Soft Margin Trees

Jorge Díez, Juan José del Coz, Antonio Bahamonde, and Oscar Luaces

Artificial Intelligence Center, University of Oviedo at Gijón, Asturias, Spain
www.aic.uniovi.es

Abstract. From a multi-class learning task, in addition to a classifier, it is possible to infer some useful knowledge about the relationship between the classes involved. In this paper we propose a method to learn a hierarchical clustering of the set of classes. The usefulness of such clusterings has been exploited in bio-medical applications to find out relations between diseases or populations of animals. The method proposed here defines a distance between classes based on the margin maximization principle, and then builds the hierarchy using a linkage procedure. Moreover, to quantify the goodness of the hierarchies we define a measure. Finally, we present a set of experiments comparing the scores achieved by our approach with other methods.

1 Introduction

In many Machine Learning and Data Mining applications, users are not only interested in learning good classifiers but also in gaining some insight into the application domain. This is the case, for instance, of learning techniques for association rule, clustering or feature selection.

Given a dataset of labeled examples, in this paper we present a method to cluster the set of classes. This learning task has received little attention in the literature; typically, clustering deals with examples instead of classes. However, once we have classes attached to examples, their clusters draw valuable information about the similarities and differences between classes.

In *Medicine*, [1,2] report methods for clustering SAGE (Serial Analysis of Gene Expression) data to detect similarities and dissimilarities between different types of cancer on the subcellular level. There are also many interesting applications in *Genetics of Populations*. The aim is to discover relationships between species or breeds according to their genetic descriptions. Thus, [3] studied 50 indigenous cattle breeds from Africa; [4] clustered a total of 1272 termites representing 56 genetically distinct colonies in central North Carolina; [5] investigated the genetic structure and variation of 21 populations of cattle in northern Eurasia and the neighbouring Near Eastern regions; [6], in order to facilitate the assessments of epidemiological risks, showed the genetic structure of human populations using genotypes at 377 autosomal microsatellite loci in 1056 individuals from 52 populations.

Roughly speaking the clustering of classes has been faced in two ways. The straightforward approach represents each class using a single feature vector,

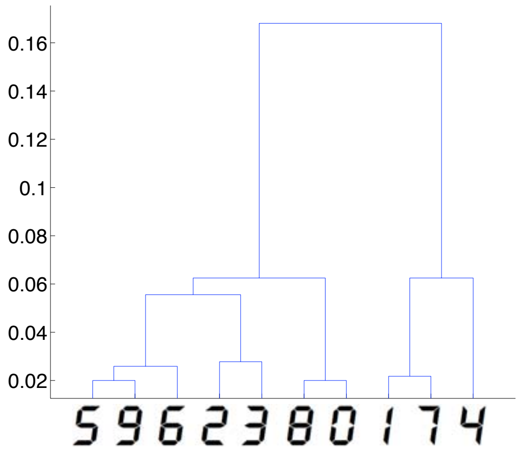

Fig. 1. Hierarchical clustering of classes obtained on led dataset. This domain contains 7 boolean attributes, representing the 7 light-emitting diodes, and 10 classes, the set of decimal digits. To make the task more difficult, the problem has some noise. Each attribute value has the 10% probability of having its value inverted.

usually the centroid of the examples of the class in the input space [7]. Then, the collection of these vectors is clustered using algorithms as k-means, Self-Organized Maps [8] or a hierarchical clustering. The main drawback of this approach is that representing a class by a single vector may imply an important loss of information.

The second approach proceeds in two stages. First, a clustering is obtained using the complete dataset of individual examples. Then, in an *ad hoc* way, the users analyze the groups so obtained. Typically, the individual examples are represented in a 2-D map using visualizations tools like Self-Organized Maps [9,10,1]; from this map an application-specific discussion *infers* a graph that represents the relationship between classes [11]. Using these types of methods, knowledge discovery depends heavily on the capability of users to interpret the clustering of individual examples. Using this approach, it is a difficult task to find the true relationship between all classes of a classification task.

The method proposed in this paper builds a hierarchical clustering [12] of the set of classes. The core idea is to define a metric between classes. Given that the starting data is a classification learning task, the metric of classes is defined from a bundle of binary classifiers. Then an agglomerative hierarchical clustering method will provide a binary tree or *dendrogram* with classes placed at leaves. The dendrogram can be broken at different levels to yield different clusterings of the set of classes.

The organization of the classes so learned is a meaningful tree that will be easily interpreted by an expert of the domain. Figure 1 depicts the relationships of classes of the well-know *led* dataset. Notice that the topmost split of classes separates those that share the bottom led (left hand side) from those in which that led is off, classes $1, 7$, and 4. As one could hope, the closest classes are the groups $\{8, 0\}$, $\{5, 9, 6\}$, $\{1, 7\}$, and $\{2, 3\}$.

The organization of the paper is the following. In the next section we present an overview of related work. Then, we introduce the central idea of the paper, the definition of the distance of classes that uses an algorithm based on the margin-maximization principle. Additionally, we present a measure to compare different hierarchical clusterings of classes. In Section 5, we report some experimental results supporting our approach. We shall show that our method is able to build meaningful hierarchical clusters of classes, significantly better than those produced by other methods. We close the paper drawing some conclusions.

2 Related Work

There are two kinds of related work. The papers that present biomedical applications, quoted in the Introduction, describe mainly *ad hoc* methods to cluster the set of classes. On the other hand, there are a number of approaches that build clusters trying to assemble multi-class classifiers based on Support Vector Machines (SVM) [13]. This section is devoted to explain the relation of these papers with the method presented here.

First of all, let us recall that there are two types of approaches for multi-class classification using SVM. One is considering all data in a single optimization problem [14,15]. The other is decomposing the multi-class task into a series of binary SVMs, such as One-vs-All (OVA) [13], One-vs-One (OVO) [16], and using a directed acyclic graph (DAG) [17]. As none of these methods significantly outperforms the others, this is an active research subject.

Lately, some authors [7,18,19,20] have proposed decomposition algorithms which follow a similar approach: learn a decision tree (a special case of DAG) based on a hierarchy of classes. First, these methods build a dendrogram of classes; and then, a binary SVM is learned for each internal node of that hierarchy in order to separate the examples of each subset of classes (see Figures 1 and 2). The classification procedure goes from the root to leaves guided by the predictions of SVMs classifiers at internal nodes. All these methods basically differ in the way they build the hierarchy of classes.

In [7] the authors propose a method to construct a binary tree. It proceeds top-down, and at each node it uses a *k-means* clustering of the centroids of classes to divide the set of classes into two groups.

In [19] the authors present the so called Dendrogram-based Support Vector Machines (DSVM). In order to build the dendrogram, DSVM computes the centroid of each class, and then uses an agglomerative hierarchical algorithm. We shall include this method in the Experimental Results section.

In [18], the algorithm Half-Against-Half (HAH) is presented. After some discussion about different methods to build the hierarchy structure (generating it at random or using prior knowledge), the authors propose to use a hierarchical clustering algorithm based on the mean distance between classes. Since the authors do not state clearly the concrete hierarchical clustering algorithm used, we shall not include this method in Section 5.

However, the work that inspired this paper is [20]. In fact, our approach can be seen as a generalization of it. The authors present a multi-class classifier for high-dimensional input spaces, called *Margin Trees Classifier*, that achieves an accuracy comparable to that of OVO SVM on 7 cancer microarray data sets. The algorithm is somehow similar to those cited before since it uses a hierarchy of classes to build a decision tree classifier. The authors report a study comparing different procedures to build the hierarchy: complete linkage, single linkage and a greedy algorithm. They conclude proposing the complete linkage.

However, the main idea introduced by Tibshirani and Hastie is that they use the margin to compute the distance between two classes. In their approach, the dimension of the input space is always greater than the number of examples. Therefore, all classes are separable. In the next section we present a generalization to the non-separable case of this method, applying the basic principles of margin maximization.

In our opinion, another important contribution of [20] is that the authors are the first to remark the additional interpretability of the model obtained in biological tasks. Despite they describe their method only as a multi-class classifier, they also point out the utility of the cluster of classes in real applications. In this paper we want to explore this idea and we shall focus in the method as a clustering of classes.

3 Soft Margin Trees

Let $\mathcal{X}$ be an input space, and $\mathcal{Y} = \{C_1, ..., C_k\}$ a finite set of classes. We consider a multi-classification task given by a training set $S = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$ drawn from an unknown distribution $Pr(X, Y)$ from the product $\mathcal{X} \times \mathcal{Y}$. Within this context, our learning task is to build a dendrogram T in which each class labels exactly one leave, T has k leaves, and $k - 1$ internal nodes.

In Figure 1 you can see the dendrogram obtained applying the method described here to the led dataset [21]. Here, y-axis represents the distance between clusters grouped together.

In order to define an algorithm for agglomerative hierarchical clustering we require two elements:

1. A linkage scheme to recalculate inter-cluster distances when the two most similar or near clusters are merged.
2. A method to calculate a symmetric dissimilarity matrix D based on pairwise dissimilarities or distances. The value in the l-th row, m-th column is the distance between classes C_l and C_m. Notice that it will be necessary to apply this method $\binom{k}{2}$ times to calculate D.

In the following subsections we describe our proposals in these elements of the method for clustering classes.

3.1 Linkage Method

An agglomerative hierarchical clustering algorithm starts defining one cluster for each class. Then, the algorithm proceeds iteratively joining together (*linking*) the two closest clusters. Differences between linkage methods arise from different ways to define distances between clusters. Thus, using different linkage methods can produce different dendrograms. Here, due to the lack of space, we can not study the effect of using different linkage methods.

Following the conclusions drawn by [20], we shall use the *complete linkage* method. In their experimental results all methods produce the same accuracy, but complete linkage gives rise to more balanced trees. This kind of trees are more interpretable than those obtained by other methods. In the experiments reported in Section 5 all hierarchical clustering algorithms compared use the complete linkage method. They only differ in the way they compute the dissimilarity matrix.

3.2 A Margin-Based Metric for Classes

In [20], the distance between two separable classes is defined as the margin between them. To compute the distance between classes C_l and C_m, the class labels (y_i) of the examples of those classes are relabeled (y'_i) as $+1$ and -1, respectively. The hyperplane of maximum margin that separates the nearest examples of those classes is defined by a weight vector $\boldsymbol{w}$ and a bias b that can be obtained solving the following optimization problem:

$$\begin{aligned}(\boldsymbol{w}, b) &= \underset{\|\boldsymbol{w}\|=1}{\operatorname{argmax}} (M) \\ &\text{s.t. } y'_i(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b) \geq M, \quad \forall y_i \in C_l \cup C_m.\end{aligned} \tag{1}$$

Finally, in Margin Trees, the distance between classes C_l and C_m is the margin:

$$D(l, m) = 2 \cdot M. \tag{2}$$

Actually, the optimization problem in Equation 1 is equivalent, see for instance [22,23], to a typical hard-margin SVM formulation:

$$\begin{aligned}&\min \ \frac{1}{2}||\boldsymbol{w}||^2 \\ &\text{s.t. } \ y'_i(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b) \geq 1, \quad \forall y_i \in C_l \cup C_m.\end{aligned} \tag{3}$$

In this case, the margin between both classes is equal to:

$$D(l, m) = \frac{2}{||\boldsymbol{w}||}. \tag{4}$$

Our proposal is to use a *soft*-margin SVM formulation instead of a hard-margin one:

$$\begin{aligned} \min \ & \frac{1}{2}||\boldsymbol{w}||^2 + C \sum_{\forall y_i \in C_l \cup C_m} \xi_i, \\ \text{s.t.} \ & y_i'(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0, \qquad \forall y_i \in C_l \cup C_m, \end{aligned} \tag{5}$$

where C is a regularization parameter. The distance between classes C_l and C_m is defined by the following expression:

$$D(l,m) = \frac{1}{\frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{\forall y_i \in C_l \cup C_m} \xi_i}. \tag{6}$$

The idea is that when classes are very different, the classifier will be given by a simple model (i.e., $||\boldsymbol{w}||$ will be low) and/or the number of misclassified examples or points inside the margin will be small ($\sum \xi_i \to 0$), so the distance (Equation 6) will be high. When classes are similar, the model will be complex (the norm of weight vector will be high) and/or there will be a lot of misclassified examples or points inside the margin ($\sum \xi_i \gg 0$); in this case the distance will be small. Thus, the optimization problem in Equation 5 minimizes an expression that captures faithfully the differences between two classes.

It must be noted that, in order to ensure that all distances of matrix D share an identical scale, the regularization parameter C must be the same in the $\binom{k}{2}$ SVM binary classifiers needed to calculate all pairwise distances.

The differences between distances of Margin Trees and those proposed in this paper may be quite subtle in linearly separable cases. Then, both depend only on the norm of weight vectors, but these vectors may differ. However, in general, our metric has a couple of advantages over that of Margin Trees: 1) it can be applied to non-separable problems, and 2) even in a separable case, the soft-margin approach can find a different solution because it takes into account, at the same time, the complexity of the model and the expected loss. In fact, these are the same benefits that can be obtained using the soft margin approach instead of the hard margin in traditional SVM.

Figure 2 shows these ideas. There are some examples in a 2-dimensional space labeled in four classes linearly separable. At a first glance, it seems that there are two groups of classes: {1,3} and {2,4}. Applying the Margin Trees algorithm (left side panel), Equations 1 or 3, the two most similar classes are 1 and 2; a couple of outlayers make the hard margin quite small. The separating hyperplane and margin frontiers are displayed in Figure 2. On the other hand, our method (right side panel), Equation 5, takes advantage of the possibility to misclassify some examples or to place them inside the margin. Therefore, the method proposed in this paper captures the relationship between those classes better than Margin Trees.

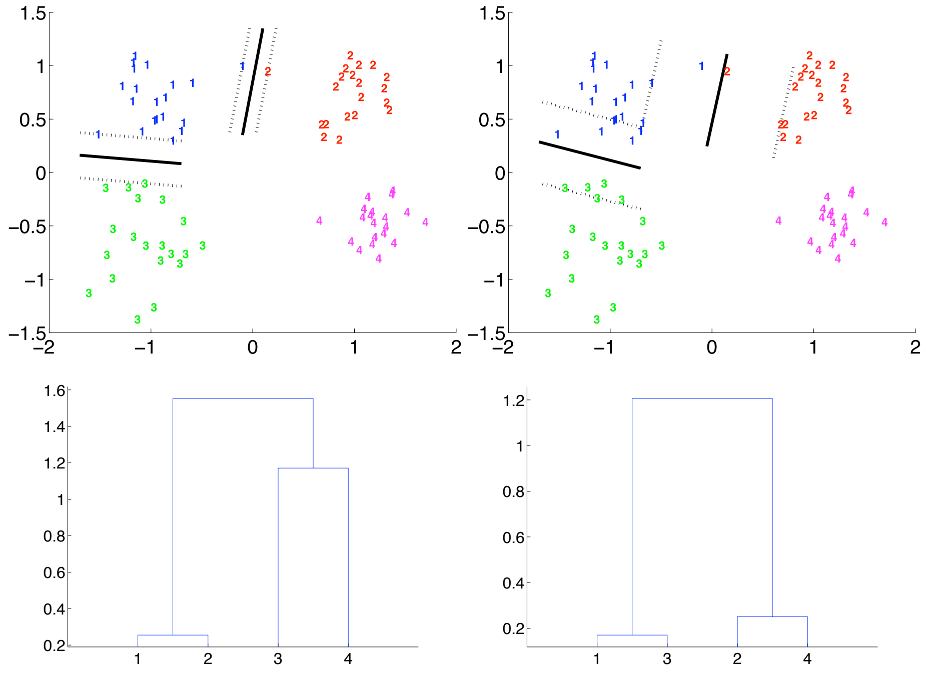

Fig. 2. Example of a Margin Tree (left panel) and a Soft Margin Tree (right panel). Separating hyperplanes and hierarchical clustering of classes are displayed. The method proposed in this paper captures the relationship between those classes better than Margin Trees.

4 A Method and a Metric to Compare Hierarchical Clustering of Classes

A very important aspect of learning methods is to measure the quality of the obtained knowledge. In supervised learning this task is accomplished by a set of well established metrics. They allow users to compare different techniques and to estimate the future performance of predictive models. But unsupervised learning algorithms, like clustering, are difficult to compare.

The learning task of clustering a set of classes is an unsupervised task, we do not know the *correct* organization of classes. However, we can take advantage of having a dataset with examples labeled by those classes.

Given a hierarchical clustering of k classes, as was mentioned in Section 2, it is possible to build a classifier learning $k-1$ binary SVMs. Let $h : \mathcal{X} \to \mathcal{Y}$ be such a classifier. A first approach to define a metric for the quality of clusterings of classes could be to measure the accuracy of h. In fact, if the hierarchy is *good*, the accuracy of h must be high.

Table 1. Description of the datasets used in the experiments. They are divided in two parts. The first is a collection of classification tasks non-linearly separable drawn from the UCI repository [21]. The second part are cancer datasets that were previously used in [20].

Dataset	#classes	#examples	#features
zoo	7	101	16
glass	6	214	9
ecoli	8	336	7
dermatology	6	366	33
vehicle	4	846	18
vowel	11	990	11
led	10	1000	7
yeast	10	1484	8
car	4	1728	6
image	7	2310	19
landsat	6	6435	36
brain	5	42	5597
lymphoma	3	62	4026
srbct	4	63	2308
stanford	14	261	7452
9 tumors	9	60	7131
11 tumors	11	174	12533
14 tumors	14	190	16063

But this is an incomplete view of the problem. It is more important to study what happens when the classifier fails, than to calculate only the accuracy. A hierarchical clustering of classes represents the similarity between those classes. When h missclassifies an example $\boldsymbol{x}_i$, if the hierarchy is good, the predicted class $h(\boldsymbol{x}_i)$, must be near (in the hierarchy) to the true class y_i .

Therefore, we propose to measure the distance in the hierarchy between predictions and true classes; that is, the number of arcs in the dendrogram T placed between the leaves labeled by those classes. We called this measure *Prediction Distance (PD)*:

$$PD(h(\boldsymbol{x}_i), y_i, T) = \#\texttt{arcs}(h(\boldsymbol{x}_i), y_i, T). \tag{7}$$

For instance, the Prediction Distance of an example of class 5 (see Figure 1) classified as class 9 is 2, but if prediction were class 2, PD would rise to 5. The farthest classes from class 5 are 1 and 7, both at distance 8. Notice that the maximum distance between two classes in a dendrogram of k classes is k.

Averaging this metric over a set of test examples we shall measure the goodness of a hierarchy T.

5 Experimental Results

In order to evaluate the benefits of our approach we conducted a battery of experiments. The aim is to show that our approach is able to build meaningful hierarchical clusters of classes, significantly better than those produced by other methods.

As it was indicated in Section 3.1, each algorithm for hierarchical clustering employed in the experiments used the complete linkage method to build the tree. Therefore, the differences between the algorithms arise from the way they compute the matrixes of distances between each pair of classes. We considered four approaches:

– *Random.* A random symmetrical distance matrix is computed. Since *PD* is an unbounded measure, we used *Random* method to obtain an upper bound on all datasets. It is expected that the other approaches perform significantly better.
– *DSVM* [19]. Dendrogram-based Support Vector Machines computes the distance matrix by means of the Euclidean distance between the centroids of each pair of classes.
– *Margin Trees (MT).* The distance matrix is computed using the Margin Trees idea from [20], showed in Equations 3 and 4.
– *Soft Margin Trees (SMT).* The distance matrix is computed applying Equations 5 and 6.

Once we have built a hierarchical clustering of classes, a multi-class classifier may be learned with a binary SVM attached to each internal node. We shall report the accuracy of these classifiers ("0/1" errors) in addition to the Prediction Distance (*PD*) (Equation 7). Although the quality of the trees can be measured only by means of the prediction distance, we include the accuracy to compare their predictive power with the *SVM* multiclass.

The datasets used in the experiments are described in Table 1. There are datasets with more number of examples than features and vice versa. The first group is formed by datasets obtained from the UCI repository [21]. We selected those datasets that fulfill the following rules: continuous or ordinal attribute values, no more than 40 attributes, no more than 10000 examples, and excluding datasets with missing values and with less than 4 classes. The second group is composed by the datasets used in [20]. They are 7 datasets which target is to classify cancer patients from gene expressions captured by microarrays.

All the scores reported were estimated by means of a 5-fold cross validation repeated 2 times. We did not use the 10-fold procedure, since in certain datasets there are too few examples in some classes. The *SVM* implementation used was *libsvm* [24] with the linear kernel in all cases. Cancer datasets can be separated by a linear-SVM, so the utility of using different kernels it is not clear.

For each hierarchical system considered (*Random, DSVM, MT and SMT*), we used the same regularization parameter C in all models learned at each node in the tree. To select this parameter, we utilized a 2-fold cross validation repeated 5 times on training data searching within $C \in [10^{-2}, \dots, 10^{2}]$. The aim in this

Table 2. Cross validation results both in "0/1" errors and *PD*

	SVM	*Random*		*DSVM*		*MT*		*SMT*	
Dataset	0/1	0/1	*PD*	0/1	*PD*	0/1	*PD*	0/1	*PD*
zoo	4.98	5.48	0.2336	4.50	0.2750	*MT* classifier only works	with separable datasets	4.98	0.1788
glass	33.20	40.44	1.7156	35.76	1.1292			37.40	1.1502
ecoli	11.75	16.58	0.6888	14.01	0.3873			13.71	0.3782
dermatology	3.55	3.14	0.1312	4.36	0.1036			3.82	0.0928
vehicle	20.33	22.99	0.7400	20.63	0.4817			20.63	0.4805
vowel	19.19	54.60	3.2303	26.77	1.1354			29.70	1.3449
led	27.70	36.95	2.0790	27.80	1.1950			27.95	1.1860
yeast	41.61	46.36	2.2685	42.79	1.4340			43.43	1.3672
car	14.35	19.50	0.6754	15.11	0.4751			17.30	0.4161
image	4.07	12.53	0.5890	10.13	0.3812			4.29	0.1126
landsat	13.05	18.91	0.7876	14.96	0.5829			13.18	0.4051
brain	13.19	14.44	0.5278	14.58	0.5611	13.33	0.5125	13.33	0.5125
lymphoma	0.00	0.00	0.0000	0.00	0.0000	0.00	0.0000	0.00	0.0000
srbct	1.60	1.60	0.0564	0.83	0.0167	0.83	0.0250	0.83	0.0250
stanford	5.36	5.35	0.3759	6.12	0.2700	5.93	0.3162	5.93	0.3162
9 tumors	48.33	51.67	2.4667	45.83	2.3083	44.17	2.3083	44.17	2.3083
11 tumors	10.08	9.81	0.5603	9.81	0.5492	10.09	0.5206	10.09	0.5206
14 tumors	30.53	32.11	2.0368	31.58	1.9026	28.68	1.5289	28.68	1.5289
Average	16.83	21.80	1.0646	18.09	0.7327			17.75	0.6847

grid search was to optimize the *PD*. In the case of the *SMT*, the C parameter found by the search was used both for computing the distance matrix and to build the classifiers of each internal node.

Following [25], we used the Wilcoxon signed ranks test to compare the performance of the classifiers by pairs when the measurements are "0/1" errors or *PD*.

In Table 2, we show the results obtained in the experiments. Considering the quality of the hierarchies measured by *PD*, we can see that *SMT* achieved the best results. The differences are significant with $p < 0.01$ for *Random*, and with $p < 0.03$ for *DSVM*. The *MT* algorithm was not applied on UCI datasets since they are not separable. To summarize the results, in Table 3 we show the number of victories, defeats and ties between each pair of algorithms. We can see that *SMT* is the unique algorithm that never loses *versus Random* in all datasets; and it only ties ones, in *lymphoma* dataset where all algorithms obtain the best performance.

Additionally, it is important to remark that *SMT* and *MT* achieve the same results in those datasets in which the number of features is higher than the number of examples.

With respect to "0/1" errors, Table 2 shows that the results attained by *SVM* multiclass are better in general (as it happens in [20]). Actually, *SVM* is significantly better than *Random* and *DSVM* with $p < 0.01$, and better than *SMT* with $p < 0.04$. Comparing multi-class classifiers attached to hierarchies of

Table 3. Summary of *PD* scores. Number of wins (w), losses (l), and ties (t) between pairs of algorithms considered. In the case of *DSVM versus SMT* (4/12/2), it must be read as follows: *DSVM* is better than our approach *SMT* 4 times, worse 12 times, and in 2 datasets they obtain equal results.

w/l/t	*MT*	*SMT*	*Random*
DSVM	2/3/2	4/12/2	15/2/1
MT		0/0/7	6/0/1
SMT			17/0/1

classes, *SMT* is significantly better than *Random* with $p < 0.01$. These results show that these methods have less predictive power than *SVM*, although they produce valuable descriptive models represented in the hierarchies of classes.

Notice that sometimes an algorithm A obtains better "0/1" errors, but a worse *PD* than other algorithm B. Even in that case, these scores mean that the hierarchy learned by A is worse than that produced by B. The reason is that although the hypothesis h_A learned by A missclassifies less examples than the corresponding hypothesis h_B of B, the errors of h_A are far in the corresponding hierarchy from true classes. On the other hand, the more frequent misclassifications of h_B are close to the true classes.

6 Conclusions

In this paper we presented a new method for clustering a set of classes of a multi-class learning task. These clusterings have shown their usefulness in fields such as *medicine* or *genetics of populations*. The aim in all these cases is not only to learn an accurate classifier but also to gain some insight into the application domain.

Our approach computes a distance matrix between classes using a method based on the soft margin of each pair of classes. Then, using a complete linkage method, it is possible to build a dendrogram where the leaves are labeled by classes. We tested *Soft Margin Trees* with other algorithms for clustering classes and it was shown that *SMT* produces significantly better hierarchical clusters of classes than those produced by other methods.

Acknowledgments

The research reported here is supported in part under grant TIN2008-06247 from the MICINN (Ministerio de Ciencia e Innovación of Spain).

References

1. Patra, J., Ang, E.L., Meher, P., Zhen, Q.: A new som-based visualization technique for dna microarray data. In: IJCNN 2006. International Joint Conference on Neural Networks, pp. 4429–4434 (2006)

2. Ng, R., Sander, J., Sleumer, M., et al.: Hierarchical cluster analysis of SAGE data for cancer profiling. In: Proceedings of BIOKDD 2001 Workshop on Data Mining in Bioinformatics, pp. 65–72 (2001)
3. Hanotte, O., Bradley, D.G., Ochieng, J.W., Verjee, Y., Hill, E.W., Rege, J.E.O.: African Pastoralism: Genetic Imprints of Origins and Migrations. Science 296(5566), 336–339 (2002)
4. Vargo, E.: Hierarchical analysis of colony and population genetic structure of the eastern subterranean termite, reticulitermes flavipes, using two classes of molecular markers. Evolution 57(12), 2805–2818 (2003)
5. Li, M., Tapio, I., Vilkki, J., Ivanova, Z., Kiselyova, T., Marzanov, N., Cinkulov, M., Stojanovic, S., Ammosov, I., Popov, R., Kantanen, J.: The genetic structure of cattle populations (Bos taurus) in northern Eurasia and the neighbouring Near Eastern regions: implications for breeding strategies and conservation. Molecular Ecology 16(18), 3839–3853 (2007)
6. Rosenberg, N.A., Pritchard, J.K., Weber, J.L., Cann, H.M., Kidd, K.K., Zhivotovsky, L.A., Feldman, M.W.: Genetic Structure of Human Populations. Science 298(5602), 2381–2385 (2002)
7. Vural, V., Dy, J.G.: A hierarchical method for multi-class support vector machines. In: ICML 2004: Proceedings of the twenty-first international conference on Machine learning, pp. 105–112. ACM Press, New York (2004)
8. Kohonen, T.: Self-organizing maps. Springer-Verlag New York, Inc., Secaucus (1997)
9. Flexer, A.: On the use of self-organizing maps for clustering and visualization. In: Principles of Data Mining and Knowledge Discovery, pp. 80–88. Springer, Heidelberg (1999)
10. Vesanto, J.: Som-based data visualization methods. Intelligent Data Analysis 3, 111–126 (1999)
11. Nikkilä, J., Törönen, P., Kaski, S., Venna, J., Castrén, E., Wong, G.: Analysis and visualization of gene expression data using Self-Organizing Maps. Neural Networks 15(8-9), 953–966 (2002)
12. Johnson, S.: Hierarchical clustering schemes. Psychometrika 32(3), 241–254 (1967)
13. Vapnik, V.: Statistical Learning Theory. John Wiley, New York (1998)
14. Weston, J., Watkins, C.: Multi-class support vector machines. In: Verleysen, M. (ed.) Proceedings of the 6th European Symposium on Artificial Neural Networks (ESANN), D. Facto Press, Brussels (1999)
15. Crammer, K., Singer, Y.: On the algorithmic implementation of multiclass kernel-based vector machines. Journal of Machine Learning Research 2, 265–292 (2001)
16. Kreßel, U.: Pairwise classification and support vector machines. In: Schölkopf, B., Burges, C.J.C., Smola, A.J. (eds.) Advances in Kernel Methods – Support Vector Learning, pp. 255–268. MIT Press, Cambridge (1999)
17. Platt, J.C., Cristianini, N., Shawe-taylor, J.: Large margin dags for multiclass classification. In: Advances in Neural Information Processing Systems, pp. 547–553. MIT Press, Cambridge (2000)
18. Lei, H., Govindaraju, V.: Half-against-half multi-class support vector machines. In: Oza, N.C., Polikar, R., Kittler, J., Roli, F. (eds.) MCS 2005. LNCS, vol. 3541, pp. 156–164. Springer, Heidelberg (2005)
19. Benabdeslem, K., Bennani, Y.: Dendogram based svm for multi-class classification. In: 28th International Conference on Information Technology Interfaces, pp. 173–178 (2006)
20. Tibshirani, R., Hastie, T.: Margin Trees for High-dimensional Classification. Journal of Machine Learning Research 8, 637–652 (2007)

21. Asuncion, A., Newman, D.J.: UCI machine learning repository. School of Information and Computer Sciences. University of California, Irvine, California, USA (2007)
22. Boser, B.E., Guyon, I., Vapnik, V.: A training algorithm for optimal margin classifiers. Computational Learing Theory, 144–152 (1992)
23. Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines and other kernel-based learning methods. Cambridge University Press, Cambridge (2000)
24. Wu, T., Lin, C., Weng, R.: Probability Estimates for Multi-class Classification by Pairwise Coupling. The Journal of Machine Learning Research 5, 975–1005 (2004)
25. Demšar, J.: Statistical Comparisons of Classifiers over Multiple Data Sets. Journal of Machine Learning Research 7, 1–30 (2006)