Title: Probabilistic Class Hierarchies for Multiclass Classification

Author: Daniel Silva-Palacios Cèsar Ferri María José Ramírez-Quintana

PII: S1877-7503(17)31218-8
DOI: https://doi.org/doi:10.1016/j.jocs.2018.01.006
Reference: JOCS 824

To appear in:

Received date: 1-11-2017
Accepted date: 28-1-2018

Please cite this article as: Daniel Silva-Palacios, C*graveesar* Ferri, María José Ramírez-Quintana, Probabilistic Class Hierarchies for Multiclass Classification, *<![CDATA[Journal of Computational Science]]>* (2018), https://doi.org/10.1016/j.jocs.2018.01.006

Probabilistic Class Hierarchies for Multiclass Classification

Daniel Silva-Palacios, Cèsar Ferri, María José Ramírez-Quintana[1]

DSIC, Universitat Politècnica de València,
Camí de Vera s/n, 46022, Valencia, Spain

Abstract

The improvement in the performance of classifiers has been the focus of attention of many researchers over the last few decades. Obtaining accurate predictions becomes more complicated as the number of classes increases. Most families of classification techniques generate models that define decision boundaries trying to separate the classes as well as possible. As an alternative, in this paper, we propose to hierarchically decompose the original multiclass problem by reducing the number of classes involved in each local subproblem. This is done by deriving a similarity matrix from the misclassification errors given by a first classifier that is learned for this, and then, using the similarity matrix to build a tree-like hierarchy of specialized classifiers. Then, we present two approaches to solve the multiclass problem: the first one traverses the tree of classifiers in a top-down manner similar to the way some hierarchical classification methods do for dealing with hierarchical domains; the second one is inspired in the way probabilistic decision trees compute class membership probabilities. To improve the efficiency of our methods, we propose a criterion to reduce the size of the hierarchy. We experimentally evaluate all of the proposals on a collection of multiclass datasets showing that, in general, the generated classifier hierarchies outperform the original (flat) multiclass classification.

Keywords: `Multiclass classification, Class hierarchy`

[☆]This is an extended version of our conference paper that was invited to the JoCS special issue (https://doi.org/10.1016/j.procs.2017.05.218).

[1]Email: dasilpa@posgrado.upv.es (Silva), {cferri,mramirez}@dsic.upv.es (Ferri,Ramirez)

inference, Hierarchy of classifiers

1. Introduction

In machine learning, classification is the problem of identifying to which of a set of categories (classes) a new instance belongs. When the categories contain more than two different labels, the problem is distinguished as multiclass classification. In the basic scenario of multiclass classification, it is assumed that: 1) only one class label is assigned to each instance, (i.e., this is a single-class classification as opposed to multi-label classification, which allows multiple class labels for each instance); and 2) class labels are independent, (i.e., there are not relationships among class labels as opposed to hierarchical classification problems where classes are organised into a hierarchical structure).

One of the main objectives when solving a classification task is to make predictions as "precise" as possible, where the notion of "precise" depends on the evaluation measure that is used to assess the quality of the classifier. For instance, accuracy is one of the most popular evaluation measures used to assess classifier performance. However, obtaining good predictions is not always a simple task. This is especially complicated in multiclass problems since the classifier needs to select among a high number of classes in order to make the predictions. For this reason, in the last few decades, numerous efforts have been made to improve classifier performance in multiclass problems. These approaches can be divided into one of the following three groups:

a) *Learning technique specialisation*: For instance, the notion of *multiclassifier or ensemble learning*, which is a general approach based on the idea of using more than one model to obtain predictions (as a combination of individual predictions). There are two main strategies to construct a multiclassifier: to use the same learning technique to create all of the models [1, 2, 3], and to use a different learning technique to build each model in the multiclassifier [4, 5].

b) *Training data transformation*: For instance, approaches based on instance and feature selection [6, 7], which are developed with the aim of selecting the most relevant instances or features to be used in the model construction and also oversampling and undersampling methods [8, 9], which are proposed

to deal with imbalanced datasets that have been shown to usually enhance classification performance [10].

c) *Prediction adjustment*: A common way to improve estimated probabilities is to apply calibration techniques [11, 12, 13] in order to approximate the predicted scores to the actual probabilities.

All of the above-mentioned methods and strategies assume that there does not exist any relationship between class labels (i.e., class labels are independent).

Another possible approach that has been explored (also with the core target of improving multiclass classification accuracy) consists in decomposing the multiclass problem into a collection of (somehow related) subproblems that are smaller or simpler than the original one.

Two alternative ways of decomposing the original problem have been studied. The first way relies on the idea that a problem becomes less complex if its dimensionality is reduced. For instance, in [14] the instance attributes are iteratively split into disjoint sets and then a new classification problem is defined for each partition. The second way relies on the idea that a multiclass problem becomes simpler if the number of classes is reduced. For instance, in [15] and [16], a class hierarchy is constructed (by assuming that there exists some relationship between the class labels) and then each internal node of the hierarchy defines a new classification problem involving only its children class labels. This second way of addressing the multiclass problem is inspired in the top-down hierarchical classification method. The different approaches proposed mainly differ in how the class hierarchy is automatically generated (based on instances or based on predictions) and/or which learning technique is used for learning the intermediate classifiers.

In this paper, we also propose to decompose the original multiclass problem by using a class hierarchy from which a tree-like structure of classifiers is constructed. Similar to [16], the relationship between classes is derived from the confusion matrix of a "flat"[2] classifier (interpreting the confusion matrix as an indicator of how simple or hard it is for the classifier to distinguish the classes). However, instead of deriving the similarity between classes directly from the values of the confusion matrix (as [16] does), we transform the

[2]In the literature, it is common to refer to a classifier that does not take into consideration any relationship between the class labels as "flat".

matrix trying to obtain as much information as possible from it. From the transformed matrix, we define a semi-metric function that is used to generate the class hierarchy. With the aim of more precisely deriving the similarity between classes, in this paper we propose using two different approaches: the confusion matrix of the multiclass classifier, and a version of the similarity matrix based on the probability estimations of the classifier. We also propose two alternatives to apply the hierarchy of local classifiers for classifying new instances: the first one consists in applying the set of classifiers in a branch (from the root to a leaf), whereas the second one consists in combining all of the classifiers in the hierarchy by employing the estimated probabilities. We experimentally evaluate our proposals using a large collection of datasets and techniques, and we analyse how our methods behave when classes are unbalanced.

The paper is organized as follows. In Section 2, we review some previous works in the field of class hierarchy learning. Section 3 presents our method for defining the similarity between classes from the confusion matrix by using both class estimations and probability estimations. In Section 4, we describe how to decompose a multiclass problem according to the class hierarchy, and then we present two different methods to solve the original problem using the hierarchy of classifier generated. Section 5 presents different experiments that we conducted in order to evaluate our approach. Finally, Section 6 concludes the paper.

2. Related Work on Class Hierarchy Generation

Typically, the different approaches for automatically generating class hierarchies are divided into two groups according to how the underlying distance is defined: *instance-based* methods and *prediction-based* methods.

The so-called instance-based methods use any distance or similarity function d defined between the instances, and then the class hierarchy is built by applying a clustering algorithm using d. The most commonly applied clustering algorithm is the Hierarchical Agglomerative Clustering (HAC), but others techniques have also been applied (such as Kmeans [17] or the Formal Concept Analysis [18]). Most approaches in the context of ontology construction [19, 20, 21] and concept hierarchy learning from texts [22] fall into this category. Other instance-based approaches first pre-process or transform the instances and then the hierarchy is induced by using any distance function between the transformed items. For instance, [15] applies a linear

discriminant projection to transform documents (represented as vectors) into a low-dimensional space, and then a similarity between two classes is defined as the distance between their centroids. A similar approach is presented in [23] for learning the ontologies used in a recommender system. From the user-item matrix that describes the item ratings given by the users, the authors derive a collection of ontologies by using several distance functions and applying both agglomerative and divisive clustering.

The prediction-based approaches use the predictions given by a classifier that is, in many cases, specially constructed for this. For the framework of multi-label hierarchical classification, in [24], an ARTMAP neural network is used as a self-organizing expert system to derive hierarchical knowledge structures. A similar approach is presented in [25], where the authors use an association rule learner that extracts class hierarchies from the predictions given by a multi-label classifier. In the field of (non-hierarchical) multiclass classification, [16] presents an approach to improve document classification by combining Naive Bayes (NB) and Support Vector Machines (SVM). Specifically, once a NB classifier has been learned, each row of its confusion matrix (after normalising it) is used to represent each class as a tuple of numerical values. Classes are compared by applying the Euclidean distance.

Although it is also prediction-based, our proposal differs from the last approaches just mentioned in that the similarity between classes is not obtained from either the predictions of the classifier or by applying any well-known distance function over the values of the confusion matrix. Instead, the confusion matrix is turned into a similarity matrix.

3. A New Prediction-based Approach for Learning Class Hierarchies

In this section, we present our proposal for learning class hierarchies using a classifier. We define the similarity between classes based on a confusion matrix. The matrix is built by using two different methods: the first method uses the class estimations of the classifiers, while the second method considers the class probability estimations. The hierarchy is obtained from the confusion matrix by applying an HAC algorithm.

3.1. Calculating the similarity between classes

Let $C = \{c_1 \dots c_n\}$ be a set of n class labels and let D be a set of labeled instances of the form $\langle x, y \rangle$ where x is a m-tuple whose components

are the input attributes and $y \in C$ is the target attribute, i.e. the class. Given a flat multiclass classifier F trained using D, its confusion matrix M is an $n \times n$ matrix that describes the performance of F. The rows of M represent the instances in actual classes, whereas the columns represent the instances in predicted classes. Thus, the elements of M placed at the main diagonal (M_{c_i,c_i}) are the instances of D that F correctly classifies, whereas the rest of elements of M are the misclassification errors, that is $M_{c_i,c_j}, c_i \neq c_j$, represents the instances of class c_i that F classifies as being of class c_j. For the sake of simplicity, in what follows, we denote any class c_i by its subindex i.

In general, in any confusion matrix, the following can be observed: 1) misclassification errors are not usually uniformly distributed, which indicates that it is more difficult for the classifier to separate some classes than others; and 2) M is usually non-symmetrical, which means that to really measure the degree of confusion between two classes i and j, we must take into account all of the errors the classifier makes involving both classes, i.e., $M_{i,j}$ and $M_{j,i}$. Our proposal is based on this reasoning.

Given a confusion matrix M, we denote as $\overline{M}$ the result of normalising M (by rows), i.e. $\overline{M}_{ij} = \frac{M_{ij}}{\sum_{k=1}^{n} M_{ik}}$.

From $\overline{M}$ we formally define the similarity between classes regarding their level of distinguishability:

Definition 1. Class overlapping

The degree of overlapping between two classes i and j is

$$overlap(i,j) = \begin{cases} \frac{\overline{M}_{ij}+\overline{M}_{ji}}{2} & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases}$$

The result of applying the overlap function to $\overline{M}$ is a symmetric matrix $\overline{M}_O$ that we call the *Overlapping Matrix*.

Definition 2. Class similarity

The similarity between two classes i and j is

$$d_S(i,j) = 1 - overlap(i,j)$$

The result of applying d_S to $\overline{M}$ is also a symmetric matrix $\overline{M}_S$ that we call the *Similarity Matrix*. Note that $d_S(i,j) \in [0..1], \forall i,j \in C$. This means

that $d_S(i,j) = 0$ when classes i and j are completely indistinguishable from each other, whereas $d_S(i,j) = 1$ indicates that the classes are completely overlapping.

Remark 1. We remark that, (C, d_S) is a semi-metric space [26] since d_S satisfies the non-negativity, symmetry, and identity conditions; however, for some pairs of classes, d_S can violate the triangle inequality property (as shown in Example 1).

Example 1. Let us consider a classification problem with four class labels $C = \{a, b, c, d\}$. Suppose that we have trained a multiclass classifier F (using any classification technique) whose confusion matrix M is depicted in Table 1a. From M we can see that F perfectly classifies the instances belonging to class d but makes a lot of mistakes classifying the instances belonging to the other classes. In fact, we could say that F is a poor classifier (whose accuracy is 0.55). Tables 1b to 1d show (step by step) the different matrices obtained by applying our method.

		Predicted			
		a	**b**	**c**	**d**
Real	**a**	20	0	30	0
	b	0	20	30	0
	c	30	30	10	0
	d	0	0	0	100

(a) Confusion Matrix M.

	a	**b**	**c**	**d**
a	0.400	0.000	0.600	0.000
b	0.000	0.400	0.600	0.000
c	0.429	0.429	0.143	0.000
d	0.000	0.000	0.000	1.000

(b) Normalised Confusion Matrix $\overline{M}$.

	a	**b**	**c**	**d**
a	**1**			
b	0.000	**1**		
c	0.514	0.514	**1**	
d	0.000	0.000	0.000	**1**

(c) Overlapping Matrix $\overline{M}_O$.

	a	**b**	**c**	**d**
a	**0**			
b	1.000	**0**		
c	0.486	0.486	**0**	
d	1.000	1.000	1.000	**0**

(d) The similarity Matrix $\overline{M}_S$.

Table 1: Example that illustrates the calculation of the similarity between classes from the confusion matrix of a multiclass classifier.

In this example, it is easy to see that the triangle inequality property does not hold in d_S since for classes a and b it holds that

$$d_S(a,b) = 1 > d_S(a,c) + d_S(c,b) = 0.972$$

3.2. Calculating class similarity by using estimated probabilities

For most real problems, it is more important and useful to know not only the predictions given by the model but also how confident the classifier is in them. For instance, suppose that a bank is using a classifier to make decisions about whether or not to grant loans. Imagine that the classifier predicts class "yes" for two customers, p_1 and p_2, meaning that the loan is granted. However, are both decisions equally risky for the bank? Would the bank change its decision if it knew that the classifier is 90% sure when predicting p_1 as being of class "yes" but only 55% for p_2?

A probabilistic classifier (or probability estimator) is a decision system that accompanies each prediction with a probability estimation. Given an instance x, a probabilistic classifier estimates its probability of belonging to each class c_i, $\Pi(c_i \mid x)$, and assigns to x the following vector of predicted probabilities $\vec{p}(x)$:

$$\vec{p}(x) = \langle \Pi(c_1 \mid x), \ldots, \Pi(c_n \mid x) \rangle \quad c_i \in C$$

Finally, the predicted class for x is the class with higher probability, that is,

$$\hat{x} = argmax_{c_i \in C} \langle \Pi(c_1 \mid x), \ldots, \Pi(c_n \mid x) \rangle$$

The confusion matrix of a probabilistic classifier is constructed by adding the estimated probability vectors of all the instances according to their real classes, such that

$$M_{ij} = \sum_{\langle x,i \rangle \in D} \Pi(j \mid x)$$

Table 2 illustrates the process of creating the confusion matrix through the predictions given for three instances e_1, e_2, and e_3 of real classes a, b, and a, respectively.

From the confusion matrix of a probabilistic classifier, we can derive the similarity between classes by following the procedure explained in Section 3.1.

3.3. Learning the Class Hierarchy

Once the distance matrix has been obtained, the class hierarchy is built by applying an agglomerative hierarchical clustering algorithm. The agglomerative approach works in a bottom-up manner starting by assigning each

		Predicted		
		a	b	c
Real	a	0.7	0.1	0.2
	b			
	c			

(a) $\widehat{e_1} =< 0.7, 0.1, 0.2 >$
$real\_class(e_1) = a$

		Predicted		
		a	b	c
Real	a	0.7	0.1	0.2
	b	0.4	0.6	0
	c			

(b) $\widehat{e_2} =< 0.4, 0.6, 0 >$
$real\_class(e_2) = b$

		Predicted		
		a	b	c
Real	a	1.5	0.1	0.4
	b	0.4	0.6	0
	c			

(c) $\widehat{e_3} =< 0.8, 0, 0.2 >$
$real\_class(e_3) = a$

Table 2: Example of creation of the confusion matrix using the predictions given by a probabilistic classifier for three examples e_1, e_2, and e_3.

element to a single cluster (singleton) and then iteratively merging pairs of clusters until only one cluster is obtained. Clusters are joined based on the distance between them, which is referred to as the linkage distance. Linkage distances are, among others, the complete distance (the maximum distance between elements of each cluster), the single linkage distance (the minimum distance between elements of each cluster), and the average linkage distance (the mean distance between elements of each cluster). The hierarchical clustering is graphically represented as a binary tree called a dendrogram that shows both how the clusters are grouped and the distance at which the grouping has taken place.

From the dendrogram, the hierarchy of classes is derived by considering that the clusters created between the leaves (the set of original classes) and the root constitute the internal nodes of the hierarchy. We could obtain different hierarchies depending on the linkage distance used and the kind of estimations (classes or probabilities) used to infer the similarity matrix. This is illustrated in Figure 1 for the Flare dataset from the UCI repository [27]. This figure shows the dendrogram and the corresponding hierarchy generated by the agglomerative clustering using the complete linkage and our semi-metric distance d_S. To derive d_S, we have used both class and probability estimations. Note that, from the point of view of the similarity between classes, estimating class values or membership probabilities (even for the same data) is not equivalent (i.e., the computed d_S function is different) and, hence, it leads to different class hierarchies as shown in Figure 1.

4. Using Class Hierarchies to Decompose Multiclass Problems

To solve the original multiclass problem, we propose learning one classifier at every internal node in the class hierarchy (including the root) in order to

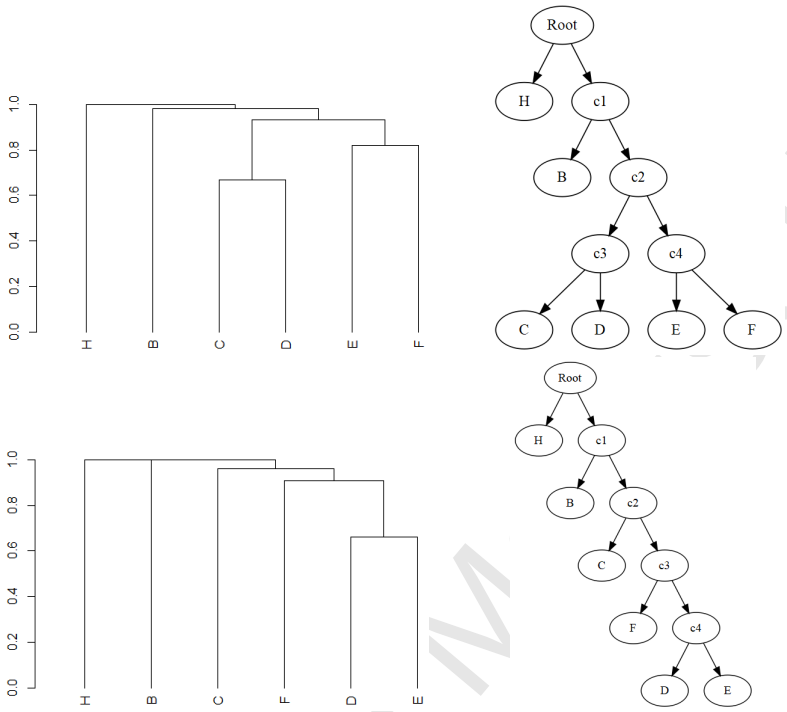


Figure 1: Comparison of the dendrograms (left) and class hierarchies (right) induced by using the confusion matrix generated by a Probabilistic Classifier (top) and a Multiclass Classifier (bottom) for the Flare dataset from the UCI repository.

distinguish between its child nodes. As a result, we obtain a set of classifiers that is arranged in a tree-like structure. This method is inspired in the *Local Classifier per Parent Node (LCPN)* approach [28], which is one of the most commonly used methods in the field of hierarchical classification.

4.1. Class Hierarchy compression

Since the agglomerative clustering algorithm aggregates two groups at each step, the class hierarchy and the classifier tree derived from it are binary trees. That means that for a problem with n classes, we have to train $(n-1)$ flat classifiers. We present a procedure to simplify the class hierarchy in order to reduce the number of internal nodes, and thus the number of flat classifiers we have to train. This procedure, therefore, generally implies a reduction in complexity of the method. In hierarchical clustering, the idea of using a post-process to reduce the dendrogram was also applied in [29] to develop a multidimensional hierarchical clustering.

Algorithm 1: Class Hierarchy Compression with Threshold

```
Function CompressionProcess (class_hierarchy ,threshold) is
    /* Depth-first search of the class hierarchy.                    */
    node ← getRootNode(class_hierarchy);
    node ← VerifyLevel(node);
    children ← getChildren(node);
    foreach child in children do
        node.child ← CompressionProcess(child);
    return node;
Function VerifyLevel(node) is
    /* While there is a child with same height.                     */
    while (exists(node)) do
        children ← getChildren(node);
        foreach child in children do
            nodeHeight ← getHeight(node);
            childHeight ← getHeight(child);
            distance ← nodeHeight - childHeight;
            if (distance <= threshold) then
                node ← JoinNodes(node,child);
                break;
    return node ;
Function JoinNodes(nodeFather,nodeChild) is
    nodeFather ← removeChild(nodeChild);
    decendents ← getChildren(nodeChild);
    foreach descendent in decendents do
        nodeFather ← addChildNode(descendent);
    return nodeFather;
```

In order to reduce the size of the class hierarchy, the dendrogram is traversed in a depth-first search such that if there are two contiguous clusters[3] i and $(i+1)$ whose linkage distance does not exceed a certain threshold θ, i.e., $\mid linkage(i) - linkage(i+1) \mid < \theta$, then the clusters are merged, where $linkage(A)$ is the linkage distance at which cluster A has been created. Algorithm 1 presents the compression procedure.

The threshold can be chosen by the user, be dependent on the domain, or be estimated. In this paper, we propose to determine the threshold from the linkage distances that are employed for the HAC algorithm to group the clusters. More specifically, for a given dendrogram consisting of n clusters, the threshold θ is determined as

$$\theta = \alpha \cdot maximum_{k \in [1..(n-1)]} \mid linkage(k) - linkage(k+1) \mid$$

where $\alpha \in [0..1]$ is the compression parameter. A value $\alpha = 0$ means that

[3]For the sake of simplicity, we use the order that is used to create the clusters to denote them.

only clusters at the same linkage distance will be joined in the post-process, whereas $\alpha = 1$ means that all of the clusters collapse in the dendrogram root[4].

Figure 2 shows the compression process applied to the Flare dataset (from the UCI repository [27]). As can be observed, for $\alpha = 0.1$ (Figure 2b), the nodes *root* and $c1$ in the original hierarchy (2a) are merged into one node that is the root of the resulting hierarchy (2b); in the same way, nodes $c2$ and $c4$ in 2a are merged into one node ($c2$) in Figure 2b. Additionally, for $\alpha = 0.4$ (Figure 2c), the nodes *root*, $c1$, $c2$, and $c3$ are merged into the node root of the resulting hierarchy (2c). As a consequence, the number of flat classifiers to be learned has been reduced from five to three for $\alpha = 0.1$ and from five to two for $\alpha = 0.4$.

In the next section, we experimentally estimate the value of α in order to analyse whether or not it is possible to reach a balanced compromise between reducing the number of classifiers and not losing too much information about the dissimilarity between classes.

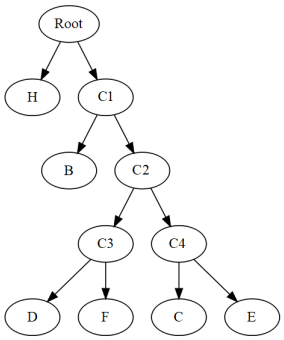
4.2. Top-down Prediction

The first method we propose for making predictions relies on the fact of having a hierarchy of flat classifiers that is composed of binary or multiclass classifiers. In this scenario, to classify a new instance, the tree of classifiers is traversed in a top-down manner applying the classifiers from the root until a leaf is reached. This means that, for a hierarchy made up of n classifiers, the number of flat classifiers we have to apply for classifying an instance varies from $(n/2)$, when the class hierarchy (and the tree of classifiers) is balanced, up to $(n-1)$, when the class hierarchy is a chain.

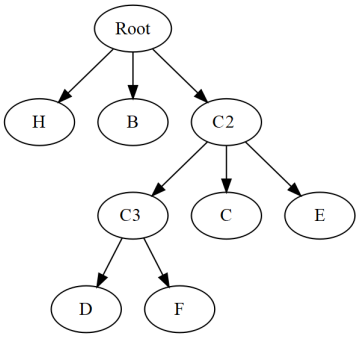
4.3. Top-down and Bottom-up Prediction

The second method we propose for making predictions is devised to work with hierarchies that are composed of flat classifiers that are probability estimators. Given an instance x, the idea is to traverse the tree in a top-down manner applying all of the classifiers and in each node ν estimating the probability of belonging to each of its children. When the leaves are reached, a probability vector is constructed in every leaf. The probability vectors are

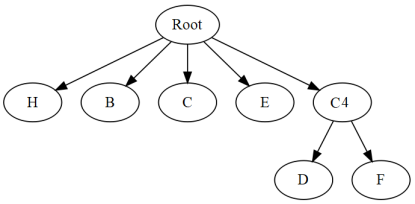
[4]In fact, depending on the dendrogram, the class hierarchy may collapse into the root at a value of $\alpha < 1$.



(a) Original class hierarchy.



(b) Compressed hierarchy for $\alpha = 0.1$.



(c) Compressed hierarchy for $\alpha = 0.4$.

Figure 2: Example of the Hierarchy Compression Process applied to the Flare DataSet.

the conditional probabilities $\langle \Pi(c_1 \mid x), \ldots, \Pi(c_n \mid x) \rangle$. Given that the leaves represent the original classes, the vector in a leaf c_i has all its components equal to zero except the component $\Pi(c_i \mid x) = 1$. In a second stage, the probability vectors are propagated bottom up.

Given a non-leaf node ν in the hierarchy of classifiers, $Chl(\nu)$ denotes the set of nodes that are the children of ν. Then, the probability vector of x at node ν, $\vec{p}(x, \nu)$, is obtained as follows

$$\vec{p}(x, \nu) = \sum_{\mu \in Chl(\nu)} (\Pi(\mu \mid x) \times \vec{p}(x, \mu))$$

Finally, the class predicted for x is the highest component in vector $\vec{p}(x, root)$.

5. Experimental Analysis

In this section, we evaluate the performance of our proposals for solving multiclass problems (as described in Section 4).

The experiments were performed over 15 different datasets (Table 3) taken from the UCI [27] and the LIBSVM[5] public repositories. All of the datasets are multivariate, multiclass, non-hierarchical a priori, and the criterion we followed to select them was to include datasets of different size and different numbers of classes. We have preprocessed the datasets removing the instances with missing values. Additionally, we have applied under-sampling to datasets 7, 9, and 13 (to cope with class imbalance).

DataSet	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Id	**Arrhytmia**	**Covtype**	**Dermatology**	**Flare**	**Forest**	**Glass**	**Letters**	**Pendigits**
NumInst	416	2100	358	1066	523	214	2600	7494
NumAtt	330	54	34	19	27	9	16	16
NumClass	7	7	6	6	4	6	26	10

DataSet	**9**	**10**	**11**	**12**	**13**	**14**	**15**
Id	**SatImage**	**Segmentation**	**Sports**	**TrafficLight**	**Usps**	**Vertebral**	**Zoo**
NumInst	1795	3000	8000	300	3000	310	101
NumAtt	36	18	13	10	256	6	16
NumClass	6	7	10	6	10	3	7

Table 3: Information about the datasets used in the experiments: number of instances, attributes, and classes.

5.1. Evaluating the use of Class Hierarchies to decompose Multiclass Problems

The purpose of this study is to analyse the performance of our method for solving multiclass problems by decomposing them using a class hierarchy that is derived from the confusion matrix.

The experiments we carried out follow the schema shown in Figure 3. First, a flat classifier is built and the class hierarchy is inferred as explained in Section 3. Next, the class hierarchy is compressed to reduce the number of flat classifiers to be learned, and, finally, the tree of classifiers is generated according to the transformed class hierarchy. In this first experiment, to infer the class hierarchies, we use the complete linkage distance in the agglomerative hierarchical clustering algorithm and a compression parameter of $\alpha = 0$.

In order to analyse the suitability of the class hierarchies generated by our method, we include two existing methods for generating hierarchies in the classes. The first method, which we denote as d_C, calculates the distance between the centroids of each class [15]. The second approach, which

[5] `https://www.csie.ntu.edu.tw/$\sim$cjlin/libsvmtools/datasets/`

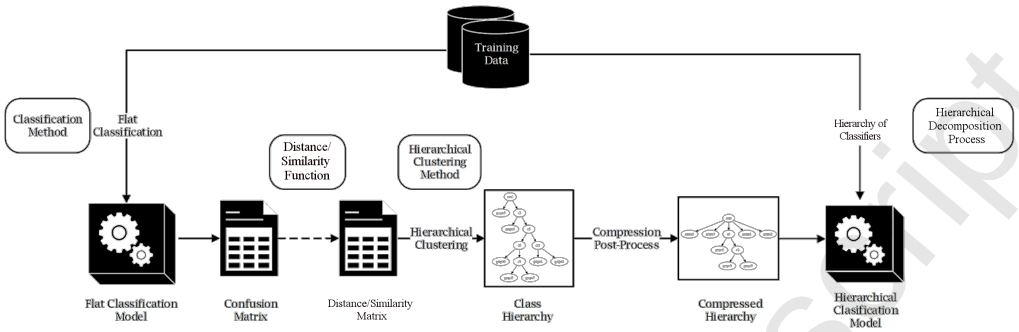


Figure 3: Decomposition schema of Multiclass Problems guided by Class Hierarchy inference.

we denote as d_E, computes similarities by applying the Euclidean distance over the normalized confusion matrix generated by the flat classifier[16]. Our proposals, which use a semi-metric function to compute the similarity matrix, are denoted by d_S when the confusion matrix is calculated using class estimations and d_{SP} when we use probability estimations (Section 3.2). In addition, to analyse whether the multiclass classification can be improved by using class hierarchies, we compare the results obtained with our top-down method (Section 4.2) with those obtained by the flat classifier, which was first trained to induce the class hierarchy.

In the experiments, we apply six classification techniques in an R [30] script by means of the libraries caret [31], rpart, e1071, and C50. Specifically, we use the following classification algorithms: a decision tree, J48; Naive Bayes, NB; a recursive partitioning tree, RPART; a neural network, NNET; a parallel random forest, RF; and a support vector machine, SVM. We adopt a 10-fold cross-validation for the complete process, and we use accuracy as the evaluation measure. In what follows, the letters S, SP, C, and E denote that the classifiers have been built from a class hierarchy that is inferred from our semi-metric distance using class estimates d_S and using probability estimates d_{SP}, the distance between centroids d_C, and the Euclidean distance d_E, respectively. Additionally, F stands for the flat classifier approach. Table 4 shows the results obtained. For each classification technique and dataset, the best result is underlined, and the best method using a hierarchy is highlighted in bold.

In general, the results show that, in terms of accuracy, the methods that use a class hierarchy outperform the flat classification for most datasets and

Options		Datasets														
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
J48	F	0,714	0,695	0,947	0,732	0,849	0,720	0,736	0,962	0,826	0,871	0,624	0,740	0,844	0,806	0,925
	E	**0,716**	0,708	0,947	0,738	0,874	0,685	0,720	0,956	0,811	0,957	0,620	0,747	0,848	0,810	**0,943**
	C	0,680	**0,711**	**0,955**	0,728	**0,877**	**0,739**	0,703	0,954	0,816	0,957	0,620	0,711	**0,861**	0,823	0,929
	S	0,709	0,707	0,949	**0,744**	0,866	0,714	0,720	**0,961**	0,817	0,956	**0,628**	0,724	0,852	0,810	0,933
	SP	0,706	0,704	0,938	0,733	0,866	0,704	**0,734**	0,961	**0,826**	**0,962**	0,612	**0,769**	0,843	**0,829**	0,943
NB	F	0,589	0,218	0,916	0,593	0,870	0,571	0,674	0,883	0,822	0,824	0,571	0,543	0,738	0,813	0,882
	E	0,589	0,165	**0,802**	0,548	0,852	0,559	0,581	0,805	0,750	0,871	0,511	0,452	0,632	**0,810**	0,599
	C	0,589	**0,240**	0,785	**0,590**	**0,860**	**0,599**	**0,707**	**0,864**	**0,795**	0,878	**0,538**	**0,482**	**0,694**	**0,810**	0,796
	S	0,589	0,146	0,771	0,551	0,852	0,583	0,535	0,793	0,749	0,871	0,520	0,371	0,684	**0,810**	0,645
	SP	**0,589**	0,190	0,760	0,521	0,854	0,597	0,525	0,818	0,758	**0,886**	0,535	0,405	0,668	0,803	**0,824**
NNET	F	0,587	0,283	0,972	0,761	0,876	0,703	0,362	0,463	0,843	0,605	0,229	0,659	0,672	**0,797**	0,941
	E	0,644	**0,508**	0,958	**0,759**	0,898	0,864	**0,770**	0,920	**0,876**	**0,937**	0,491	0,634	**0,935**	**0,797**	0,953
	C	**0,685**	0,454	0,966	0,751	0,906	0,694	0,720	**0,921**	0,870	0,913	0,508	0,560	0,931	0,768	0,954
	S	0,661	0,503	0,964	**0,759**	**0,917**	0,703	0,763	0,913	0,870	0,933	**0,517**	**0,707**	0,931	**0,797**	**0,963**
	SP	0,670	0,482	**0,972**	0,752	0,904	**0,933**	0,767	0,857	0,876	0,916	0,381	0,639	0,927	0,777	0,962
RPART	F	0,738	0,625	0,941	0,739	0,868	0,700	0,463	0,836	0,793	0,917	0,559	0,746	0,760	0,819	0,873
	E	0,764	0,666	0,924	0,729	**0,868**	0,708	0,645	0,901	0,803	**0,954**	0,589	0,707	0,814	**0,816**	0,866
	C	0,757	**0,669**	0,941	0,723	0,858	0,695	**0,646**	0,882	0,799	0,932	**0,596**	**0,736**	**0,840**	**0,816**	0,832
	S	**0,769**	0,667	0,927	**0,736**	0,866	0,680	0,636	**0,906**	0,798	0,948	0,584	0,724	0,830	**0,816**	0,866
	SP	0,766	0,664	**0,944**	0,726	0,866	**0,955**	0,623	0,885	**0,804**	0,948	0,582	0,727	0,821	**0,816**	**0,866**
RF	F	0,800	0,772	0,978	0,738	0,883	0,818	0,875	0,991	0,882	0,974	0,710	0,806	0,943	0,848	0,973
	E	**0,805**	**0,775**	0,978	0,750	0,891	0,816	0,849	0,991	0,876	0,976	0,708	0,807	0,900	0,845	**0,982**
	C	0,793	0,769	0,980	0,735	**0,889**	0,797	0,822	0,988	**0,877**	0,972	0,700	0,790	0,926	**0,852**	0,981
	S	0,800	0,774	**0,983**	**0,751**	0,889	0,831	**0,880**	0,991	0,875	0,975	0,707	0,803	**0,938**	0,845	0,973
	SP	0,752	0,755	0,972	0,734	**0,891**	**0,980**	0,837	**0,992**	0,874	**0,980**	**0,711**	**0,817**	0,937	0,845	0,981
SVM	F	**0,589**	0,684	0,970	0,756	0,895	0,714	0,834	0,995	0,870	0,939	0,626	0,571	0,969	0,848	0,951
	E	**0,589**	0,650	0,937	**0,749**	0,898	0,659	0,753	0,994	0,872	**0,958**	0,607	0,670	0,956	**0,826**	**0,972**
	C	**0,589**	**0,702**	0,925	0,735	0,896	**0,680**	**0,819**	0,994	0,871	0,934	0,612	0,571	**0,965**	**0,826**	0,962
	S	**0,589**	0,690	**0,945**	0,744	0,898	0,666	0,803	**0,996**	**0,873**	**0,958**	0,616	**0,677**	0,957	**0,826**	**0,972**
	SP	**0,589**	0,683	0,945	0,746	**0,898**	0,477	0,814	0,995	0,871	0,580	**0,621**	0,674	0,958	**0,826**	0,943

Table 4: Accuracy values obtained by the different classification techniques. The second column represents the methodology applied in building the hierarchy (Centroid= C, Euclidean= E, Semi-Metric = S, Semi-Metric with Probabilities = SP) with respect to flat classification (F). The distance that obtains the best result for each method is highlighted in bold, and the best result for method and dataset is underlined.

techniques (except for the NB followed by SVM whose flat classifiers are mainly more accurate than the classifiers using hierarchies). Hence, we can say that, in general, the use of class hierarchies for decomposing the original multiclass problems seems to be suitable for addressing multiclass problems. With regard to the way in which the class hierarchies are induced, on average, our semi-metric functions d_S and d_{SP} obtain the best results for almost all of the techniques except for the NB (where the distance between centroids d_C gives better accuracy values) and RPART (for which d_S and/or d_{SP} and d_C obtain similar results). When d_S and d_{SP} are compared, it can be observed that the approaches obtain similar results in general, although the exception is NB. For this method, d_{SP} usually obtains the worst performance, probably because of the bad probability estimations associated to NB models.

Next, we analyse how the size of the datasets (in terms of number of classes and instances) affects the performance of the methods. To do this, we have first grouped the datasets into three categories according to $NumClass$: small ($NumClass =< 6$, datasets 3,4,5,6,9,12,14); medium ($6 < NumClass < 10$, datasets 1,2,10,15); and large ($10 =< NumClass$, datasets 7,8,11,13). We observe that for the small-size group there are no big differences when using the flat, S, and C approaches. For medium-size and large datasets, C and S/SP approaches are the best for almost all of the learning techniques. When grouping the datasets according to the number of instances, it can be observed that for small-size datasets ($NumInst =< 550$, datasets 1,3,5,6,12,15), all of the methods perform similarly; for medium-size datasets ($550 < NumInst =< 2100$, datasets 2,4,9,14) and large datasets ($NumInst > 2100$, datasets 7,8,10,11,13), the methods that use a class hierarchy are better than the flat ones, with the methods that are based on the Euclidean and the semi-metric distances being the best ones.

5.2. Analysing the impact of the compression parameter

In the previous section, we used the value $\alpha = 0$ to define the threshold that is used to compress the class hierarchy. In this section, we analyse the impact of the parameter α on our methods.

In order to do that, we have conducted a first analysis to determine if there is some evidence that allows us to assume that by using this α value our methods will indeed perform better. This experiment was carried out on five datasets from Table 3, which were selected according to $NumClass$ $(3, 5, 6, 11, 13)$. We use our S method and the average linkage distance to generate the class hierarchies (given that we are using only five datasets, we

α	Classification Techniques					Average
	SVM	J48	PART	RF	NNET	
0	0,819	0,803	**0,783**	0,870	**0,791**	**0,813**
0.1	0,822	0,805	0,779	0,870	0,789	**0,813**
0.2	0,824	**0,806**	0,781	**0,871**	0,764	0,809
0.3	0,823	0,804	0,779	0,870	0,763	0,808
0.4	0,824	0,798	0,779	0,869	0,748	0,803
0.5	**0,824**	0,802	0,777	0,869	0,745	0,803

Table 5: Average of the accuracy of the S method per α and learning technique.

chose the average linkage since it is less sensitive to outlier instance values). We varied the α value from 0 to 0.5 in increments of 0.1. Table 5 shows the average of the accuracy for the five datasets and five learning techniques.

For this analysis, we performed a second experiment on all of the datasets in Table 3 using $\alpha = 0$ and $\alpha = 0.1$. Although the values 0 and 0.1 might seem to be similar (according to the average results in Table 5), the results in Table 6 show that, for almost all of the learning techniques, the S method with the compression parameter $\alpha = 0$ outperforms the results obtained at $\alpha = 0.1$ except for the NB method, which obtains a better average performance for $\alpha = 0.1$. Therefore, we will use $\alpha = 0$ in the next section.

5.3. Evaluating the effect of using probability estimators for problem decomposition

In this section, we experimentally evaluate the performance of the top-down&bottom-up method based on probability estimations as described in Section 4.3. We denote this method as PS when the d_S distance is used for generating the class hierarchy and as PSP when using the d_{SP} distance. To compare our two methods, in Table 7 we have also included the results for accuracy obtained by our top-down method S.

As can be observed, there are no big differences among the three methods. In general, the methods based on probabilities (PS and PSP) behave better for almost all of the learning techniques except for the SVM and NNET, for which the S method is more accurate for most datasets. We observe that the NB method behaves better when we use the PS and PSP method. This can be due to the fact that decomposing the problem into subproblems that have fewer classes improves the probability estimations of the NB algorithm. With regard to the construction of the class hierarchy, it seems that the distance derived from probability estimations does not improve the predictions. Hence, we can conclude that to induce the class hierarchy, it is advisable to

Options	α	Datasets							
		1	2	3	4	5	6	7	8
SVM	0	**0,589**	**0,690**	**0,945**	0,744	0,898	0,666	0,803	**0,996**
	0,1	**0,589**	0,677	0,942	**0,747**	**0,902**	**0,685**	**0,832**	**0,996**
J48	0	0,709	**0,707**	0,949	**0,744**	**0,866**	**0,714**	0,720	**0,961**
	0,1	**0,736**	0,701	**0,950**	0,734	0,864	0,702	**0,732**	**0,961**
RPART	0	**0,769**	**0,667**	0,927	0,736	**0,866**	**0,680**	**0,636**	**0,906**
	0,1	0,759	0,646	**0,944**	**0,740**	**0,866**	0,665	0,477	0,885
RF	0	**0,800**	**0,774**	**0,983**	**0,751**	0,889	**0,831**	**0,880**	**0,991**
	0,1	0,781	0,759	0,975	0,742	**0,891**	0,824	0,878	**0,991**
NNET	0	**0,661**	**0,503**	**0,964**	**0,759**	**0,917**	**0,703**	**0,763**	**0,913**
	0,1	0,592	0,327	0,955	0,753	0,891	0,685	0,301	0,817
NB	0	**0,589**	0,146	0,771	**0,551**	0,852	0,583	0,535	0,793
	0,1	**0,589**	**0,165**	**0,930**	0,524	**0,862**	**0,606**	**0,670**	**0,840**
Options	α	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**Average**
SVM	0	**0,873**	**0,958**	0,616	0,677	0,957	0,826	**0,972**	**0,814**
	0,1	0,872	0,700	**0,628**	**0,701**	**0,969**	**0,848**	**0,972**	0,804
J48	0	0,817	0,956	**0,628**	0,724	**0,852**	**0,810**	0,933	0,806
	0,1	**0,820**	**0,964**	0,618	**0,739**	0,850	0,806	**0,943**	**0,808**
RPART	0	**0,798**	**0,948**	**0,584**	0,724	**0,830**	0,816	0,866	**0,784**
	0,1	0,797	**0,948**	0,583	**0,727**	0,770	**0,819**	**0,893**	0,768
RF	0	0,875	0,975	0,707	0,803	0,938	0,845	**0,973**	**0,868**
	0,1	**0,880**	**0,981**	**0,710**	**0,814**	**0,953**	**0,848**	0,972	0,867
NNET	0	**0,870**	**0,933**	**0,517**	**0,707**	**0,931**	**0,797**	**0,963**	**0,793**
	0,1	0,851	0,923	0,364	0,673	0,739	0,787	0,944	0,707
NB	0	0,749	0,871	0,520	0,371	0,684	0,810	0,645	0,631
	0,1	**0,813**	**0,891**	**0,557**	**0,431**	**0,725**	**0,813**	**0,893**	**0,687**

Table 6: Comparison of the performance of the S method using the values 0 and 0.1 for the compression parameter α.

use a distance that is derived from class predictions and probability estimators to construct the hierarchy of classifiers.

6. Conclusions

In this paper, we have proposed an approach to improve accuracy in multiclass classification problems. The idea is that in situations where there exists a high number of classes, traditional methods find it difficult to correctly discern the new observations given the high number of possibilities. The proposal consists in building specialised classifiers for the classes that present the most common mistakes (i.e., to build a chain of specialised classifiers for simpler problems). Therefore, the method is based on the inference of the hierarchy of classes. From the confusion matrix obtained from training data, we derive a similarity matrix, then a hierarchical agglomerative clustering technique derives the hierarchy of classes. We have proposed a semi-metric to calculate the distances between classes given a confusion matrix. We compared two different methods to generate the confusion matrix:

Options		DataSets															Average
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
SVM	S	**0,589**	0,690	**0,945**	0,744	0,898	0,666	0,803	**0,996**	**0,873**	**0,958**	0,616	0,677	0,957	0,826	**0,972**	**0,814**
	PS	**0,589**	0,690	0,937	0,755	0,898	**0,700**	**0,843**	0,995	0,868	0,706	0,615	0,680	**0,967**	**0,829**	0,952	0,802
	PSP	**0,589**	**0,691**	0,934	**0,757**	**0,902**	0,684	0,840	0,995	0,871	0,716	**0,623**	**0,710**	0,965	0,823	0,945	0,803
J48	S	0,709	**0,707**	0,949	0,744	**0,866**	**0,714**	0,720	**0,961**	0,817	0,956	**0,628**	0,724	**0,852**	0,810	0,933	0,806
	PS	0,724	0,703	**0,950**	**0,749**	**0,866**	0,703	0,730	0,959	**0,827**	0,962	0,613	0,739	0,836	0,813	0,943	0,808
	PSP	**0,726**	0,691	0,947	0,741	0,864	**0,714**	**0,736**	**0,961**	0,821	**0,963**	0,611	**0,745**	0,843	**0,816**	**0,962**	**0,809**
RPART	S	**0,769**	**0,667**	0,927	**0,736**	**0,866**	0,680	**0,636**	**0,906**	0,798	0,948	**0,584**	**0,724**	**0,830**	**0,816**	**0,866**	0,784
	PS	0,767	0,662	**0,930**	0,721	**0,866**	**0,948**	0,536	0,886	**0,806**	**0,955**	0,582	**0,724**	0,826	**0,816**	**0,866**	**0,793**
	PSP	0,766	0,665	**0,930**	0,727	**0,866**	0,665	0,622	0,886	0,804	**0,955**	0,582	**0,724**	0,821	**0,816**	**0,866**	0,780
RF	S	**0,800**	**0,774**	**0,983**	**0,751**	0,889	0,831	**0,880**	**0,991**	0,875	0,975	0,707	**0,803**	0,938	**0,845**	**0,973**	0,868
	PS	0,783	0,760	0,978	0,743	**0,895**	**0,981**	0,878	0,988	**0,880**	0,980	**0,712**	0,787	**0,953**	0,835	0,972	**0,875**
	PSP	0,762	0,752	0,978	0,736	0,893	0,816	0,862	0,990	0,876	**0,981**	**0,712**	0,790	0,945	0,839	0,962	0,860
NNET	S	0,661	**0,503**	0,964	**0,759**	**0,917**	0,703	0,763	0,913	**0,870**	0,933	0,517	**0,707**	**0,931**	0,797	**0,963**	**0,793**
	PS	0,610	0,463	**0,969**	0,757	0,912	**0,923**	0,578	0,928	0,859	0,933	**0,527**	0,677	0,920	**0,803**	**0,963**	0,788
	PSP	**0,671**	0,497	0,966	0,758	0,895	0,712	**0,772**	**0,931**	0,869	**0,937**	0,519	0,657	**0,931**	0,794	0,962	0,791
NB	S	**0,589**	0,146	0,771	**0,551**	0,852	0,583	0,535	0,793	0,749	0,871	0,520	0,371	**0,684**	**0,810**	0,645	0,631
	PS	**0,589**	0,151	**0,802**	0,532	**0,854**	0,580	**0,595**	0,805	0,758	0,870	0,522	0,389	0,660	**0,810**	**0,835**	**0,650**
	PSP	**0,589**	**0,187**	0,765	0,521	**0,854**	**0,597**	0,547	**0,820**	**0,758**	**0,882**	**0,537**	**0,405**	0,660	**0,810**	0,802	0,649

Table 7: Comparing the methods based on class estimations and probability estimation to solve multiclass problems using a hierarchy of classifiers.

one based on the crisp class predictions, and one based on the class probability estimations. We also introduced a modification in the original algorithm that probabilistically combines the classes of the leaves of the hierarchy of classifiers for making the predictions. Finally, we introduced a method to compress hierarchies, which allows us to better represent the hierarchical structure and also helps to reduce the complexity of hierarchical classification since it reduces the number of local classifiers. The proposed method compresses the hierarchy using a minimum distance threshold.

Experiments with several multiclass datasets illustrate the validity of our proposal. We have shown that the new technique is able to improve accuracy with respect to the basic flat approach. In the experiments, we also included other proposals to build the hierarchy of classifiers. The new method based on the semi-metric distance obtained a better performance for the majority of datasets. Finally, we evaluated the behaviour of the improvements proposed for the original algorithm.

As future work, we propose to better analyse the relation between number of classes and performance obtained by the chain of classifiers approach. We are also interested in studying the effect of class balance on the performance of the method. Finally, we plan to further study methods that are based on the combination of the local classifiers of the hierarchy.

[1] L. Breiman, L. Breiman, Bagging predictors, in: Machine Learning, 1996, pp. 123–140.

[2] Y. Freund, R. E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, in: European conference on computational learning theory, Springer, 1995, pp. 23–37.

[3] T. G. Dietterich, Ensemble methods in machine learning, in: Multiple Classifier Systems, LBCS-1857, Springer, 2000, pp. 1–15.

[4] D. H. Wolpert, Stacked generalization, Neural Networks 5 (1992) 241–259.

[5] J. Gama, P. Brazdil, Cascade generalization, Machine Learning 41 (3) (2000) 315–343.

[6] J. A. Olvera-López, J. A. Carrasco-Ochoa, J. F. Martínez-Trinidad, J. Kittler, A review of instance selection methods, Artificial Intelligence Review 34 (2) (2010) 133–143.

[7] H. Liu, L. Yu, Toward integrating feature selection algorithms for classification and clustering, IEEE Transactions on knowledge and data engineering 17 (4) (2005) 491–502.

[8] N. Chawla, K. Bowyer, L. Hall, W. Kegelmeyer, Smote: Synthetic minority over-sampling technique, Journal of Artificial Intelligence Research 16 (2002) 321–357.

[9] I. Tomek, Two modifications of cnn, IEEE Transactions on Systems Man and Communications 6 (1976) 769–772.

[10] V. López, A. Fernández, S. García, V. Palade, F. Herrera, An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics, Information Sciences 250 (2013) 113–141.

[11] A. Niculescu-Mizil, R. Caruana, Predicting good probabilities with supervised learning, in: Proceedings of the 22nd international conference on Machine learning, ACM, 2005, pp. 625–632.

[12] B. Zadrozny, C. Elkan, Transforming classifier scores into accurate multiclass probability estimates, in: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2002, pp. 694–699.

[13] A. Bella, C. Ferri, J. Hernández-Orallo, M. Ramírez-Quintana, On the effect of calibration in classifier combination, Applied Intelligence.

[14] B. Zupan, M. Bohanec, J. Demšar, I. Bratko, Learning by discovering concept hierarchies, Artificial Intelligence 109 (1) (1999) 211–242.

[15] T. Li, S. Zhu, M. Ogihara, Hierarchical document classification using automatically generated hierarchy, Intelligent Information Systems 29 (2007) 211–230.

[16] S. Godbole, S. Sarawagi, S. Chakrabarti, Scaling multi-class support vector machines using inter-class confusion, in: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2002, pp. 513–518.

[17] M. Steinbach, G. Karypis, V. Kumar, A comparison of document clustering techniques, in: In KDD Workshop on Text Mining, 2000.

[18] P. Cimiano, A. Hotho, S. Staab, Learning concept hierarchies from text corpora using formal concept analysis, J. Artif. Int. Res. 24 (1) (2005) 305–339.

[19] P. Cimiano, A. Pivk, L. Schmidt-Thieme, S. Staab, Learning taxonomic relations from heterogeneous sources of evidence, Ontology Learning from Text: Methods, evaluation and applications 123 (2005) 59–73.

[20] F. Benites, E. Sapozhnikova, Learning different concept hierarchies and the relations between them from classified data, Intel. Data Analysis for Real-Life Applications: Theory and Practice (2012) 18–34.

[21] J. G. Jung J.J., Yu YH., Collaborative web browsing based on ontology learning from bookmarks, in: ICCS 2004 International Conference on Computer Science, Vol. 3038 of Lecture Notes in Computer Science, 2004, pp. 513–520.

[22] S.-L. Chuang, L.-F. Chien, A practical web-based approach to generating topic hierarchy for text segments, in: Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management, CIKM '04, ACM, 2004, pp. 127–136.

[23] V. Schickel-Zuber, B. Faltings, Using hierarchical clustering for learning the ontologies used in recommendation systems, in: 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'2007, 2007, pp. 599–608.

[24] G. A. Carpenter, S. Martens, O. J. Ogas, Self-organizing information fusion and hierarchical knowledge discovery: a new framework using artmap neural networks, Neural Networks 18 (3) (2005) 287–295.

[25] F. Brucker, F. Benites, E. Sapozhnikova, Multi-label classification and extracting predicted class hierarchies, Pattern Recognition 44 (3) (2011) 724–738.

[26] F. Galvin, S. Shore, Distance functions and topologies, The American Mathematical Monthly 98 (7) (1991) 620–623.

[27] K. Bache, M. Lichman, UCI machine learning repository (2013).
URL http://archive.ics.uci.edu/ml

[28] J. Silla, N. Carlos, A. A. Freitas, A survey of hierarchical classification across different application domains, Data Mining and Knowledge Discovery 22 (1-2) (2011) 31–72.

[29] R. Dugad, N. Ahuja, Unsupervised multidimensional hierarchical clustering, in: Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on, Vol. 5, IEEE, 1998, pp. 2761–2764.

[30] R Core Team, R: A Language and Environment for Statistical Computing, R Foundation for Statistical Computing, Vienna, Austria (2015).
URL http://www.R-project.org/

[31] M. K. C. from Jed Wing, S. Weston, A. Williams, C. Keefer, A. Engelhardt, T. Cooper, Z. Mayer, the R Core Team, caret: Classification and Regression Training, r package version 6.0-30 (2014).
URL http://CRAN.R-project.org/package=caret