Hierarchical Bottom-Up Safe Semi-Supervised Support Vector Machines for Multi-Class Transductive Learning

Thiago F. Covões, Rodrigo C. Barros, Tiago S. da Silva, Eduardo R. Hruschka, Andre C. P. L. F. de Carvalho

Instituto de Ciências Matemáticas e de Computação — ICMC
Universidade de São Paulo — Campus São Carlos
Caixa Postal 668, 13560-970 São Carlos-SP
{tcovoes, rcbarros, tiago.silva, erh, andre}@icmc.usp.br

Abstract. Semi-supervised approaches have been successfully applied to many machine learning problems. A particular case of semi-supervised settings is transductive learning, in which the goal is solely to label the available unlabeled data, instead of generating a predictive model that should generalize well to unseen data. Nevertheless, there are cases in which a transductive learner may perform even worse than an inductive learner that is trained with only the labeled data. For alleviating this problem, an effort towards safe semi-supervised support vector machines (S^4VM) was made, so that a transductive SVM would never degenerate the performance when compared to its inductive counterpart. Even though robust, S^4VM still lacks the ability of naturally dealing with multi-class problems, having to rely on multi-class encoding schemes, such as one-vs-one and one-vs-all strategies. These schemes may not take advantage of the full potential of S^4VM, especially in complex multi-class problems with overlapping classes. In this article, we address this problem by providing a binary-tree scheme for aggregating distinct S^4VMs in a bottom-up fashion. The proposed approach is named *HiBUST*, which stands for Hierarchical Bottom-Up S^4VM Tree. Experimental results show that *HiBUST* can provide increased predictive performance for many multi-class problems when compared to a one-vs-one S^4VM. In addition, we show that *HiBUST* is also beneficial for complex binary-class problems.

Categories and Subject Descriptors: H.2.8 [**Database Management**]: Database Applications—*Data mining*; I.5.2 [**Pattern Recognition**]: Design Methodology—*Classifier design and evaluation*

Keywords: multi-class SVMs, safe support vector machines, semi-supervised learning, transductive learning

1. INTRODUCTION

In transductive learning, given a labeled sample $\mathcal{X}^{(L)} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$ and an unlabeled sample $\mathcal{X}^{(U)} = \{\hat{\mathbf{x}}_j\}_{j=1}^{u}$ where $\mathbf{x}_i, \hat{\mathbf{x}}_j \in \mathbb{R}^d$ and $y_i \in \{1, \ldots, c\}$, the goal is to infer a function $f : \mathcal{X}^{(U)} \mapsto \mathcal{Y}^{(U)}$ so that f is expected to be a good predictor of the unlabeled data $\mathcal{X}^{(U)}$. Note that f is defined only on the given training sample, and it is not required to make predictions on a distinct set of data, *e.g.*, on a (new) test set [Chapelle et al. 1999; Zhu and Goldberg 2009; Zhang et al. 2010; Bahadori et al. 2011; Yu et al. 2012]. While the inductive approach is useful when a global model is needed in an approximate form, the transductive approach is more appropriate for applications where the focus is not on the generalization capability of the model, but rather on particular cases [Vapnik 1998; Gammerman et al. 1998].

Recently, a robust approach for transductive learning, named *Safe Semi-Supervised Support Vector Machines* (S^4VMs) [Li and Zhou 2011], has been proposed. S^4VMs exploit multiple candidate low-density separators simultaneously, instead of looking for a single optimal low-density separator

The authors would like to thank Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) and Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) for funding this research.

resulting from the set of labeled and unlabeled data. Considering that the amount of unlabeled data is usually much larger than that of labeled data, there exists more than one large-margin low-density separator that properly fits the data, and a poor choice may result in degenerated performance. The predictive accuracy of S^{4}VMs, on the other hand, is never significantly worse when compared to inductive SVMs [Li and Zhou 2011].

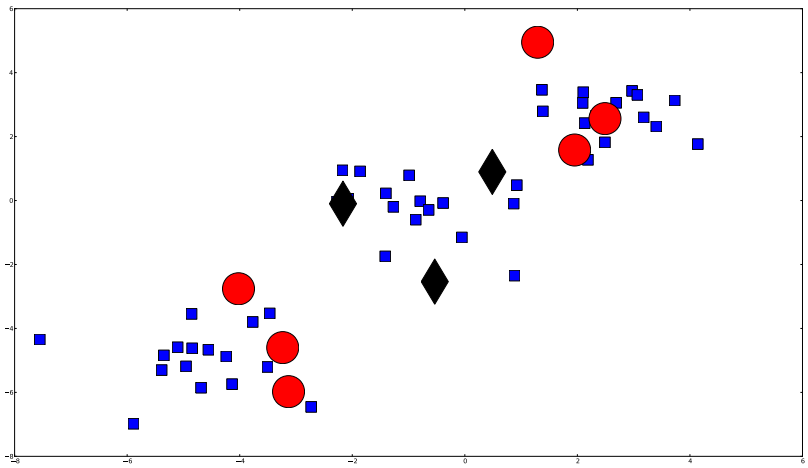
Even though S^{4}VMs present several advantages over other transductive learners, they do not naturally deal with multi-class problems. Due to various complexities, a direct mathematical solution for multi-class problems using a single SVM formulation is usually avoided [Duan and Keerthi 2005]. Popular strategies for multi-class SVM classification include *one-versus-all* classification with a *winner-takes-all* strategy (*OVA-WTA* SVM) and *one-versus-one* classification with *max-wins voting* (*OVO-MWV* SVM).

Due to its formulation, S^{4}VM does not output confidence values for its predictions, thus the strategy *OVA-WTA* cannot be employed. In order to circumvent such a difficulty, one can use the *OVO-MWV* strategy, in which $\frac{c\times(c-1)}{2}$ S^{4}VMs are executed for a c-class problem. According to Rifkin and Klautau [2004], assuming that the base classifiers are properly tuned, there is not much difference among these schemes. However, classifiers that exploit relationships between classes could offer superior performance [Rifkin and Klautau 2004], which is precisely the assumption we try to demonstrate herein.

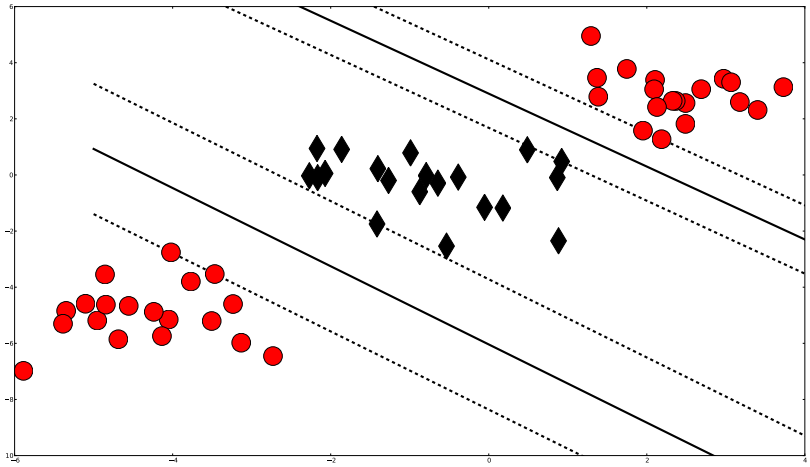
Another problem that arises regards the small number of labeled objects available in transductive learning settings, where selecting a kernel and optimizing its parameters becomes impractical due to the lack of enough data. In the case of multiple classes, this problem gets even worse because of the need for multiple classifiers. Bearing in mind that classes may not be linearly-separable in the input space, a method capable of combining multiple linear SVMs can be of interest in such settings. To illustrate this problem, let us consider Figure 1. In its first part (Figure 1a) a data set with 2 classes (red circles and black diamonds) and 9 labeled objects is presented. One can see that the class represented by red circles is composed by two clusters. Besides, note that the classes are not linearly separable. Thus, even tough this problem can be considered simple, only one linear SVM will not obtain good results. However, if we decompose the problem in such a way that a linear SVM is constructed to differentiate between pairs of clusters of different classes, the problem can be solved by using two linear SVMs. The result of this approach is presented in Figure 1b.

Having similar scenarios in mind, we propose a hierarchical S^{4}VM approach that is capable of *naturally* dealing with multi-class problems. By taking advantage of both class information and distances between data objects, our approach builds a bottom-up binary tree structure that divides the complex input space into sub-spaces, with the underlying (and intuitive) assumption that the respective sub-problems are easier to solve than the whole classification problem. More precisely, the assumption we make is that classes may be distributed across distinct clusters, and thus objects from the same class can lie on different regions of the input space. The proposed algorithm, named Hierarchical Bottom-Up S^{4}VM Tree (*HiBUST*), performs semi-supervised constrained clustering to divide the input space into sub-regions, and then recursively merges the closest sub-regions from distinct classes. Next, it makes use of S^{4}VM to separate two regions of different classes. Experimental results obtained from a variety of different data sets show that *HiBUST* is capable of naturally dealing with transductive multi-class problems, increasing the accuracy of a single S^{4}VM over the whole data set.

The remainder of this article is organized as follows. In Section 2, we contextualize our contribution by briefly reviewing the literature on semi-supervised support vector machines. Section 3 addresses the proposed *HiBUST* algorithm, whereas Section 4 presents an experimental analysis to validate its applicability. We conclude the article in Section 5.



(a) Training data — blue squares represent unlabeled objects, red circles and black diamonds represent labeled objects of (two) different classes.



(b) Classification obtained using problem decomposition.

Fig. 1. Pedagogical example for problem decomposition.

2. RELATED WORK

Semi-supervised support vector machines (S^3VMs) — also known as *Transductive SVMs* (TSVMs) — are extensions of supervised SVMs for performing semi-supervised learning by simultaneously learning the optimal hyperplane and the labels for unlabeled objects [Joachims 1999; Fung and Mangasarian 2001; Chapelle and Zien 2005; Chapelle et al. 2006]. They find the large-margin separator by favoring the decision boundary that lies in low-density data regions. More formally, the goal is to find a function $f : \mathbf{x} \mapsto \{\pm 1\}$ and $\hat{\mathbf{y}} \in \{\pm 1\}^u$, where $u = |\mathcal{X}^{(U)}|$, *i.e.*, the number of unlabeled objects. In order to find f, one formulates a (minimization) optimization problem as follows:

$$\min_{f,\hat{\mathbf{y}} \in \mathcal{B}} \frac{||f||_{\mathcal{H}}}{2} + C_1 \sum_{i=1}^{l} \ell(y_i, f(\mathbf{x}_i)) + C_2 \sum_{j=1}^{u} \ell(\hat{y}_j, f(\hat{\mathbf{x}}_j)) \tag{1}$$

where $\mathcal{B} = \{\hat{\mathbf{y}} \in \{\pm 1\}^u | -\beta \leq \frac{\sum_{j=1}^{u} \hat{y}_j}{u} - \frac{\sum_{i=1}^{l} \hat{y}_i}{l} \leq \beta\}$ is induced through the balancing constraint that avoids the trivial solution of classifying all unlabeled data to the same class; $\mathcal{H}$ is the *Reproducing Kernel Hilbert Space* induced by a given kernel K; $\ell(y, f(\mathbf{x}))$ is the Hinge loss; and C_1 and C_2 are regularization parameters for trading-off complexity and empirical error on labeled and unlabeled data, respectively.

Note, however, that the minimization problem in (1) is *non-convex*, and thus subject to local minima. Avoiding inappropriate local minima when approaching the optimal solution for the problem in (1) is the main goal towards safe semi-supervised support vector machines, *i.e.*, SVMs that never degenerate the performance achieved by a standard supervised approach that only makes use of the labeled data.

An approach towards safe SVMs was implemented by Li and Zhou [2011] in the so-called *safe semi-supervised support vector machines* (S^{4}VMs), which works as follows. Let $\mathcal{M} = \{\hat{\mathbf{y}}\}_{t=1}^{T}$ be the set of predictions provided by low-density separators, $\mathbf{y}^*$ be the ground truth label-assignment, and $\mathbf{y}^{svm}$ be the label-assignment provided by an inductive SVM trained with the labeled data. For any given label-assignment $\mathbf{y} = \{\pm 1\}^u$, consider that $earn(\mathbf{y}, \mathbf{y}^*, \mathbf{y}^{svm})$ denotes the accuracy improvement over the inductive SVM, and that $loss(\mathbf{y}, \mathbf{y}^*, \mathbf{y}^{svm})$ denotes the respective accuracy loss. If the ground truth $\mathbf{y}^*$ was indeed known, the solution would be trivially found by looking for the $\mathbf{y}$ that maximizes $J(\mathbf{y}, \mathbf{y}^*, \mathbf{y}^{svm}) = earn(\mathbf{y}, \mathbf{y}^*, \mathbf{y}^{svm}) - \lambda loss(\mathbf{y}, \mathbf{y}^*, \mathbf{y}^{svm})$. Since the ground truth is unknown, S^{4}VM assumes that $y^* \in \mathcal{M}$, and hence the problem can be seen as the one of optimizing the worst-case improvement over the inductive SVM, that is:

$$\bar{\mathbf{y}} = \arg\max_{\mathbf{y} \in \{\pm 1\}^u} \min_{\hat{\mathbf{y}} \in \mathcal{M}} J(\mathbf{y}, \hat{\mathbf{y}}, \mathbf{y}^{svm}). \tag{2}$$

Note that S^{4}VM takes into account all large-margin low-density separators for discovering the optimal labeling $\hat{\mathbf{y}}$. To find diverse low-density separators $\{f_t\}_{t=1}^{T}$ and their respective label-assignments $\{\hat{\mathbf{y}}_t\}_{t=1}^{T}$, consider $h(f, \hat{\mathbf{y}})$ as being the function to be minimized in (1). Thus, one has the following objective function:

$$\min_{\{f_t, \hat{\mathbf{y}}_t \in \mathcal{B}\}_{t=1}^{T}} \sum_{t=1}^{T} h(f_t, \hat{\mathbf{y}}_t) + M\Omega(\{\hat{\mathbf{y}}_t\}_{t=1}^{T}) \tag{3}$$

where T is the number of separators, Ω is a penalty function regarding their diversity, and M is a large constant (*e.g.*, 10^5) to enforce large diversity. The objective function in (3) favors separators with large-margin as well as with large diversity. Since the optimization problem in (3) is non-convex, Li and Zhou [2011] propose a simple heuristic sampling search, based on k-means clustering, to find representative separators. For more details on S^{4}VMs, please refer to the original reference [Li and Zhou 2011].

As already mentioned, even though S^{4}VMs present several advantages over other transductive learners, they do not naturally deal with multi-class problems (which is also the case of inductive SVMs). Due to various complexities, a direct mathematical solution for multi-class problems using a single SVM formulation is usually avoided [Duan and Keerthi 2005]. Multi-class encoding schemes addressed in Section 1, such as *OVA-WTA*, *OVO-MWV*, and error-correcting output codes [Dietterich and Bakiri 1994], are usually adopted for extending binary classifiers such as SVMs, though each scheme has advantages and disadvantages [Lorena et al. 2008].

It is our impression that the research community does not concentrate enough efforts on providing robust solutions for employing SVMs in multi-class problems (either for supervised or semi-supervised learning). This is probably due to the fact that SVMs can achieve higher accuracy rates with the

current multi-class encoding schemes compared to natural multi-class classifiers, such as Decision Trees and Neural Networks. However, extra gains can be achieved by considering better algorithms for learning SVMs [Rifkin and Klautau 2004].

We focus on *divide-and-conquer* approaches, which consist of strategies for problem decomposition and aggregation. The use of clustering for decomposing classification problems was studied by Vilalta and Rish [2003] in the context of Naïve Bayes classification. The parameters of the clusters for each class were estimated separately to avoid poor estimates caused by objects in different regions of the input space. Barros et al. [2012] propose a mixture of experts algorithm that generates several decision trees from clusters generated by the EM algorithm [Dempster et al. 1977]. In the work of Rida et al. [1999], Cheng et al. [2010], Pang et al. [2011], a SVM[1] is induced on each cluster found in the input space. If the region covered by one local expert contains objects from more than two classes, it is necessary to use a multi-class scheme, such as OVA-WTA. These approaches require critical parameters to be set, such as the number of clusters per class, which are usually unknown *a priori*.

In the work of Vural and Dy [2004], Fei and Liu [2006], Tibshirani and Hastie [2007], Lorena and de Carvalho [2010], and Liu et al. [2011], the multi-class problem is decomposed into a binary tree of SVM classifiers. These algorithms assume that each class is comprised of a single cluster. They also require the selection of a suitable kernel, as well as its parameters, a task which is often determined by a cross-validation procedure. However, for transductive learning settings in which only a small number of labeled objects is available, cross-validation procedures become impractical. Hence, there is a need for algorithms able to combine different linear SVMs, which are known for being state-of-the-art classifiers for binary problems, in order to create complex decision boundaries for multi-class problems when classes are multimodal. Barros et al. [2011] propose a decomposition scheme with SVM for inductive learning that does not consider the one-cluster-per-class assumption. Nevertheless, the authors employ a cross-validation procedure for discovering the number of clusters per class, which is not possible when only a few labeled objects are available, as just discussed.

Our approach differs from the previously presented algorithms as it is the first attempt that simultaneously circumvents three of their fundamental issues by: (i) employing a constrained clustering algorithm capable of estimating the number of clusters for each class; (ii) using an aggregation scheme based on a binary tree that guarantees that we always consider binary problems, regardless of how many classes the problem is comprised of; and (iii) not requiring additional user-defined parameters other than the ones already needed for S^{4}VM.

3. HIERARCHICAL BOTTOM-UP S^{4}VM TREE (*HIBUST*)

As briefly anticipated in Section 2, we propose a new method for addressing S^{4}VM issues regarding multi-class problems. Our approach is named *Hierarchical Bottom-Up S^{4}VM Tree* (*HiBUST*). Essentially, our underlying premise is that some transduction problems are quite difficult to solve when executing a single S^{4}VM and, in such problems, clustering the data set into smaller sub-problems and addressing these smaller problems can make it easier to solve the original problem. Formally speaking, our algorithm approximates the target function in specific regions of the input space, instead of making use of the full input space. These specific regions are not randomly chosen, but defined through a constrained k-means-based clustering algorithm.

More specifically, *HiBUST*'s main steps can be summarized as follows:

(1) decompose classes into clusters (sub-problems) by using a semi-supervised constrained clustering algorithm;

[1]In the work of Rida et al. [1999], any classifier can be employed in the framework.

(2) label some objects *iff* the data partition generated by the constrained clustering algorithm provides confidence-levels suitable for such, and then make use of them accordingly (as discussed later);
(3) build a binary tree in a bottom-up fashion by merging sub-problems according to the distances between centroids;
(4) process unlabeled objects, in a top-down fashion, by traversing them through the tree according to the output of the S^{4}VM for each node of the tree, and then label these objects according to the classes of the leaves they have reached.

Step (2) is optional — it may be used to increase the number of labeled objects so that more evidence is available to S^{4}VM. Each of these steps is described in detail in the next subsections.

3.1 Step 1 — Problem Decomposition

To perform clustering considering the information about the classes of each object, we adopt what we call a Multiple Clusters per Class k-means (MCCK) algorithm [Sestaro et al. 2012]. MCCK receives partial information about the desired partition in the form of must-link (ML) and cannot-link (CL) constraints [Wagstaff et al. 2001]. A ML constraint indicates that two objects should lie in the same cluster, whereas a CL constraint indicates that two objects should lie in different clusters. For a review on constrained clustering, we refer the reader to Basu et al. [2008] and references therein.

Since we assume the availability of a small set of labeled objects, we can derive pairwise constraints from the class labels. However, there is a caveat on the use of the information provided by class labels, particularly due to the (sometimes subtle) difference between class and cluster labels. Since classes can be comprised of multiple clusters, the use of the information provided by class labels for enhancing clustering results may be misleading. More specifically, if two objects belong to the same class but to different clusters, a ML constraint between them will guide the clustering algorithm to merge these two clusters, which can be detrimental to the clustering process.

Note that CL constraints are valid as long as the *cluster assumption*, which states that objects in the same cluster are likely to come from the same class [Chapelle et al. 2006], holds. In other words, the use of class labels as if they were cluster labels, implies in the *one-cluster-per-class* assumption. Although this assumption is often employed in practice, it is rarely explicitly stated in constrained clustering studies, possibly causing misunderstandings. To avoid this confusion, we denote constraints deduced from class labels by $\overline{ML}$ and $\overline{CL}$, *i.e.*, $\overline{ML}$ indicates that two objects belong to the same class, whereas $\overline{CL}$ indicates that two objects belong to different classes. A set of constraints is represented by a symmetric matrix $\mathbf{R}_{N\times N}$ (N is the total number of objects), where:

$$r_{ij} = \begin{cases} 1, & \text{if a } \overline{ML} \text{ constraint between } \mathbf{x}_i \text{ and } \mathbf{x}_j \text{ exists,} \\ -1, & \text{if a } \overline{CL} \text{ constraint between } \mathbf{x}_i \text{ and } \mathbf{x}_j \text{ exists,} \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

The intuition behind MCCK is that if a constraint is violated as an object is assigned to a cluster, the current number of clusters (for that particular class) may be insufficient, and a new cluster should be created from that object. Initially, there are c clusters, *i.e.*, one for each class. Due to the process of creating clusters, a mapping is necessary to control which clusters belong to each class. Initially, this mapping has a one-to-one correspondence between classes and clusters. The mapping is updated when new clusters are created — later this update procedure is described in more detail. MCCK sequentially assigns objects to the nearest clusters. However, objects involved in constraints are only assigned to a cluster if the constraints are not violated. Since we do not assume that there is only a single cluster per class, the constraints are not verified by simply taking into account the objects of each cluster. Instead, we do so by jointly considering classes and their respective clusters.

For instance, suppose that we have two objects, $\mathbf{x}_i$ and $\mathbf{x}_j$. Suppose $\mathbf{x}_i$ has been assigned to cluster C_n, whose prototype is μ_n. Further assume that $r_{ij} = 1$ and that $\boldsymbol{\mu}_m$ is the nearest prototype to $\mathbf{x}_j$. Then, $\mathbf{x}_j$ will be assigned to C_m only if C_m and C_n belong to the same class — this is verified through the mapping previously mentioned. If the clusters belong to different classes, a new cluster is created, and its prototype is $\mathbf{x}_j$. For the mapping update, first the class that $\mathbf{x}_j$ belongs to is identified. Next, the new cluster is added to the mapping of that class. Clearly, this process is order-dependent. To alleviate this problem, at each iteration the objects are processed in a random order. After all objects have been assigned to clusters, their prototypes (in this case centroids) are updated.

As the creation of clusters is also order-dependent, *unnecessary* clusters may arise. A pair of unnecessary clusters occurs when their prototypes are nearest neighbors, and both belong to the same class. Hence, after the prototypes have been updated, we verify if a pair of clusters meets this criterion. If this is the case, those two clusters are merged. The prototype of this new cluster will be the average vector of the two merged prototypes. Again, the mapping between clusters and classes is updated accordingly.

The main steps of MCCK are described in Figure 2, where $d(\mathbf{x}_i, \boldsymbol{\mu}_n) = \|\mathbf{x}_i - \boldsymbol{\mu}_n\|^2$. Step 11 verifies if the assignment will cause a constraint to be violated. Note that even though $\overline{ML}$ constraints are not directly verified, they are implicitly examined by checking the classes. As each cluster belongs to a single class, the $\overline{ML}$ constraints will not be violated, as long as two objects in a $\overline{ML}$ constraint belong to clusters of the same class. The same termination criteria commonly used in k-means can be adopted, *e.g.*, a maximum number of iterations or a minimum distance between consecutive centroids. At the end of its execution, MCCK provides a mapping $\{\mathcal{H}_i\}_{i=1}^{c}$ with the indices of the clusters that belongs to each class, as well as the partition itself ($\{C_i\}_{i=1}^{k}$).

3.2 Step 2 — Confidence-Based Labeling

When performing k-means-based clustering, the assumption is that the data is generated by a Gaussian Mixture Model (GMM), where each component (Gaussian distribution) is a cluster. In this case, the component mean is the cluster centroid and the covariance matrix is the identity matrix. In addition, it is well-known that k-means performs hard-assignments, *i.e.*, the probability that one object is generated by a component is either 0 or 1. For the purpose of obtaining a degree of belief with respect to which cluster generated each object, one can convert the crisp partition obtained by MCCK into a soft partition by computing the posterior probabilities given by the GMM that represents the respective data partition. Specifically, the probability that ($\mathbf{x}_i$) is generated by the component (C_j) is:

$$P(j|\mathbf{x}_i) = \frac{\pi_j \mathcal{N}(\mathbf{x}_i|\boldsymbol{\mu}_j, \mathbf{I})}{\sum_{n=1}^{k} \pi_n \mathcal{N}(\mathbf{x}_i|\boldsymbol{\mu}_n, \mathbf{I})}, \tag{5}$$

where $\mathcal{N}(\cdot)$ is the Gaussian distribution, $\mathbf{I}$ is the identity matrix, π_j is the mixing coefficient of the jth component, *i.e.*, the proportion of objects assigned to it by the hard-assignment, and k is the number of components.

Assuming a user-defined threshold, θ, one can label objects that have a high probability of belonging to a cluster, following standard procedures widely used in semi-supervised settings. More precisely, assuming that every cluster centroid is a representative vector of a given class, $y_i = c_j$ *iff* $P(j|\hat{\mathbf{x}}_i) > \theta$, $\forall \hat{\mathbf{x}}_i \in \mathcal{X}^{(U)}$, where c_j is the class represented by the cluster centroid ($\boldsymbol{\mu}_j$). Both $\mathcal{X}^{(L)}$ and $\mathcal{X}^{(U)}$ are updated accordingly. Informally speaking, the confidence level (θ) can be seen as a certainty measure that the model has about an object being sampled from a particular cluster and, as a consequence, from a particular class.

function MCCK($\mathcal{X} = \{\mathcal{X}^{(U)} \cup \mathcal{X}^{(L)}\}, \mathbf{R}$)
  $k \leftarrow c$ // One cluster for each class
  for each $j \in \{1, \ldots, k\}$ **do**
    Let $\mathbf{x}_r$ be an object randomly selected from the jth class
    $\boldsymbol{\mu}_j \leftarrow \mathbf{x}_r$
    $\mathcal{H}_j \leftarrow \{j\}$ // Mapping between clusters and classes
  end for
  for each $\mathbf{x}_i \in \mathcal{X}$ **do**
    $n \leftarrow \arg\min_{n \in \{1,\ldots,k\}} d(\mathbf{x}_i, \boldsymbol{\mu}_n)$
    Consider that $n \in \mathcal{H}_{c_n}$
    if $\{\exists a, j, s | (r_{ia} = -1) \wedge (\mathbf{x}_a \in C_j) \wedge (C_j \in \mathcal{H}_s) \wedge (s = c_n)\}$ **then**
      $k \leftarrow k + 1$
      Consider that $\mathbf{x}_i$ belongs to the vth class
      $\mathcal{H}_v \leftarrow \mathcal{H}_v \cup \{k\}$
      $\boldsymbol{\mu}_k \leftarrow \mathbf{x}_i$
      $C_k \leftarrow \{\mathbf{x}_i\}$
    else
      $C_n \leftarrow C_n \cup \{\mathbf{x}_i\}$
    end if
  end for
  Update prototypes
  for each pair (C_p, C_q) of clusters **do**
    $n \leftarrow \arg\min_{n \in \{1,\ldots,k\} \setminus \{p\}} d(\boldsymbol{\mu}_p, \boldsymbol{\mu}_n)$
    $m \leftarrow \arg\min_{m \in \{1,\ldots,k\} \setminus \{q\}} d(\boldsymbol{\mu}_q, \boldsymbol{\mu}_m)$
    Consider that $p \in \mathcal{H}_{c_p}$
    Consider that $q \in \mathcal{H}_{c_q}$
    if $c_p = c_q \wedge n = q \wedge m = p$ **then**
      Merge clusters C_p and C_q
      Update the mapping $\mathcal{H}_{c_p}$
    end if
  end for
  If the convergence criterion was not met, then go to Step 8
  Return $\{\mathcal{H}_i\}_{i=1}^{c}$ (mapping) and $\{C_i\}_{i=1}^{k}$ (data partition)
end function

Fig. 2. Multiple Clusters per Class k-means algorithm (MCCK)

3.3 Step 3 — Tree Construction

The clusters obtained by MCCK correspond to the leaves of the binary tree to be constructed. Note that several different groups may have labeled objects that share the same class, since we generate at least one leaf per class.

After creating the leaves, a recursive procedure is initiated. A new group (internal node) is generated by merging the two most similar nodes, *i.e.*, those whose centroids are the closest according to the Euclidean distance. Once a node is created, its corresponding objects (that came up from its children) have their class label replaced by a new unique *pseudo-class*. Since we focus on binary trees, each node is merged only once. If the two closest leaf nodes are from the same class, a new leaf node is created with the union of the objects belonging to each node, and the old nodes are deleted, *i.e.*, they do not appear in the resulting tree. The recursion stops when there is only one group (root node) that has not been merged. The tree construction procedure is described in Figure 3.

Each node in the tree (leaf or internal) is comprised of three components:

(1) *labeledInst*: the set of labeled objects in the node;
(2) *class*: the actual class (*pseudo-class*) that this leaf (internal) node corresponds to;

```
1: function TreeConstruction(X^(L), {H_i}_{i=1}^c, {C_i}_{i=1}^k)
2:   Leaves ← ∅
3:   for each H_i ∈ {H_i}_{i=1}^c do
4:     for each j ∈ H_i do
5:       create new leaf node n
6:       n.labeledInst ← {x_r ∈ C_j | x_r ∈ X^(L)}
7:       n.centroid ← mean vector of n.labeledInst
8:       n.class ← i
9:       Leaves ← Leaves ∪ {n}
10:    end for
11:  end for
12:  root ← Merging(Leaves)
13:  Return root
14: end function
```

```
1: function Merging(L)
2:   if |L| = 1 then
3:     Return unique node in L (root)
4:   else
5:     Let n_1 and n_2 be the two nodes from L with the closest centroids according to the Euclidean distance
6:     if n_1.class = n_2.class then      // Merge leaves
7:       create leaf node t
8:       t.class ← n_1.class
9:     else
10:      create new internal node t
11:      t.leftChild ← n_1
12:      t.rightChild ← n_2
13:      t.class ← new unique pseudo-class
14:    end if
15:    t.labeledInst ← n_1.labeledInst ∪ n_2.labeledInst
16:    t.centroid ← mean vector of t.labeledInst
17:    L ← L ∪ {t}
18:    L ← L \ {n_1}
19:    L ← L \ {n_2}
20:    Return Merging(L)
21:  end if
22: end function
```

Fig. 3. Tree construction procedure.

(3) *centroid*: the mean vector of the labeled objects.

3.4 Step 4 — Tree-based Labeling

Once the tree has been built, we can distribute the unlabeled objects through the tree, starting from the root node. For such, a S^4VM is created for each internal node, which will be responsible for defining which path (left or right) the objects will be distributed to. Recall that the children of a node come from two different classes (for leaf nodes) or pseudo-classes (for internal nodes). Thence, S^4VM needs only to deal with binary problems, regardless of the number of classes in the original data set — which means its application is straightforward and does not rely on any voting scheme between pairwise class subsets. At the end of the process, each object will be at a leaf node, and its predicted label is the label of the class corresponding to that leaf. This recursive process is summarized in Figure 4.

```
1: function Classify(root, X^(U))
2:   if isLeaf(root) then
3:     Pred ← ∅
4:     for each x̂_i ∈ X^(U) do
5:       Pred ← Pred ∪ {(x̂_i, root.class)}
6:     end for
7:   else
8:     negObj ← root.leftChild.labeledInst
9:     posObj ← root.rightChild.labeledInst
10:    r ← S4VM(negObj, posObj, X^(U))
11:    Let X^(n) be all unlabeled objects classified as negative
12:    Let X^(p) be all unlabeled objects classified as positive
13:    Pred ← Classify(root.leftChild, X^(n))
14:    Pred ← Pred ∪ Classify(root.rightChild, X^(p))
15:  end if
16:  return Pred
17: end function
```

Fig. 4. *HiBUST*'s classification procedure.

3.5 An Illustrative Example

Now that each step of *HiBUST* has been detailed, it is useful to present an illustrative example of its execution mode. For such, consider again the pedagogical example from Figure 1a. The result of the decomposition applied in Step 1 is presented in Figure 5a. We can observe that the class previously represented by red circles (a few labeled objects) has been split into two clusters (red stars and green pentagons). For the sake of simplicity, we assume that confidence levels (Step 2) are here not employed to label objects, *i.e.*, the optional parameter θ has not been informed by the user.

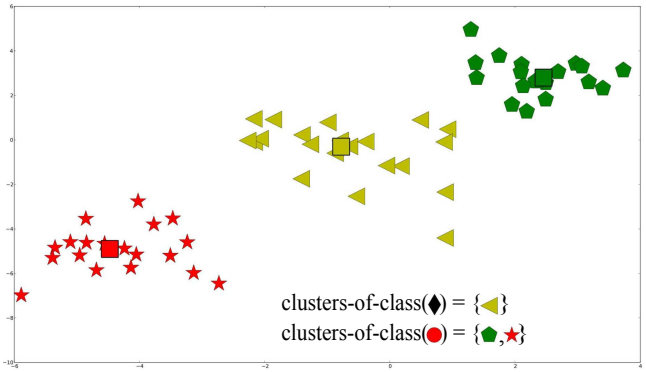
Considering the three clusters generated by MCCK (sub-problems), the binary tree can be built in a bottom-up fashion. First, the closest pair of centroids is identified — in this case, the centroids of the clusters represented by yellow left-triangles and green pentagons. An internal node is thus created by merging these sub-problems. Figure 5b illustrates the tree at this stage. Next, the root node is created by merging the recently created internal node with the sub-problem related to the cluster represented by the red stars. The final tree is illustrated in Figure 5c.

In order to label the unlabeled data (blue squares in Figure 1a), the tree is used as a mechanism to filter the data, starting from the root node down to the leaves. A S^4VM is used to label the objects as belonging to either the red-star class or to the pseudo-class that represents the remaining data. Objects classified as belonging to the red-star class follow directly to the corresponding red-circle's leaf node.

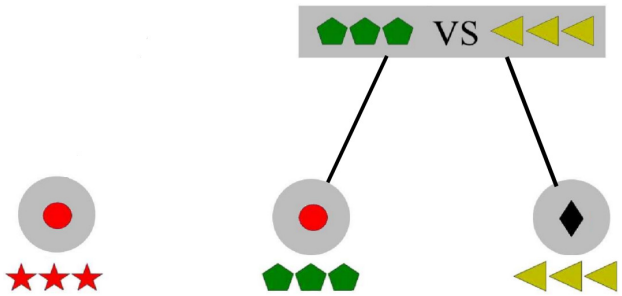
The remaining objects are processed in another internal node, which uses a S^4VM to label objects as either belonging to the green-pentagon class or to the yellow-left-triangle class. Again, according to the result of this particular S^4VM, objects follow to their corresponding leaf (red circles or black diamonds).

4. EXPERIMENTAL ANALYSIS

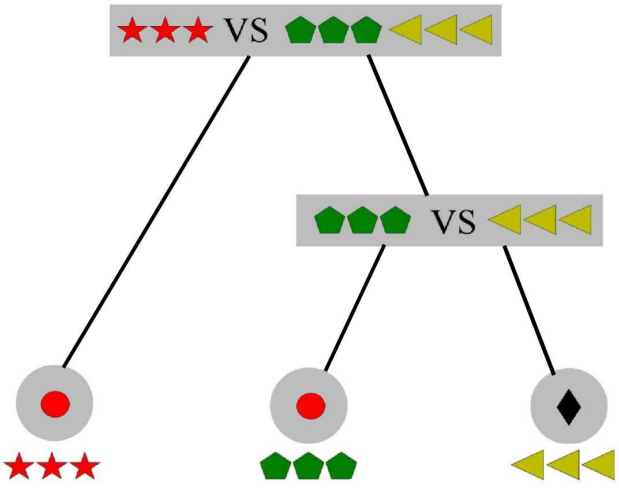
In this section, we present the experimental methodology and the results obtained to validate the effectiveness of *HiBUST* for solving semi-supervised transductive learning problems.



(a) Result of classes to clusters decomposition (Step 1). Different markers represent different clusters and the centroid of each cluster is denoted by a square. See Fig. 1a for a description of the classes.



(b) Tree obtained after the first merge performed by the aggregattion scheme (Step 3). Leaves are represented by circles with the symbol of the corresponding class inside, and the labeled objects are presented in each leaf according to the cluster they were assigned to. Each internal node is represented by a rectangle with the definition of the binary sub-problem that it addresses.



(c) Final tree obtained by the aggregattion scheme (Step 3).

Fig. 5. *HiBUST*'s main steps for the pedagogical example.

Table I. Summary of the employed data sets.

data set	#objects	#attributes	#classes	class distribution	source
9Gauss	900	2	9	100 (all)	[Campello et al. 2009]
Balance-Scale	625	4	3	288-49-288	UCI repository
Ionosphere	351	34	2	126-225	UCI repository
Iris	150	4	3	50 (all)	UCI repository
Mfeat	2000	6	10	200 (all)	UCI repository
Pendigits	3165	16	3	1055 (all)	UCI repository
Segment	2310	19	7	330 (all)	UCI repository
USPS	1500	241	2	1200-300	[Chapelle et al. 2006]
Vehicle	846	18	4	212-217-218-199	UCI repository
Wine	178	13	3	59-71-48	UCI repository

4.1 Methodology

We compare *HiBUST* to S^{4}VM [Li and Zhou 2011], which is a state-of-the-art transductive SVM method. Experiments were performed on 10 data sets commonly used as benchmarks in the literature. Most of them are available at the UCI repository[2]. We have used the 9Gauss data set [Campello et al. 2009], which is comprised of 9 balanced clusters (each cluster is considered as a class) arranged according to Gaussian distributions that have some degree of overlapping. Besides, we have employed the USPS data set, which is a well-known semi-supervised learning benchmark data set [Chapelle et al. 2006]. Following Bilenko et al. [2004], for the Pendigits data set, only the classes 3, 8, and 9 were considered, since they represent a difficult classification problem [Bilenko et al. 2004]. Table I describes the data sets employed in the experiments.

For all data sets, we randomly selected 10, 20, and 30 objects per class to be used as labeled data, and the remaining objects were used as unlabeled data. The experiments were repeated 10 times and the average accuracies and standard deviations were recorded. Since MCCK is initialization-dependent, for each experiment we run MCCK 10 times and selected the partition with the best simplified silhouette value [Hruschka et al. 2006]. The simplified silhouette is a relative validity criterion that can be used to choose, among a set of different data partitions, the best available clustering. For all the employed S^{4}VM, we used the MATLAB implementation by Li and Zhou [2011][3], using default values for its parameters and a linear kernel. Li and Zhou [2011] show evidence that S^{4}VM is robust to the selection of its parameters. For data sets with more than two classes, we execute S^{4}VM in a *one-versus-one* classification scheme with a *max-wins-voting* strategy (*OVO-MWV*).

We report two kinds of experimental analyses. First, in Section 4.2, we compare the results of *HiBUST* to the S^{4}VM algorithm [Li and Zhou 2011]. For these comparisons, we assume that the user has not provided a value for the θ parameter, *i.e.*, we do not use confidence-based labeling (optional Step 2). Later, in Section 4.3, we analyze the impact of using the confidence-based labeling in the classification accuracy.

4.2 Results

Table II presents the mean accuracy obtained by S^{4}VM and *HiBUST* for each data set. First, consider the results for the two binary-class data sets, namely Ionosphere and USPS. For both of them, *HiBUST* obtained equal or better results than S^{4}VM, regardless of the number of labeled objects. These results suggest that these data sets fit into the category of data whose number of clusters per class is greater than one. For this kind of data set, combining different linear boundaries indeed results in an increased classification performance, as we have previously assumed when presenting

[2] `http://archive.ics.uci.edu/ml`
[3] `http://lamda.nju.edu.cn/Default.aspx?Page=code_S4VM`

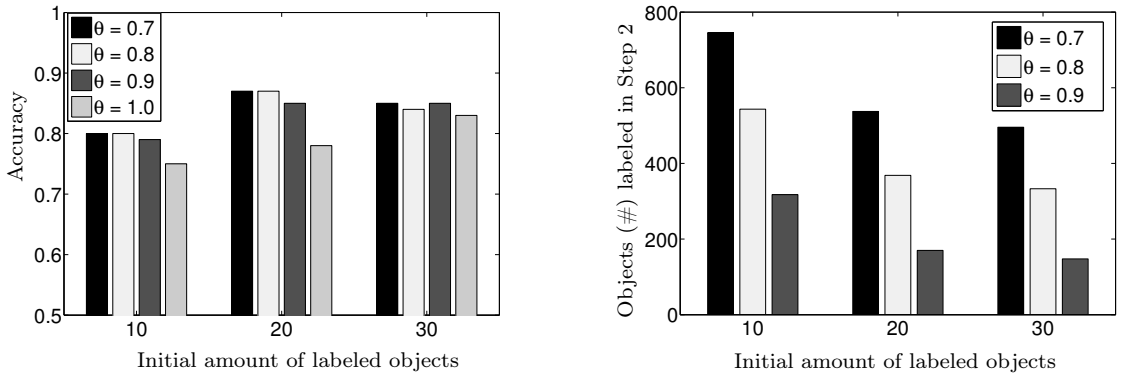
Table II. Average accuracy ± standard deviation for different amounts of labeled objects per class (#LO).

#LO	Data sets	S^4VM	*HiBUST*
10	Ionosphere	0.74 ± 0.04	**0.78 ± 0.05**
	USPS	**0.75 ± 0.05**	**0.75 ± 0.05**
	9Gauss	0.67 ± 0.04	**0.85 ± 0.02**
	Iris	0.81 ± 0.04	**0.92 ± 0.02**
	Mfeat	0.60 ± 0.04	**0.64 ± 0.01**
	Wine	0.87 ± 0.02	**0.94 ± 0.01**
	Pendigits	0.96 ± 0.02	**0.96 ± 0.01**
	Segment	**0.84 ± 0.03**	0.81 ± 0.02
	Balance-Scale	**0.82 ± 0.04**	0.57 ± 0.04
	Vehicle	**0.59 ± 0.03**	0.50 ± 0.03
20	Ionosphere	0.74 ± 0.04	**0.81 ± 0.03**
	USPS	0.77 ± 0.04	**0.78 ± 0.04**
	9Gauss	0.70 ± 0.05	**0.88 ± 0.01**
	Iris	0.86 ± 0.02	**0.94 ± 0.01**
	Mfeat	0.59 ± 0.02	**0.65 ± 0.01**
	Wine	0.90 ± 0.04	**0.94 ± 0.01**
	Pendigits	**0.98 ± 0.00**	0.97 ± 0.01
	Segment	**0.87 ± 0.02**	0.85 ± 0.01
	Balance-Scale	**0.90 ± 0.01**	0.60 ± 0.02
	Vehicle	**0.66 ± 0.02**	0.55 ± 0.01
30	Ionosphere	0.74 ± 0.04	**0.84 ± 0.01**
	USPS	0.78 ± 0.02	**0.83 ± 0.04**
	9Gauss	0.69 ± 0.04	**0.88 ± 0.02**
	Iris	0.94 ± 0.02	**0.94 ± 0.01**
	Mfeat	0.61 ± 0.03	**0.66 ± 0.01**
	Wine	**0.93 ± 0.01**	**0.93 ± 0.01**
	Pendigits	**0.98 ± 0.00**	0.97 ± 0.01
	Segment	0.85 ± 0.02	**0.86 ± 0.01**
	Balance-Scale	**0.91 ± 0.00**	0.62 ± 0.03
	Vehicle	**0.69 ± 0.02**	0.57 ± 0.02

our method. Hence, we believe *HiBUST* can be a better option than S^4VM even for binary-class problems, especially considering the lack of sufficient labeled objects for tuning a kernel.

Now, consider the 9Gauss and Iris data sets, which have 9 and 3 classes, respectively. Roughly speaking, one could take for granted that these data sets are comprised by one cluster per class — although this is somehow arguable for the Iris data, as it has been extensively discussed in the literature. Both of them have overlapping classes, which is detrimental for SVM-based methods. Bearing this in mind, the use of a decomposition scheme that breaks the problem into sub-problems is, once again, beneficial. More specifically, MCCK creates clusters in the overlapping areas among classes, which is helpful for classifying objects in such specific regions. This can be seen by the accuracy results of *HiBUST* in these data sets, which are better than the ones obtained by S^4VM for all amounts of labeled objects.

The results for the Mfeat and Wine data sets are also favorable for *HiBUST*. For the Pendigits and Segment data sets, S^4VM presented slightly better results than *HiBUST* on average. In the remaining data sets — Balance-Scale and Vehicle — S^4VM outperformed *HiBUST*. Note that Balance-Scale is a highly imbalanced data set, and this seems to indicate that *HiBUST* is sensitive to imbalanced data. Considering that imbalanced problems are common in several applications, we plan to investigate in detail this problem in future work.



(a) Accuracy.

(b) Number of objects labeled based on confidence.

Fig. 6. Results obtained varying θ — USPS data set.

4.3 Sensitivity of Confidence-Based Labeling

For assessing the impact of Step 2 (confidence-based labeling) in *HiBUST*, we varied θ within $\{0.7, 0.8, 0.9\}$. We believe that these values cover distinct scenarios — from average to high confidence — for increasing the number of labeled objects available to induce the classifier. For most data sets, there were no objects labeled considering these probabilities. Thus, their respective results with these different values of θ are the same as those presented in the previous section.

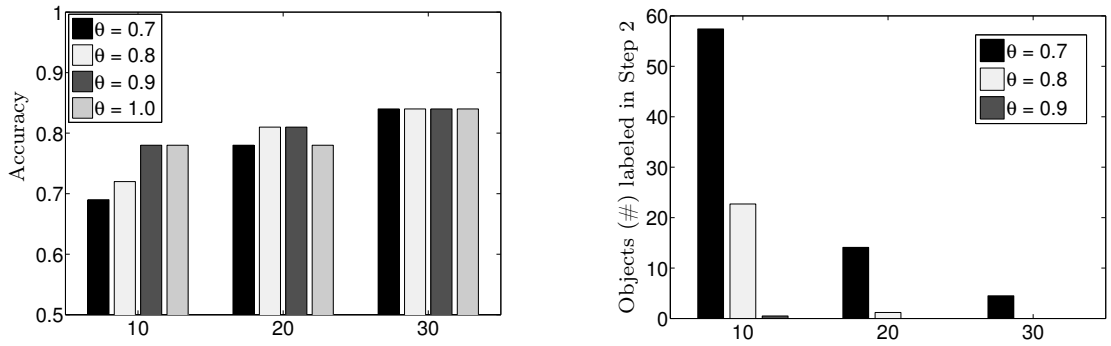
Nevertheless, for two data sets (USPS and Ionosphere) different results were obtained through the confidence-based labeling. For instance, consider Figures 6 and 7, which present the number of objects labeled from Step 2 and the mean accuracy obtained by *HiBUST* for the different values of θ. For the mean accuracy, we replicate the results without using the confidence-based labeling (*i.e.*, $\theta = 1.0$). Observe that, for the USPS data set, the number of objects labeled was quite large (more than 100) even for $\theta = 0.9$. This suggests that some of its clusters are very dense and well-separated. Through this mechanism of confidence-based labeling, more labeled data were available for each S^4VM within an internal nodes of *HiBUST*. Hence, S^4VM was able to select better hyperplanes which, in turn, resulted in better predictive performance.

For the Ionosphere data set, the clusters seem to be less dense, and just a few objects had probabilities greater than 0.9 of being generated by the Gaussian that represents its cluster. When labeling objects with smaller values of θ ($\theta = 0.8$ and $\theta = 0.7$), the accuracy of *HiBUST* decreases comparatively to the results obtained without confidence-based labeling. As expected, these results suggest that it is worth employing a high value of θ (such as $\theta = 0.9$), given that in most cases an equal or better predictive performance was achieved when compared to the absence of the confidence-based labeling scheme.

5. CONCLUSIONS

In this article, we presented a new aggregation scheme for safe semi-supervised support vector machines based on a binary tree built in a bottom-up fashion. This method, named *HiBUST*, allows a binary SVM to naturally deal with multi-class transductive problems. It takes advantage of both class information and distances between data objects for dividing the complex input space into sub-spaces, with the underlying assumption that sub-problems are easier to solve than the whole classification problem. In addition, it assumes that the objects of the classes may be distributed across distinct clusters and thus objects from the same class can lie on different regions of the input space.

We performed an empirical analysis of *HiBUST* by comparing it to S^4VM [Li and Zhou 2011]



(a) Accuracy.

(b) Number of objects labeled based on confidence.

Fig. 7. Results obtained varying θ — Ionosphere data set.

in 10 classification problems, considering different amounts of labeled objects. In the performed analyses, *HiBUST* presented equal or better predictive performance than S^{4}VM in the majority of the cases. Besides, results for two binary-class data sets suggested that *HiBUST* can also be useful for such problems, especially if the amount of labeled objects is insufficient for tuning a kernel. We also analyzed a scheme to label objects based on the confidence provided by the constrained clustering algorithm employed by *HiBUST*. Results suggest that the confidence-based labeling scheme can provide increased performance. Indeed, when high-confidence objects are labeled, each S^{4}VM within an internal node of the tree created by *HiBUST* presented better capability of separating objects from different classes.

As future work, we plan to investigate the use of the Expectation Maximization (EM) algorithm [Dempster et al. 1977] for refining the parameters of each cluster obtained by MCCK and thus obtaining better estimates of the confidence-based labeling probabilities. In addition, an analysis on the use of *HiBUST* for imbalanced multi-class problems is a promising future work.

REFERENCES

Bahadori, M. T., Liu, Y., and Zhang, D. Learning with Minimum Supervision: a general framework for transductive transfer learning. In *International Conference on Data Mining*. Vancouver, BC, Canada, pp. 61–70, 2011.

Barros, R. C., Basgalupp, M. P., de Carvalho, A. C. P. L. F., and Quiles, M. G. Clus-DTI: improving decision-tree classification with a clustering-based decision-tree induction algorithm. *Journal of the Brazilian Computer Society* 18 (4): 351–362, 2012.

Barros, R. C., Cerri, R., Jaskowiak, P. A., and de Carvalho, A. C. P. L. F. A Bottom-Up Oblique Decision Tree Induction Algorithm. In *International Conference on Intelligent Systems Design and Applications*. Cordoba, Spain, pp. 450 –456, 2011.

Basu, S., Davidson, I., and Wagstaff, K. *Constrained Clustering: advances in algorithms, theory, and applications*. Chapman & Hall/CRC, Boca Raton, FL, USA, 2008.

Bilenko, M., Basu, S., and Mooney, R. J. Integrating Constraints and Metric Learning in Semi-Supervised Clustering. In *Proceedings of the International Conference on Machine Learning*. New York, NY, USA, 2004.

Campello, R. J., Hruschka, E. R., and Alves, V. S. On the Efficiency of Evolutionary Fuzzy Clustering. *Journal of Heuristics* 15 (1): 43–75, 2009.

Chapelle, O., Chi, M., and Zien, A. A Continuation Method for Semi-Supervised SVMs. In *Proceedings of the International Conference on Machine learning*. New York, NY, USA, 2006.

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

Chapelle, O., Vapnik, V., and Weston, J. Transductive Inference for Estimating Values of Functions. In *Advances in Neural Information Processing Systems*. Denver, CO, USA, pp. 421–427, 1999.

Chapelle, O. and Zien, A. Semi-Supervised Classification by Low Density Separation. In *Proceedings of the International Workshop on Artificial Intelligence and Statistics*. Savannah Hotel, Barbados, pp. 57–64, 2005.

Cheng, H., Tan, P.-N., and Jin, R. Efficient Algorithm for Localized Support Vector Machine. *IEEE Transactions on Knowledge and Data Engineering* 22 (4): 537–549, 2010.

Dempster, A. P., Laird, N. M., and Rubin, D. B. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society* 39 (1): 1–38, 1977.

Dietterich, T. G. and Bakiri, G. Solving Multiclass Learning Problems via Error-Correcting Output Codes. *Journal of Artificial Intelligence Research* 2 (1): 263–286, 1994.

Duan, K.-B. and Keerthi, S. S. Which is the Best Multiclass SVM Method? An Empirical Study. In *Proceedings of the International Workshop on Multiple Classifier Systems*. Seaside, CA, USA, pp. 278–285, 2005.

Fei, B. and Liu, J. Binary Tree of SVM: a new fast multiclass training and classification algorithm. *IEEE Transactions on Neural Networks* 17 (3): 696–704, 2006.

Fung, G. and Mangasarian, O. L. Semi-Supervised Support Vector Machines for Unlabeled Data Classification. *Optimization Methods and Software* vol. 15, pp. 29–44, 2001.

Gammerman, A., Vovk, V., and Vapnik, V. Learning by Transduction. In *Uncertainty in Artificial Intelligence*. Morgan Kaufmann, San Francisco, CA, USA, pp. 148–155, 1998.

Hruschka, E. R., Campello, R. J., and de Castro, L. N. Evolving Clusters in Gene-Expression Data. *Information Sciences* 176 (13): 1898 – 1927, 2006.

Joachims, T. Transductive Inference for Text Classification using Support Vector Machines. In *Proceedings of the International Conference on Machine Learning*. San Francisco, CA, USA, pp. 200–209, 1999.

Li, Y.-F. and Zhou, Z.-H. Towards Making Unlabeled Data Never Hurt. In *International Conference on Machine Learning*. Bellevue, Washington, USA, pp. 1081–1088, 2011.

Liu, D., Yan, S., Mu, Y., Hua, X.-S., Chang, S.-F., and Zhang, H.-J. Towards Optimal Discriminating Order for Multiclass Classification. In *Proceedings of the IEEE International Conference on Data Mining*. Vancouver, BC, Canada, pp. 388–397, 2011.

Lorena, A., de Carvalho, A., and Gama, J. A Review on the Combination of Binary Classifiers in Multiclass Problems. *Artificial Intelligence Review* vol. 30, pp. 19–37, 2008.

Lorena, A. C. and de Carvalho, A. C. P. L. F. Building Binary-Tree-based Multiclass Classifiers Using Separability Measures. *Neurocomputing* 73 (16-18): 2837–2845, 2010.

Pang, S., Ban, T., Kadobayashi, Y., and Kasabov, N. Personalized Mode Transductive Spanning SVM Classification Tree. *Information Sciences* 181 (11): 2071 – 2085, 2011.

Rida, A., Labbi, A., and Pellegrini, C. Local Experts Combination through Density Decomposition. In *International Workshop on AI and Statistics*. Ft. Lauerdale. Florida, USA, 1999.

Rifkin, R. and Klautau, A. In Defense of One-vs-All Classification. *Journal of Machine Learning Research* vol. 5, pp. 101–141, 2004.

Sestaro, D., Covões, T., and Hruschka, E. A Semi-Supervised Approach to Estimate the Number of Clusters per Class. In *Proceedings of the Brazilian Symposium on Neural Networks*. Curitiba, PR, Brazil, pp. 73–78, 2012.

Tibshirani, R. and Hastie, T. Margin Trees for High-dimensional Classification. *Journal of Machine Learning Research* vol. 8, pp. 637–652, 2007.

Vapnik, V. *Statistical Learning Theory*. Wiley, New York, NY, USA, 1998.

Vilalta, R. and Rish, I. A Decomposition of Classes via Clustering to Explain and Improve Naive Bayes. In *European Conference on Machine Learning*. Cavtat-Dubrovnik, Croatia, pp. 444–455, 2003.

Vural, V. and Dy, J. G. A Hierarchical Method for Multi-Class Support Vector Machines. In *Proceedings of the International Conference on Machine learning*. Banff, Alberta, Canada, pp. 105–112, 2004.

Wagstaff, K., Cardie, C., Rogers, S., and Schroedl, S. Constrained K-Means Clustering with Background Knowledge. San Francisco, CA, USA, pp. 577–584, 2001.

Yu, J., Bian, W., Song, M., Cheng, J., and Tao, D. Graph Based Transductive Learning for Cartoon Correspondence Construction. *Neurocomputing* vol. 79, pp. 105–114, 2012.

Zhang, Y.-M., Zhang, Y., Yeung, D.-Y., Liu, C.-L., and Hou, X. Transductive Learning on Adaptive Graphs. In *Association for the Advancement of Artificial Intelligence Conference on Artificial Intelligence*. Atlanta, GA, USA, pp. 661–666, 2010.

Zhu, X. and Goldberg, A. B. *Introduction to Semi-Supervised Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2009.