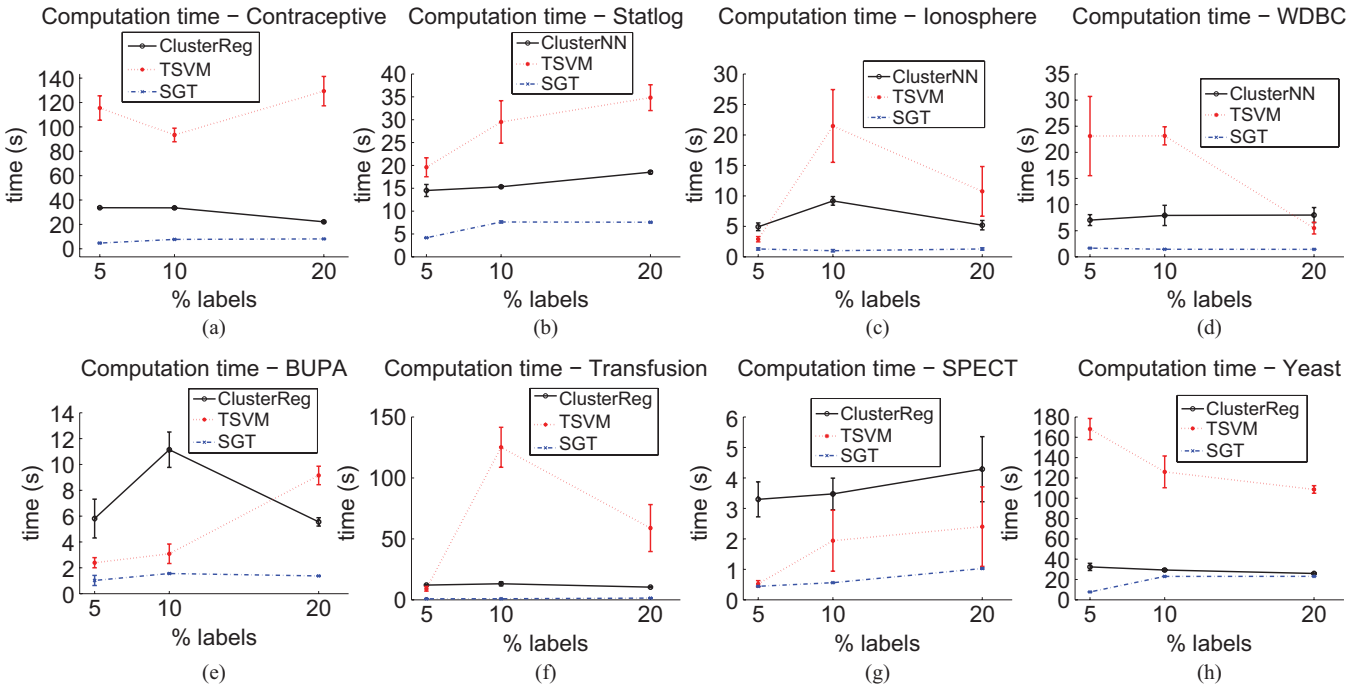

Fig. 8. Plots of mean and standard deviation of the computation time of 10-fold cross-validation executions for 5%, 10%, and 20% of labeled data, on the datasets where ClusterReg obtained the best results. (a) Contraceptive. (b) Statlog. (c) Ionosphere. (d) WDBC. (e) BUPA. (f) Transfusion. (g) SPECT. (h) Yeast.

TABLE IX

BONFERRONI–DUNN STATISTICAL TESTS TO COMPARE MULTIPLE CLASSIFIERS TO CLUSTERREG ON MULTIPLE DATASETS. WE SHOW THE DIFFERENCE IN AVERAGE RANK OF EACH ALGORITHM TO CLUSTERREG. IF SUCH DIFFERENCE IS HIGHER THAN THE CRITICAL DIFFERENCE, CLUSTERREG IS SIGNIFICANTLY SUPERIOR TO THE ALGORITHM. THE CRITICAL DIFFERENCE FOR THRESHOLD OF 0.10 WITH 5 ALGORITHMS (INCLUDING CLUSTERREG) AND 14 DATASETS IS 1.34

Percent of labeled data	SGT	TSVM	MCSSB	RegBoost
5	1.5	1.36	1.79	1.43
10	0.57	1.57	2.43	1.43
20	0.46	1.64	2.89	1.61

that delivered the error rates already shown in Tables V–VII. Specifically, we picked eight datasets where ClusterReg obtained the best performance to show the computation time, namely, Contraceptive, Statlog, Ionosphere, WDBC, BUPA, Transfusion, SPECT, and Yeast (Fig. 8(a)–(h), respectively). In order to make a fair comparison, we use only the single classifiers used in the inductive setup: SGT, TSVM, and ClusterReg.

We measured the CPU time of all algorithms in an Intel Core 2 Quad CPU Q8200 with 2 GB of memory. ClusterReg was implemented using MATLAB. The implementation can be further optimized.

V. DISCUSSION

In the transductive experiments, we analyze two types of algorithms: manifold and cluster-based classifiers. Both datasets present a difficult task for manifold-based algorithms since they do not satisfy manifold assumption. In contrast, g241c is designed as a suitable problem to cluster-based classifiers, while g241d and Digit1 are challenging problems to the existing cluster-based algorithms in the literature.

Therefore, ClusterReg outperformed the manifold-based algorithms on g241c and g241d, and, as expected, delivered better generalization performance than all other cluster-based classifiers on both datasets. The exception was SGT in g241d with 10 labels. This might indicate, for this case, that the graph neighborhood built by SGT properly represents the underlying data structure. ClusterReg also yielded competitive performance among cluster-based and manifold-based algorithms on the real-world datasets with 10 and 100 labeled instances. In particular, ClusterReg had good performance on BCI when compared to both manifold and cluster-based algorithms. This might indicate that, in this case, ClusterReg was able to properly use the information of few labeled instances.

We have a dataset with a clear cluster structure (g241c), where the cluster-based methods should perform sufficiently well, and datasets tailored to misguide such algorithms (g241d and Digit1). So, when compared to these algorithms, ClusterReg manages to make better use of the cluster structure and the few labeled instances available to find a suitable decision boundary. It is important to highlight that the results presented in Table II were achieved using the K-means algorithm with five replicates with random initialization. This may indicate that the other methods fail to find the correct gap between classes while a simple clustering algorithm can find the clusters. This fact demonstrates how useful clustering techniques can be for semisupervised classifier under the cluster assumption.

It is important to point out that ClusterReg is more robust than TSVM when classes do not correspond to clusters, which is the case for g241d and Digit1, shown in Table II. This fact may indicate that the proposed classifier is able to exploit the

information from the few labeled data in a more effective way than TSVM, since the unlabeled data do not bring very useful knowledge to cluster-based classifiers.

For the inductive setting, Tables V–VII show the generalization error and statistical test results for the employed algorithms with the presence of 5%, 10%, and 20% of labeled data, respectively. And Table IX presents the results for the Bonferroni–Dunn statistical test.

When compared to the ensemble methods MCSSB and RegBoost, ClusterReg delivered significantly better results under all amounts of labeled data, as confirmed by pairwise t-test and Bonferroni–Dunn test, in Tables V–VII and IX, respectively. Besides being ensemble approaches, these classifiers differ from ClusterReg mainly in the use of SSC assumptions. They use both manifold and cluster assumptions. When only one of them is present and/or the other assumptions are misleading, a more specialized algorithm, such as ClusterReg, might be more effective. Moreover, RegBoost seemed to be affected by the number of classes: for binary problems, it delivered better accuracy than in multiclass problems. ClusterReg is inherently multiclass and did not show such shortcoming.

ClusterReg showed similar results to SGT with 10% and 20% of labeled data. And, for 5% of labeled points, it performed significantly better, as confirmed by the t-test and Bonferroni–Dunn test. This indicates that ClusterReg is more robust to few labeled instances. We expected to have contrasting results to SGT across the datasets, as the actual structures of real-world datasets are unknown. However, these results may suggest that ClusterReg makes better use of the labeled instances when data distribution does not help enough.

Similar to ClusterReg, TSVM also possesses the cluster assumption. However, in the case where the cluster assumption holds, we expect ClusterReg to perform better when very few instances are available. As mentioned before, ClusterReg is more robust than TSVM to the position of the few labeled instances in the cluster, as it uses the clustering partition to find the decision boundary, whereas TSVM seeks the largest margin between classes, which can lead to the wrong decision boundary in the presence of overlapping classes.

In fact, the pairwise t-test results support our expectations. For 20% of labeled data, ClusterReg performed statistically better on three problems. With 10%, it delivered significantly superior results in five cases. For 5% of labeled data, ClusterReg performed statistically better on seven datasets. Furthermore, according to the Bonferroni–Dunn test (Table IX), ClusterReg is statistically superior to TSVM for all amounts of labeled data. This means that, when compared to the cluster-based algorithm TSVM, our proposed method can make better use of labeled instances and it is more robust for overlapping classes and misleading cluster structures with few labeled instances.

Even when the datasets do not follow cluster assumption, the experiments suggest that ClusterReg could still outperform TSVM. This is due to the balance of two terms in the loss function. When the cluster assumption does not hold, the second term in (8) might not be reliable; however, the first term may be able to compensate such a misleading value more effectively than TSVM does. That is, the experiments indicate that the supervised learning in ClusterReg may be more effective than in TSVM.

Regarding the computation time, SGT is the least time consuming method. However, since SGT has the manifold assumption, it may not be suitable for datasets where there is a cluster structure [2]. Focusing on the context of cluster-based methods, ClusterReg presented a competitive performance when compared to the cluster-based classifier TSVM on most of the eight datasets with different amounts of labeled data, as shown in Fig. 8. Furthermore, we can notice that the difference of execution time across 5%, 10%, and 20% of labeled data for ClusterReg is not as high as in TSVM, which might indicate that the computation time of ClusterReg is more stable under different amounts of labels.

All the experiments of ClusterReg were performed with a fixed number of hidden nodes and epochs. The results may be improved with a fine-tuning of these parameters. However, the computation time is likely to increase as these parameters change for greater numbers. In order to make the algorithm less time consuming, we could employ faster classifiers, such as RBF networks.

VI. Conclusion

We proposed a new SSC algorithm that takes advantage of partitions resulting from a clustering algorithm and uses such information to regularize the training of a classifier. The transductive experimental setting, with synthetic and real-world datasets, assessed the generalization ability of the new method in different scenarios where the cluster assumption holds and when it is misleading. In the inductive case, we used real-world datasets with different ratios of labeled data to evaluate ClusterReg, along with other methods with various approaches to the SSC assumptions.

Both sets of experiments confirmed that the proposed method is able to improve generalization performance under various scenarios when the cluster assumption holds. Among the reasons for these improvements, we can point out ClusterReg's ability to deal with the potential presence of overlapping classes and its robustness to the particular situation of each labeled instance in the clusters.

When the cluster assumption is not satisfied, the performance of cluster-based approaches will degrade to some extent. In future work, we intend to design a cluster-based technique that is able to relax the cluster assumption so that its accuracy does not decrease significantly with the number of labeled instances, as noticed in other SSC algorithms. Furthermore, SSC methods often have to handle very large amounts of data, and hence efficiency is important. Therefore, we also aim to decrease the computation time of ClusterReg by employing faster classifiers and clustering algorithms.

References

[1] X. Zhu, "Semi-supervised learning literature survey," Dept. Comput. Sci., Univ. Wisconsin-Madison, Madison, Tech. Rep. TR-1530, 2008.

[2] O. Chapelle, B. Schölkopf, and A. Zien, *Semi-Supervised Learning*. Cambridge, MA: MIT Press, 2006.

[3] T. Joachims, "Learning to classify text using support vector machines," M.S. thesis, Dept. Comput. Sci., Cornell Univ., Ithaca, NY, 2002.

[4] L. Z. Manor and P. Perona, "Self-tuning spectral clustering," in *Advances in Neural Information Processing Systems*. Cambridge, MA: MIT Press, 2004, pp. 1601–1608.
[5] R. Xu and D. Wunsch, "Survey of clustering algorithms," *IEEE Trans. Neural Netw.*, vol. 16, no. 3, pp. 645–678, May 2005.
[6] D. Gustafson and W. Kessel, "Fuzzy clustering with a fuzzy covariance matrix," in *Proc. IEEE Conf. Decision Control*, San Diego, CA, Jan. 1979, pp. 761–766.
[7] A. Blum and T. Mitchell, "Combining labeled and unlabeled data with co-training," in *Proc. 11th Annu. Conf. Comput. Learn. Theory*, 1998, pp. 92–100.
[8] Z.-H. Zhou and M. Li, "Tri-training: Exploiting unlabeled data using three classifiers," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 11, pp. 1529–1541, Nov. 2005.
[9] T. Joachims, "Transductive learning via spectral graph partitioning," in *Proc. Int. Conf. Mach. Learn.*, 2003, pp. 290–297.
[10] H. Valizadegan, R. Jin, and A. K. Jain, "Semi-supervised boosting for multi-class classification," in *Proc. Eur. Conf. Mach. Learn. Knowl. Discovery Databases*, 2008, pp. 522–537.
[11] V. Vapnik, *Statistical Learning Theory*. New York: Wiley, 1998.
[12] S. Chakraborty, "Bayesian semi-supervised learning with support vector machine," *Stat. Methodol.*, vol. 8, no. 1, pp. 68–82, 2011.
[13] Y. Wang, S. Chen, and Z.-H. Zhou, "New semi-supervised classification method based on modified cluster assumption," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 5, pp. 689–702, May 2012.
[14] P. K. Mallapragada, R. Jin, A. K. Jain, and Y. Liu, "SemiBoost: Boosting for semi-supervised learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 11, pp. 2000–2014, Nov. 2009.
[15] L. Zheng, S. Wang, Y. Liu, and C.-H. Lee, "Information theoretic regularization for semi-supervised boosting," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2009, pp. 1017–1026.
[16] M. F. A. Hady, F. Schwenker, and G. Palm, "Semi-supervised learning for tree-structured ensembles of RBF networks with co-training," *Neural Netw.*, vol. 23, no. 4, pp. 497–509, 2010.
[17] K. Chen and S. Wang, "Semi-supervised learning via regularized boosting working on multiple semi-supervised assumptions," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 1, pp. 129–143, Jan. 2011.
[18] H. Chen, P. Tiňo, and X. Yao, "Predictive ensemble pruning by expectation propagation," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 7, pp. 999–1013, Jul. 2009.
[19] H. Chen, P. Tiňo, and X. Yao, "A probabilistic ensemble pruning algorithm," in *Proc. 6th Int. Conf. Workshops Optim.-Based Data Mining Tech. Appl.*, 2006, pp. 878–882.
[20] C. M. Bishop, *Pattern Recognition on Data Mining and Machine Learning*. New York: Springer-Verlag, 2006.
[21] N. A. C. D. Campbell and R. A. Dunne, "On the pairing of the softmax activation and cross entropy penalty functions and the derivation of the softmax activation function," in *Proc. 8th Austral. Conf. Neural Netw.*, 1997, pp. 181–185.
[22] M. Kline and L. Berardi, "Revisiting squared-error and cross-entropy functions for training neural network classifiers," *Neural Comput. Appl.*, vol. 14, pp. 310–318, Dec. 2005.
[23] M. F. Moller, "A scaled conjugate gradient algorithm for fast supervised learning," *Neural Netw.*, vol. 6, no. 4, pp. 525–533, 1993.
[24] A. Frank and A. Asuncion. (2010). *UCI Machine Learning Repository* [Online]. Available: http://archive.ics.uci.edu/ml
[25] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, Dec. 2006.
[26] H. Chen, P. Tino, and X. Yao, "Probabilistic classification vector machines," *IEEE Trans. Neural Netw.*, vol. 20, no. 6, pp. 901–914, Jun. 2009.

Rodrigo G. F. Soares received the B.Sc. degree in computer engineering from the Federal University of Rio Grande do Norte, Brazil, and the M.Phil. degree in computer science from the Federal University of Pernambuco, Brazil, in 2005 and 2008, respectively. He is currently pursuing the Ph.D. degree in computer science with the University of Birmingham, Edgbaston, U.K.

His current research interests include semisupervised learning, ensemble learning, clustering, and evolutionary computation.

Mr. Soares is a recipient of a scholarship from the Capes Foundation, Brazil, and the Brazilian Council for Scientific and Technological Development Scholarship.

Huanhuan Chen (M'09) received the B.Sc. degree from the University of Science and Technology of China, Hefei, China, in 2004, and the Ph.D. degree in computer science with the thesis "Diversity and Regularization in Neural Network Ensembles" from the University of Birmingham, Edgbaston, U.K., in 2008.

His current research interests include statistical machine learning, data fusion, data mining, and evolutionary computation.

Dr. Chen was a recipient of the Dorothy Hodgkin Postgraduate Award, the IEEE TRANSACTIONS ON NEURAL NETWORKS Outstanding Paper Award in 2011, the IEEE Computational Intelligence Society Outstanding Ph.D. Dissertation Award in 2011, the CPHC/British Computer Society Distinguished Dissertations Award in 2009.

Xin Yao (F'03) is currently the Chair (Professor) of computer science with the University of Birmingham, Edgbaston, U.K., where he is the Director of the Centre of Excellence for Research in Computational Intelligence and Applications. He is the Director of the Joint USTC-Birmingham Research Institute of Intelligent Computation and Its Applications. He has been invited to give more than 65 keynote and plenary speeches at international conferences in many different countries. He has authored or co-authored more than 400 refereed publications in international journals and conferences. His H-index is 57 and total citations 16 366. His current research interests include evolutionary computation and neural network ensembles.

Prof. Yao is a Distinguished Lecturer of the IEEE Computational Intelligence Society. He was a recipient of the IEEE D. G. Fink Prize Paper Award in 2001, the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION Outstanding Paper Award in 2010, the BT Gordon Radley Award for Best Author of Innovation in 2010, the IEEE TRANSACTIONS ON NEURAL NETWORKS Outstanding Paper Award in 2011, the Prestigious Royal Society Wolfson Research Merit Award in 2012, and many other best paper awards at conferences. He has been selected for the IEEE CIS Evolutionary Computation Pioneer Award in 2013. He was the Editor-in-Chief of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION from 2003 to 2008.