

Improving Local Per Level Hierarchical Classification

Bruno C. Paes¹, Alexandre Plastino¹, Alex A. Freitas²

¹ Universidade Federal Fluminense, Brazil
{bpaes, plastino}@ic.uff.br

² University of Kent, United Kingdom
a.a.freitas@kent.ac.uk

Abstract.

In the domain of many relevant classification problems, classes are organized in hierarchies, representing specialization relationships between them. These are the so-called hierarchical classification problems. Methods based on different approaches have been used to solve them, trying to achieve better predictive performance. In this work, we propose two local per level hierarchical classifiers, which contain distinct strategies to solve inconsistent predictions, common to the local per level approach. We have compared the proposed methods with traditional strategies from different paradigms. The computational experiments, conducted over 18 hierarchical classification data sets, showed that the proposed ideas were able to reach competitive and robust results in terms of prediction accuracy.

Categories and Subject Descriptors: H.2.8 [Database Applications]: Data Mining

Keywords: classification, data mining, hierarchical classification

1. INTRODUCTION

Classification is one of the most explored tasks in the data mining area. Its main objective is to predict the unknown classes of new instances, based on other known-class instances observed in a previous training phase. In the literature, most classification problems involve classes with no specified parent-child relationships between them. These problems are known as flat classification problems. By contrast, there are several problems involving classes organized in a hierarchical structure, defined by parent-child relationships between classes, where a parent class is more generic than its child classes. These problems are known as hierarchical classification problems.

Examples of hierarchical classification problems can be found in different application areas. In bioinformatics, there are several works related to protein function hierarchical classification [Costa et al. 2007] [Costa et al. 2008] [Holden and Freitas 2008] [Secker et al. 2007]. In text mining, documents can be categorized within hierarchical structures of subjects [Dumais and Chen 2000]. In image recognition, objects can be classified into hierarchical categories of geometric shapes [Barutcuoglu and DeCoro 2006].

In hierarchical classification problems, two main types of class structures are used: a tree-based class hierarchy, where each class has at most one parent class, and a direct acyclic graph-based class hierarchy, where a child class can have multiple parent classes. Hierarchical classification problems can be further categorized into mandatory leaf node prediction problems, where every instance must be assigned a leaf class node in the hierarchy, or optional leaf node prediction problems, where the most specific class assigned to an instance can be at any level of the class hierarchy [Freitas and de Carvalho 2007]. Hierarchical classification problems can also be grouped according to the number

The development of this work was supported by CNPq and FAPERJ research grants.

Copyright©2012 Permission to copy without fee all or part of the material printed in JIDM is granted provided that the copies are not made or distributed for commercial advantage, and that notice is given that copying is by permission of the Sociedade Brasileira de Computação.

of classes that each instance can be associated to. When instances have at most one class per hierarchy level, we have a single label problem. And when each instance can be associated to a set of classes per hierarchy level, we have a multilabel problem.

Another important issue in hierarchical classification is the trade-off between reliability and usability of predictions at different levels of class generality [Freitas and de Carvalho 2007]. In general, the closer to the root level a class prediction is, the more reliable it is, since there are fewer classes and more instances per class (simplifying the classification task) at shallower levels than at deeper levels. On the other hand, in general, the further from the root a class prediction is, the more useful it is to the user, since deeper classes carry more specific information than shallower classes.

In this work, we deal with tree-based hierarchical problems, where each instance is associated to a single class per level of the hierarchy (single label problem) and where the prediction is always made at the most specific level of the class hierarchy (mandatory leaf-node prediction).

Algorithms for hierarchical classification can be divided into different categories, depending on whether they build a global model for the entire class hierarchy or a set of local models, where each model predicts a subset of classes [Freitas and de Carvalho 2007]. In this work we focus on the local classifier per level approach, where a flat classifier is trained, independently, for each level of the hierarchy. The major drawback of this approach is that classifiers at different levels can (and often do) make inconsistent class predictions. E.g., a classifier at the first level could predict class 1 while a classifier at the second level could predict class 2.2, which is inconsistent because if an instance had class 2.2 it would necessarily have class 2, given the parent-child relationship between those classes.

In spite of requiring the training of a reduced number of classifiers, one per hierarchical level, the local classifier per level approach has not been properly addressed in the literature yet. Probably, this is due to the inconsistent predictions across class levels. In this work, we propose two local per level hierarchical classification algorithms, with distinct strategies to solve inconsistent predictions. The proposed strategies are compared to other types of hierarchical algorithms across 18 datasets. The obtained results show the proposed strategies have a good and competitive predictive performance.

This article is organized as follows. Section 2 describes different approaches for hierarchical classification and reviews related work. In Section 3, two new algorithms for local per level hierarchical classification are proposed, which are the main contribution of this work. Section 4 describes the algorithms to which the proposed strategies are compared. Section 5 reports the results and analysis of computational experiments to evaluate the proposed algorithms. In Section 6, the conclusions of this work are highlighted and some future work is proposed.

2. HIERARCHICAL CLASSIFICATION APPROACHES AND RELATED WORK

In this section, we describe the hierarchical classification approaches according to the categorization described in [Silla and Freitas 2011]. This categorization criterion is related to how the hierarchical structure is explored.

2.1 Flat Classification Approach

The flat classification approach simplifies the hierarchical classification problem by transforming it into a flat classification problem. Basically, a classifier is trained to deal only with leaf classes. In Figure 1(a), the leaf nodes, which are the classes considered by the flat classification algorithm, are highlighted. This approach provides an indirect solution to the hierarchical classification problem, since, from a predicted leaf class, all its ancestor classes are also assigned to the instance.

In [Burred and Lerch 2003], the authors refer to this approach as "direct approach", which is considered as a global classifier in [Xiao et al. 2007]. In [Barbedo and Lopes 2007], authors call this

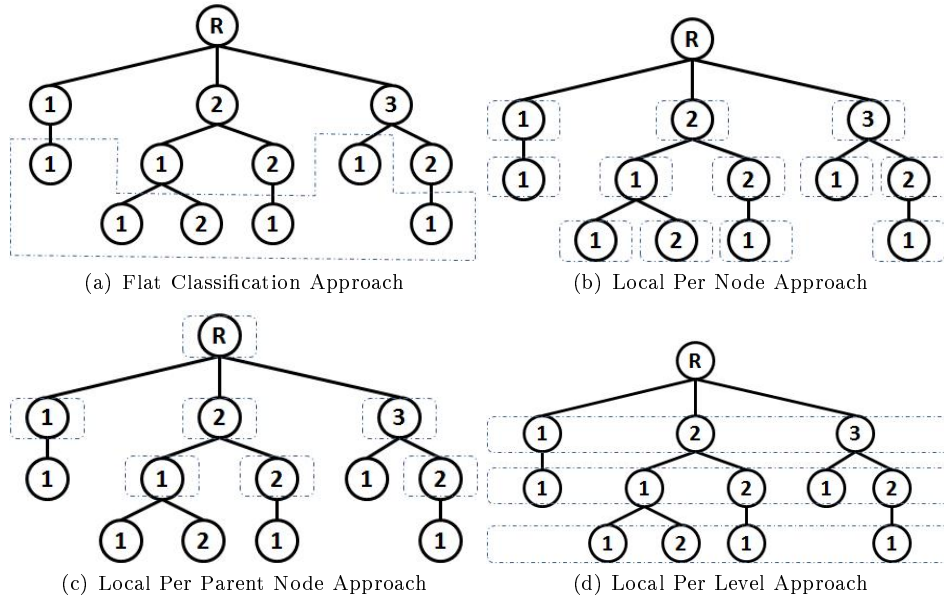


Fig. 1. Four types of hierarchical classification approaches

strategy as a bottom-up approach, since the prediction is given initially in leaf classes and then the ancestor classes can be inferred.

The major advantage of this approach is its simplicity, since it uses a single flat classifier to predict a set of classes in the hierarchy. However, this approach has a serious disadvantage: it builds a classifier without exploiting the parent-child relationships in the hierarchy.

2.2 Local Classification Approach

A local classifier explores the hierarchy of classes through a local perspective. In [Silla and Freitas 2011], local classifiers are categorized based on how they explore this local information: local per node approach, local per parent node approach and local per level approach.

In local classifiers, class predictions in the testing phase are typically executed in a top-down fashion, that is, for each new instance to be classified, this approach predicts classes from the highest to the lowest level of the hierarchy, considering, in each level, the child classes of the previous predicted class. Hence, the local approach is often referred to as a top-down method in the literature. However, it is worth noting that this top-down procedure is adopted only during the test phase, to avoid inconsistent predictions, and not during the training of the local classifiers.

2.2.1 Local Classifier Per Node Approach. The local per node approach consists of training one binary classifier for each node of the class hierarchy (except the root node). After training, the system consists of a hierarchy of flat classifiers. Figure 1(b) illustrates this kind of classifier. The dashed squares represent binary classifiers. In essence, each binary classifier predicts whether or not an instance belongs to its associated class. Different criteria to define the set of positive and negative instances to build each binary classifier are discussed in [Ceci and Malerba 2007; Eisner et al. 2005; Fagni and Sebastiani 2007].

In this approach each classifier solves just a modular binary classification problem, but, depending on the hierarchy size, a large number of classifiers must be trained. Another problem is related to multiple classifications obtained by the set of binary classifiers, allowing one instance to be associated

to different classes in the same hierarchy level (horizontal inconsistency) or classes in different branches of the hierarchy (vertical inconsistency). For example, in Figure 1(b), after the classification of an instance, the binary classifiers achieved positive predictions for the classes 2, 2.1, 2.2 and 3.2.1. Thus, at level 2, the predicted classes 2.1 and 2.2 represent a horizontal inconsistency and, in addition, the predict class 3.2.1 (level 3) is not consistent with the predicted classes in level 2 (vertical inconsistency).

Therefore, the local per node approach must be coupled with a method to avoid vertical inconsistency (in single and multilabel problems) and horizontal inconsistency (in single label problems). We now review some local per node existing methods.

The method presented in [Wu et al. 2005], called Binarized Structured Label Learning, is applied to a multilabel problem and follows the idea of training a binary classifier to each class node. The classification of an instance is done through each branch of the hierarchy and, to avoid vertical inconsistencies, the process is stopped in a branch when a negative prediction is achieved.

In [Dumais and Chen 2000], two local per node methods, which prevent vertical inconsistencies in multilabel problems, are proposed. The first method evaluates each branch of the hierarchy in a top-down fashion and, while the estimated probability of the current predicted class is greater than a threshold, the class is assigned to the instance and the system proceeds to the next level of the hierarchy. The second method is similar to the first one, but it assigns a class to the instance when the product of the probability of this class and the probability of its child class is greater than a threshold. These methods were applied in a multilabel hierarchical classification problem, of which elements were web pages.

In [Barutcuoglu and DeCoro 2006] and [Valentini 2009], other methods for multilabel problems were proposed. The main features of these methods are: in the first work, the training and use of a Bayesian network and, in the second, the vertical treatment of inconsistencies in a bottom-up way.

2.2.2 Local Classifier Per Parent Node Approach. This approach consists of training a traditional flat classifier for each parent class of the class hierarchy. In essence, for each parent class, it is build a classifier that discriminates among their child classes. Figure 1(c) illustrates the local classifier per parent node approach, where the dashed squares represent classifiers predicting their child classes.

This approach is also often referred to in the literature as a top-down approach. However, as explained earlier, this term is better used to refer to strategies to avoid inconsistent predictions in the test phase, since the training phase is not "top-down".

In [Koller and Sahami 1997], the first local classifier per parent node method was proposed, adopting the top-down strategy in the test phase.

In [Secker et al. 2007], the authors proposed a local classifier per parent method, called Selective Classifier, also using the top-down strategy in the test phase. For each parent node of the hierarchy, different classifiers are built using a subset of the training set and then evaluated on the other part of the training set (the validation set). The classifier chosen for each parent class node is the one with the highest classification accuracy on the validation set.

In [Holden and Freitas 2008], it is proposed an improvement over the Selective Classifier. This proposal also builds different classifiers for each parent node, however, the evaluation is done globally, looking for the group of classifiers that presents the best performance. Additionally, in [Secker et al. 2010], it was proposed the execution of an attribute selection procedure before building the classifiers associated to each parent node.

2.2.3 Local Classifier Per Level Approach. This approach consists of training a flat classifier for each level of the class hierarchy. Figure 1(d) illustrates this approach. For each level of the hierarchy, there is a classifier responsible for discriminating among the classes at that level.

This approach has the advantage of naturally avoiding horizontal class prediction inconsistencies (since each classifier predicts exactly one class at its level). However, it has a major drawback: the possibility of vertical inconsistencies. For example, considering the hierarchy in Figure 1(d), it is possible that class 3 is predicted at level 1, class 2.1 predicted at level 2, and class 3.2.1 predicted at level 3. In this case, the class predicted at the second level is not consistent with the classes predicted at the first and third levels.

In [Clare and King 2003; Costa et al. 2007], this approach was used as a baseline in the evaluation of other hierarchical classifiers by comparing the outputs obtained at each level, without dealing with inconsistencies.

In order to make this approach useful, it is necessary to complement it with a post-processing method that tries to correct inconsistent predictions. The main contribution of this work is the proposal of two local classifier per level methods coupled with two distinct post-processing strategies. It is worth noting that, to the best of our knowledge, there is no local classifier per level method in the literature that has a specific procedure to handle inconsistency of class predictions.

2.3 Global Classification Approach

Global classifiers basically build a single classification model taking into account the class hierarchy as a whole during a single run of the classification method. This approach is often referred to as the big-bang approach [Costa et al. 2007; Freitas and de Carvalho 2007].

This approach has the ability to take into account all the classes in the hierarchy in a single training process, unlike the local approaches which divide the training phase in modular processes, considering parts of the hierarchy. Consequently, the global approach has the disadvantage of implementing a non-modular process, resulting in a more complex hierarchical classification algorithm.

In [Rocchio 1971], the author proposed a global classifier where a new instance is assigned to the nearest class, by evaluating a distance function between the instance and each class in the hierarchy. In [Labrou and Finin 1999], it is presented a multilabel system to classify web pages. In essence, during the test phase, each page is compared, through a similarity measure, with each category in the hierarchy. If the similarity value is above a threshold, the web page is associated with the category.

Another way of building global classifiers consists of modifying existing traditional flat classifiers to make them able to consider the hierarchical structure of classes. In [Clare and King 2003], it was proposed a new version of the decision tree induction algorithm C4.5 to consider the class hierarchy (HC4.5). In [Silla and Freitas 2009], the authors changed the traditional Naive Bayes algorithm in order to consider the relationships between classes in the estimation of class probabilities.

3. NEW LOCAL CLASSIFIER PER LEVEL HIERARCHICAL CLASSIFICATION METHODS

The hierarchical classifiers proposed in this article are based on the concepts of the local classifier per level approach, described in Section 2. Generally, in this approach, a flat classifier is trained and run for each level of the class hierarchy.

Figure 2(a) illustrates a class hierarchy with four levels where, for each level, a flat classifier was trained (C_1, C_2, C_3, C_4). The different classifiers, although applied at distinct levels, can be trained using the same classification algorithm. For example, the four classifiers C_1, C_2, C_3 and C_4 may be decision trees induced by the algorithm C4.5.

The major drawback of this approach, as described earlier, is the occurrence of inconsistent predictions. Figure 2(b) illustrates an inconsistent classification, in which the shaded nodes represent the outputs of the flat classifiers applied at each level. The classification at level 3 is not consistent with the classification at the level 2, since class 3.2.1 is not a descendant of class 2.1.

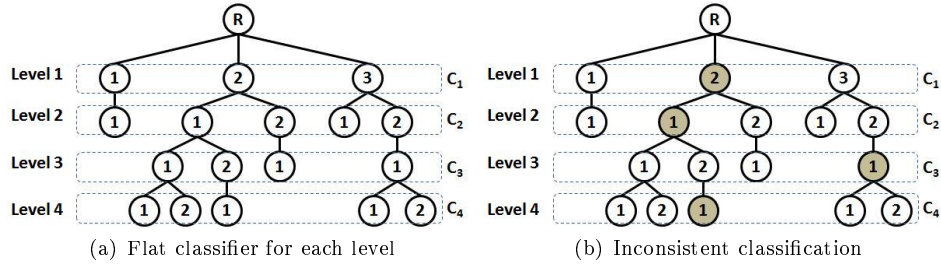


Fig. 2. Classifier Per Level Approach

In order to deal with this problem, we propose strategies that decompose the hierarchical classification problem into a set of flat classification problems (based on the local classifier per level approach) and apply a post-processing inconsistency treatment. In the next subsections, we will describe the proposed hierarchical classifiers.

3.1 Sum of Votes Hierarchical Classifier

In the first proposed classifier, called Sum of Votes (SV), the class-inconsistency treatment prioritizes the branch or path (from the root to a leaf class) which contains the greatest number of predictions – or votes – from the flat classifier’s output.

Initially, the strategy identifies the branches that contain at least one vote, i.e., at least one class predicted by one of the flat classifiers. Considering these branches, different scenarios can occur, which will be discussed next.

Scenario 1: There is a path P with a higher number of votes than the other paths and the class predicted at the lowest level in P is a leaf class. In this case, for each level N where the predicted class does not belong to the path P , the treatment of inconsistency replaces the predicted class at level N by the class in the path P belonging to the level N .

Figure 3 shows an example of this scenario. The predicted classes for each level are (Figure 3(a)): 2, 2.1, 3.2.1, and 2.1.2.1. The path with the highest number of votes contains the classes 2, 2.1, and 2.1.2.1, the last one being a leaf class. The predicted class in level 3 – 3.2.1 – does not belong to this path and the inconsistency elimination procedure transforms this class into the class 2.1.2 (Figure 3(b)).

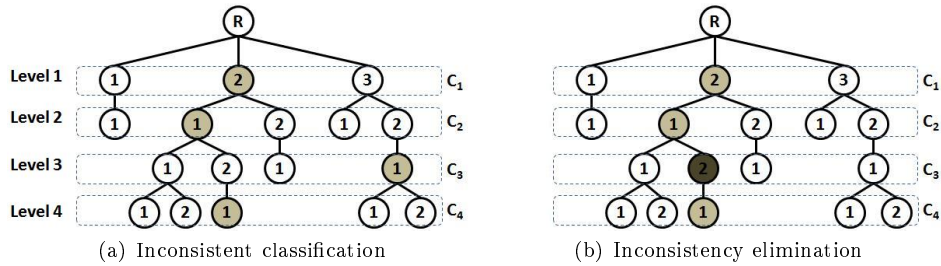


Fig. 3. Illustration of Scenario 1 for Sum of Votes Strategy

Scenario 2: There is a path P with a higher number of votes than the other paths, as the Scenario 1, but without a leaf predicted class in P . In this case, the class-inconsistency treatment consists of two steps: (1) for each level N higher than the lowest level that contains a predicted class in P , if the class in N does not belong to P , the predicted class in N is replaced by the class in the path P belonging to level N ; (2) for the levels below the lowest level that contains a predicted class in P , the classes will

be determined using the local per parent top-down strategy, i.e., for the class node with the lowest level in P , a flat classifier is built taking into account only its child classes. Then this classifier chooses one of the child classes, including it in P . If this chosen class is a leaf class, the treatment is finished. Otherwise the process is repeated for the next level, and so on, until a leaf class is included in P .

Figure 4 illustrates this scenario. The predicted classes for each level are (Figure 4(a)): 3, 1.1, 3.2.1, and 2.1.2.1. The path with the highest number of votes contains the classes 3 and 3.2.1, where the level 3 is the lowest level with predicted class in P . The strategy, in a first step, verifies that there is a predicted class in level 2 (1.1) which does not belong to this path and eliminates the inconsistency by replacing the predicted class in this level by the class 3.2 (Figure 4(b)). In the second step, the strategy performs the training of a flat classifier assigned to the non-leaf class 3.2.1, which chooses one class between its child classes 3.2.1.1 and 3.2.1.2. The output class 3.2.1.2 is included in the path P , and, since it is a leaf class, the inconsistency treatment is concluded (Figure 4(b)).

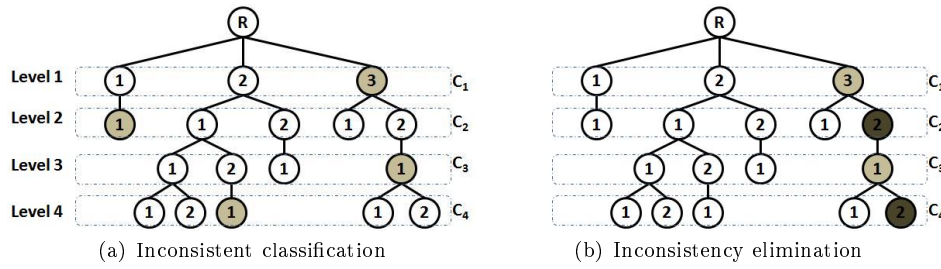


Fig. 4. Illustration of Scenario 2 for Sum of Votes Strategy

Scenario 3: There are multiple paths with the highest number of votes. In this case, the class-inconsistency treatment initially chooses, among these tied paths, the path P that contains the predicted class in the level nearest to the root of the hierarchy. After choosing the path P , if the predicted class of the lowest level in P is a leaf class, the procedure of Scenario 1 is used. Otherwise, the procedure of Scenario 2 is used.

Figure 5 presents an example of this third scenario. The predicted classes for each level are (Figure 5(a)): 3, 2.1, 2.1.2, and 3.2.1.2. There are two paths with two votes: one contains the predicted classes 3 and 3.2.1.2, and the other, the predicted classes 2.1 and 2.1.2. The system initially chooses the path that contains the predicted class 3, since this class is the nearest class to the root. After choosing this path, the example follows the Scenario 1, because a leaf class belongs to this path. Hence, the class-inconsistency elimination procedure replaces the predicted classes 2.1 and 2.1.2, which do not belong to the chosen path, by the classes 3.2 and 3.2.1 (Figure 5(b)).

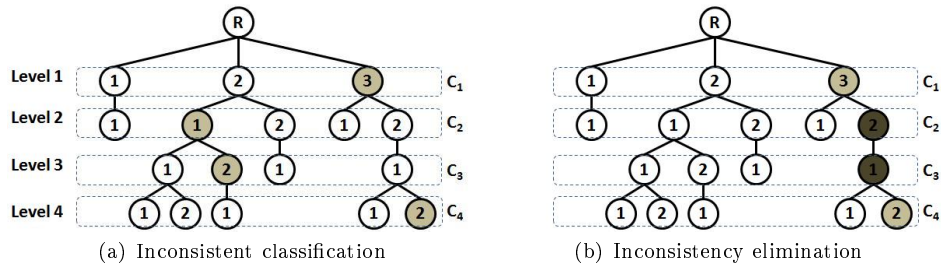


Fig. 5. Illustration of Scenario 3 for Sum of Votes Strategy

3.2 Sum of Weighted Votes Hierarchical Classifier

The second proposed hierarchical classifier, called Sum of Weighted Votes (SWV), consists of weighting the votes considered in the previous strategy with the probabilities estimated by the flat classifiers when predicting the classes for each level. Similarly to the SV strategy, this method identifies the branches that contain at least one vote and the branch that contains the largest sum of weighted votes will be adopted to eliminate possible inconsistencies. For this strategy, four different scenarios will be considered. The first three scenarios are very similar to the ones defined for the SV strategy and their description will be abbreviated whenever possible.

Scenario 1: There is a path P with a higher sum of weighted votes than the other paths and the predicted class of the lowest level for P is a leaf class. After identifying this path P , the treatment of class inconsistency follows the same procedure presented in Scenario 1 of the SV strategy.

Scenario 2: There is a path P with a higher sum of weighted votes than the other paths, as in Scenario 1, but without a leaf predicted class in P . After identifying this path P , the inconsistency treatment follows the same procedure presented in Scenario 2 of the SV strategy.

Scenario 3: There are multiple paths with the highest sum of weighted votes. In this case, the class-inconsistency treatment initially chooses, among these tied paths, the path P that contains the predicted class in the level nearest to the root of the hierarchy. After choosing the path P , if the predicted class of the lowest level in P is a leaf class, the procedure of Scenario 1 is used. Otherwise, the procedure of Scenario 2 is used.

In addition to these three scenarios, a fourth one was motivated by the case illustrated in Figure 6(a), which presents the class predictions at each level of the hierarchy for a given instance. Consider that, in the first two levels, the classes 1 and 1.1 are the real classes of the instance. Notice that these two predicted classes compose a consistent class path, including a leaf node. However, since the local classifier per level approach always predicts classes at all levels, the classes 3.2.1 and 3.2.1.1, obtained for the other two levels, are also taken into account to choose the path with the highest sum of probabilities. We observe that the path formed by the classes 3.2.1 and 3.2.1.1 would be the chosen path by the strategy, according to what was defined in Scenario 1. Therefore, considering that the instance belongs to classes 1 and 1.1, the strategy would force a classification error. Hence, it is necessary to define a fourth scenario, described below, to consider the case in which a predicted leaf class does not belong to the lowest level of the hierarchy.

Scenario 4: There is a path P with a predicted class in level 1, with a predicted leaf class in level i and with predicted classes, consistent with the two previous classes, at each level between level 1 and level i . In this case, the strategy chooses the classes in path P , even if there are other paths with a higher sum of weighted votes, and does not assign classes at the levels below the leaf class, if any. The path P is chosen and the process is finalized. This fourth scenario will always occur simultaneously with one of the other three. Then it is important to define that the fourth scenario has precedence over the others and will be applied whenever it occurs.

Figure 6 illustrates the fourth scenario. The predicted classes for each level are (Figure 6(a)): 1, 1.1, 3.2.1, and 3.2.1.1. The class-inconsistency elimination procedure checks the existence of a path that contains the predicted classes 1 and 1.1, where class 1 is at the first level and its child class 1.1 is a leaf class. Therefore, even though there is another path with a higher sum of weighted votes, the strategy decides to assign to the new instance the predicted classes in this path (Figure 6(b)).

4. TRADITIONAL HIERARCHICAL CLASSIFIERS

In order to evaluate the performance of the proposed classifiers, we implemented two hierarchical classifiers based on traditional hierarchical approaches described in Section 2: flat classification and local classifier per parent node approaches.

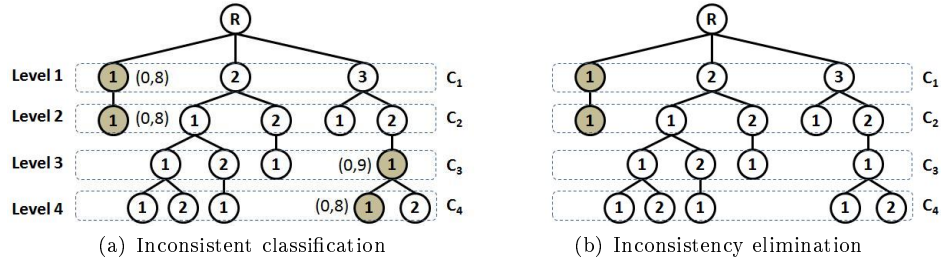


Fig. 6. Illustration of Scenario 4 in Sum of Weighted Votes Strategy

4.1 Flat on Leaves Classifier

The implemented hierarchical classifier Flat on Leaves (FL) is based on the concepts of the flat classification approach. A single flat classifier is trained taking into account only the leaf classes of the hierarchy. Therefore, the classifier outputs a leaf class and, from this predicted class, all its ancestor classes are also assigned to the instance.

Figure 7(a) illustrates the implemented FL classifier. All leaf classes – the lightly shaded ones – are included in the training of the classifier. The figure shows that, for a given instance, the classifier predicts the leaf class 2.1.1.2 in the fourth hierarchy level. From this prediction, it is possible to infer the classes from upper levels. In this example, the FL outputs the class 2.1.1 for level 3, the class 2.1 for level 2 and, finally, the class 2 for level 1.

4.2 Per Parent Top-Down Classifier

The implemented hierarchical classifier Per Parent Top-Down (PPTD) is based on the concepts of the local classifier per parent node approach, in which, a flat classifier is assigned for each non-leaf class (internal node) of the hierarchy. Each classifier (associated with an internal class node) is essentially a flat classifier, which distinguishes between its child classes. In this way, PPTD builds a hierarchy of flat classifiers.

A new instance is classified in a top-down way. Initially, the instance is evaluated by the root node classifier, which assigns one of its child classes to the instance. Then, in the next level, the classifier associated with the class node predicted at the previous level will assign one of its child classes to the instance, and so on. This process is recursively repeated until a leaf class is reached.

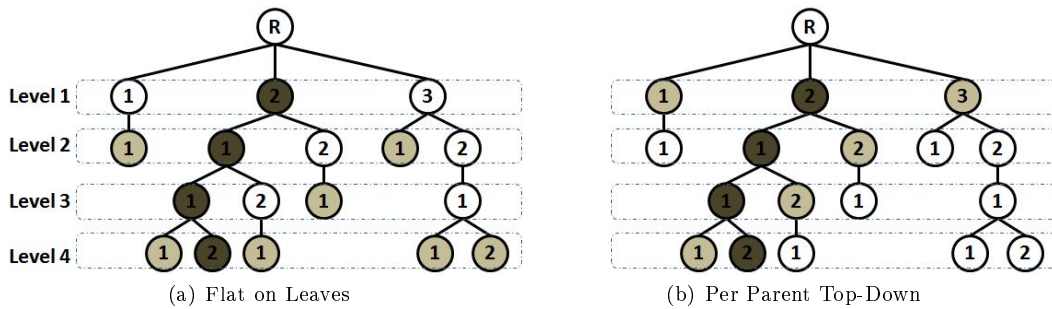


Fig. 7. Simulating the FL and PPTD classifiers

Figure 7(b) illustrates the implemented PPTD classifier. Each shaded class was included in the training of the classifier assigned to its parent class. Initially, the classifier assigned to the root node predicts the class 2. Then the classifier assigned to the class 2 is executed and predicts the class 2.1.

Repeating this procedure, the classifier assigned to the class 2.1 predicts the class 2.1.1 and, finally, the classifier assigned to the class 2.1.1 predicts the class 2.1.1.2.

5. COMPUTATIONAL EXPERIMENTS

Computational experiments were conducted in order to evaluate the performance of the proposed hierarchical classification methods. We used 18 bioinformatics data sets, where the hierarchical classes to be predicted are protein or gene functions and the predictor attributes are protein or gene properties. These datasets are organized into two large groups.

Group A contains eight data sets of protein functions. These data sets are divided into two main subgroups: GPCR (G-Protein-Coupled Receptor) and EC (Enzyme Commission). The GPCR group consists of four data sets (GPCRpfam, GPCRprints, GPCRprosite and GPCRinterpro). GPCRs are proteins that transmit signals from outside to inside the cell, changing the cell's behavior. The classes refer to types of GPCRs, and the different names of the datasets indicate the different types of predictor attributes used in each dataset, where Pfam, Prints, Prosite and Interpro are different types of protein motifs (or protein signatures). The EC group consists of four data sets (ECpfam, ECprints, ECprosite and ECinterpro), where the classes represent types of enzymes. An enzyme is a type of protein that speeds up chemical reactions. These data sets are available at <http://www.cs.kent.ac.uk/archive/people/rpg/nh56/datasets.zip>. These data sets have been used in several works on hierarchical classification problems [Costa et al. 2007; Costa et al. 2008; Freitas and de Carvalho 2007; Holden and Freitas 2008; Silla and Freitas 2011]. A pre-processing procedure was performed on these data sets to remove, from each data set, instances whose most specific class was not a leaf class node, since in this work we focus on the problem of mandatory leaf node prediction, as mentioned earlier.

Group B contains ten gene function data sets, which contain information related to the *Yeast* fungus. These data sets are presented in [Clare and King 2003] and were, originally, multilabel data, where some instances had two or more labels at a given class level. In order to be used in this work (where instances are assumed to have only a single label per class level), these data sets were converted into single label data by randomly choosing one class among the classes assigned to each level for each instance, but also guaranteeing that there was no vertical inconsistency in the chosen classes.

All data sets have the class hierarchy represented by a non-complete tree structure with four levels. In addition, for each instance, its most specific class is always assigned to a leaf node of the hierarchy. The main characteristics of the datasets are shown in Table I: the group of each data set (Group), the data set name (Data Set), the number of classes for each level of the hierarchy (#Classes) and the total number of instances in each data set (#Instances).

Table I. Characteristics of the Data Sets

Group	Data Sets	#Classes	#Instances	Group	Data Sets	#Classes	#Instances
A	GPCRpfam	12/52/79/49	6524	B	Church	4/18/36/27	1677
	GPCRprints	8/46/76/49	4880		CellCycle	4/17/34/23	1711
	GPCRprosite	9/50/79/49	5728		Derisi	4/18/35/25	1661
	GPCRinterpro	12/54/82/50	6935		Eisen	4/15/29/17	1163
	ECpfam	6/41/96/190	11057		Expr	4/17/34/25	1688
	ECprints	6/45/92/208	11048		Gasch1	4/17/34/25	1660
	ECprosite	6/42/89/187	11328		Gasch2	4/17/33/25	1678
	ECinterpro	6/41/96/187	11101		Phenotype	4/12/21/13	621
					Sequence	4/17/32/24	1680
					SPO	4/17/34/25	1649

All hierarchical classifiers have been implemented using the JAVA programming language. Some functions and algorithms were imported from the data mining tool WEKA 3.7.0 (Waikato Environment for Knowledge Analysis)[Witten and Frank 2005]. In order to adopt the traditional classifiers k-NN

and C4.5, used in the experiments as flat classifiers, we used the Ibk and J48 implementations available in the WEKA tool, respectively.

We evaluated the performance of the hierarchical classifiers using the Hierarchical F-measure (HF), defined next. This is an adaptation of the F-measure to the hierarchical context, described in [Kiritchenko et al. 2005].

Let P_i be the set of classes predicted for all levels of the hierarchy for the test instance i and T_i be the set of true classes for all class levels for instance i . Let n be the total number of instances. The Hierarchical Precision (HP) is defined by $HP = \sum_i |P_i \cap T_i| / \sum_i |P_i|$ and represents the ratio between the sum, for all instances, of the number of common classes between the sets of predicted and true classes of each instance and the sum, for all instances, of the number of predicted classes of each instance. The Hierarchical Recall (HR) is defined by $HR = \sum_i |P_i \cap T_i| / \sum_i |T_i|$ and represents the ratio between the sum, for all instances, of the number of common classes between the sets of predicted and true classes of each instance and the sum, for all instances, of the number of true classes of each instance. The Hierarchical F-measure (HF) is defined then by $HF = 2 * HP * HR / (HP + HR)$ and represents the harmonic mean of HP and HR.

The evaluation of the hierarchical classifiers was performed using 10-fold cross-validation and the HF measure. We used the paired and two-tailed version of the Student's t-test [Jain 1991], with significance level equal to 5%, to evaluate the statistical significance when comparing two hierarchical classifiers.

We first compare the performance of the two proposed hierarchical classifiers – Sum of Votes and Sum of Weighted Votes – and then we conduct the comparisons between the traditional hierarchical classifiers – Flat on Leaves and Per Parent Top-Down – and the proposed hierarchical classifier which achieved the best results (out of the two proposed classifiers).

5.1 Comparing Sum of Votes and Sum of Weighted Votes Strategies

The proposed class-inconsistency removal strategies, Sum of Votes (SV) and Sum of Weighted Votes (SWV), were compared based on six different flat classifiers: 1-NN, 3-NN, 5-NN, 7-NN, 9-NN e C4.5. In essence, the strategies SV and SWV apply the flat classifier in each hierarchy class level and then execute their heuristics for class-inconsistency elimination.

In Table II, for each combination of data set and classifier, the HF measure values achieved by 10-fold cross-validation (with the standard deviation in parentheses) are presented. For each combination of dataset and classifier, the best result (out of the two proposed class-inconsistency removal strategies) is shown in bold. In addition, for each dataset, the best result across all six classifiers is shown underlined. The symbol (•) between the two HF values (one associated with the SV and the other associated with the SWV strategy) indicates that the difference between these values are statistically significant. The symbol (-) indicates no statistical significance. For example, for the data set GPRCpfam, with flat classifier 1-NN, the SWV strategy achieved HF value equal to 70.31%, overcoming, with statistical significance, the SV strategy, which achieved HF value equal to 67.31%. In addition, the value 70.31% is underlined since it was the best result found for the GPCRpfam database, across the six classifiers. Finally, below each data set group, a total row displays, for each classifier adopted, the number of times in which each of the two class-inconsistency removal strategies obtained a better or equal HF value when compared to the other.

In Table II, we observe that the SWV strategy achieved a better predictive performance than the SV strategy. For both data set groups and all flat classifiers adopted, we note from the total rows that the number of better results (in bold) obtained by the SWV strategy is always greater than the number obtained by the SV strategy. We also observe that the number of better results per data set (underlined) obtained by the SWV strategy (16 times) is much greater than that achieved by the SV strategy (5 times).

Table II. Comparison of HF values (%) between SV and SWV class-inconsistency removal strategies

Databases	1-NN				3-NN				5-NN			
	SV		SWV		SV		SWV		SV		SWV	
GPCRpfam	67.31	(1.55)	● 70.31	(1.52)	66.60	(0.94)	● 69.72	(1.08)	66.44	(1.02)	● 69.20	(1.32)
GPCRprints	82.17	(1.15)	● 83.00	(1.20)	81.42	(0.79)	● 82.09	(0.82)	80.41	(0.82)	● 81.05	(0.87)
GPCRprosite	68.92	(1.18)	● 69.26	(1.19)	67.46	(0.92)	● 68.05	(0.86)	67.66	(0.49)	- 67.83	(0.58)
GPCRinterpro	80.14	(0.81)	● 83.09	(0.70)	79.85	(0.94)	● 82.73	(0.91)	79.87	(0.91)	● 82.15	(0.73)
ECpfam	98.77	(0.17)	- 98.77	(0.17)	98.45	(0.33)	- 98.44	(0.31)	98.27	(0.36)	● 98.37	(0.33)
ECprints	98.16	(0.24)	- 98.19	(0.24)	97.86	(0.25)	- 97.86	(0.26)	97.63	(0.31)	- 97.64	(0.30)
ECprosite	98.78	(0.27)	● 98.80	(0.26)	98.66	(0.21)	- 98.67	(0.20)	98.46	(0.27)	- 98.47	(0.27)
ECinterpro	99.08	(0.29)	- 99.08	(0.29)	98.81	(0.40)	- 98.83	(0.40)	98.78	(0.42)	- 98.83	(0.41)
Total A	2		8		2		7		0		8	
Church	19.70	(1.97)	- 19.70	(1.97)	19.63	(2.19)	● 20.27	(1.94)	20.27	(2.02)	● 20.53	(2.18)
CellCycle	24.82	(2.34)	- 24.82	(2.34)	23.77	(3.38)	● 24.52	(3.59)	27.52	(2.46)	- 28.15	(2.98)
Derisi	18.73	(2.33)	- 18.73	(2.33)	16.58	(2.26)	● 17.94	(2.57)	18.88	(2.46)	● 20.06	(2.54)
Eisen	24.54	(2.32)	- 24.54	(2.32)	23.66	(2.87)	● 25.19	(3.40)	27.36	(3.14)	- 28.31	(2.95)
Expr	25.62	(3.56)	- 25.62	(3.56)	22.42	(1.48)	● 23.44	(1.92)	26.37	(2.36)	- 27.13	(2.63)
Gasch1	28.98	(2.01)	- 28.98	(2.01)	27.29	(1.51)	● 28.62	(1.77)	29.50	(2.42)	- 30.07	(1.88)
Gasch2	25.04	(3.03)	- 25.03	(3.04)	22.92	(2.95)	- 23.26	(2.71)	23.57	(3.43)	- 23.92	(3.16)
Phenotype	22.04	(2.57)	- 22.46	(2.91)	23.14	(2.05)	- 23.66	(3.24)	22.75	(1.82)	- 23.23	(2.11)
Sequence	23.08	(3.14)	- 23.08	(3.14)	21.73	(2.25)	- 22.38	(2.56)	22.22	(2.66)	● 22.82	(2.69)
SPO	18.49	(2.66)	- 18.50	(2.68)	17.81	(1.60)	● 19.15	(1.89)	19.42	(2.63)	● 19.92	(2.73)
Total B	8		9		0		10		0		10	

Databases	7-NN				9-NN				C4.5			
	SV		SWV		SV		SWV		SV		SWV	
GPCRpfam	66.15	(0.97)	● 68.68	(1.16)	65.58	(0.93)	● 68.07	(1.10)	68.79	(1.24)	● 68.70	(1.21)
GPCRprints	80.33	(1.07)	● 80.86	(1.12)	79.75	(0.91)	● 80.28	(0.89)	78.70	(0.83)	● 79.33	(0.85)
GPCRprosite	66.67	(1.03)	● 66.83	(1.00)	65.95	(0.90)	- 66.00	(0.87)	67.24	(0.85)	- 67.08	(1.00)
GPCRinterpro	79.57	(0.68)	● 81.68	(0.53)	79.17	(1.18)	● 81.31	(0.91)	81.26	(1.09)	● 81.80	(1.03)
ECpfam	98.27	(0.34)	- 98.30	(0.29)	98.19	(0.31)	- 98.19	(0.31)	98.42	(0.23)	- 98.43	(0.23)
ECprints	97.45	(0.26)	- 97.45	(0.26)	97.25	(0.37)	- 97.22	(0.41)	97.47	(0.26)	- 97.51	(0.31)
ECprosite	98.33	(0.32)	- 98.33	(0.33)	98.09	(0.30)	- 98.08	(0.30)	98.57	(0.18)	- 98.57	(0.18)
ECinterpro	98.80	(0.32)	- 98.82	(0.34)	98.71	(0.34)	- 98.70	(0.32)	98.79	(0.38)	- 98.79	(0.38)
Total A	2		8		4		5		4		6	
Church	20.25	(2.10)	- 20.19	(2.10)	20.40	(2.05)	- 20.66	(2.04)	21.29	(2.39)	- 21.29	(2.47)
CellCycle	28.67	(3.05)	- 28.60	(3.28)	29.85	(4.24)	- 29.87	(3.86)	23.96	(2.70)	● 24.83	(3.10)
Derisi	20.38	(2.61)	● 21.16	(2.60)	20.71	(2.43)	- 21.20	(2.66)	21.59	(2.14)	- 21.78	(1.97)
Eisen	29.14	(2.69)	- 29.56	(2.80)	30.39	(2.05)	- 30.77	(1.83)	25.27	(2.86)	- 25.79	(2.67)
Expr	27.94	(2.99)	- 28.12	(3.08)	29.22	(3.26)	- 29.26	(2.52)	26.72	(2.45)	- 26.61	(2.53)
Gasch1	31.33	(2.23)	- 31.24	(2.22)	30.59	(2.62)	- 30.80	(2.56)	25.24	(2.69)	- 25.86	(2.28)
Gasch2	25.28	(2.54)	- 25.55	(2.00)	25.47	(1.68)	- 26.21	(1.34)	23.45	(2.63)	- 23.39	(2.21)
Phenotype	24.15	(2.87)	- 24.42	(3.01)	25.46	(2.96)	- 25.45	(3.32)	26.50	(2.58)	- 26.37	(3.35)
Sequence	22.78	(3.32)	- 22.76	(2.94)	23.70	(2.57)	- 23.75	(2.65)	24.44	(3.01)	● 25.71	(2.87)
SPO	21.40	(2.30)	- 21.59	(2.25)	22.01	(1.63)	- 22.31	(1.64)	20.90	(2.20)	- 21.50	(2.58)
Total B	4		6		1		9		4		7	

We also note that, for data sets in Group A, the best results were achieved when the 1-NN classifier was adopted. On the other hand, for data sets in Group B, the best results were obtained when 9-NN and C.45 classifiers were adopted.

Considering these experiments, we conclude that the Sum of Weighted Votes strategy achieves a better performance than the Sum of Votes strategy and, therefore, it will be the strategy adopted for comparison with the traditional hierarchical classifiers (Flat on Leaves and Per Parent Top-Down).

5.2 Comparing Sum of Weighted Votes and the Traditional Strategies

In the next evaluation, the local classifier per level approach using the best proposed class-inconsistency removal strategy, Sum of Weighted Votes, is compared with two traditional hierarchical classifiers, Flat on Leaves (FL) and Per Parent Top-Down (PPTD).

Tables III and IV present, for data sets of Group A and Group B, respectively, the results of the hierarchical approaches FL, PPTD, and SWV, considering all six flat classifiers adopted. For each combination of data set and flat classifier, the HF values, and their respective standard deviations, obtained by the three approaches are shown.

For each combination of dataset and flat classifier, the best result (out of the three approaches) is shown in bold. The symbol (\blacktriangle) on the right side of the HF value obtained by the FL and PPTD classifiers indicates that the value obtained by the local classifier per level approach using the proposed SWV strategy overcame the FL or PPTD classifiers. The symbol (\blacktriangledown) indicates that the HF value obtained by the FL or PPTD classifier overcame the result of the local classifier per level approach using the SWV strategy with statistical significance. The symbol (-) indicates that the difference in predictive performance was not statistically significant. For example, in Table III, for the data set GPCRpfam, with the 3-NN flat classifier, the symbol (\blacktriangle) beside the result for the FL classifier represents that the performance of the local classifier per level approach with the SWV strategy was better than the FL classifier, with statistical significance.

In Table III, we observe that, for the data sets of Group A, the difference of performance among the three classifiers compared was not significant. From the total rows, we note that the number of better results associated with the SWV strategy (in bold) was equal to or higher than the number of better results associated with the traditional hierarchical classifiers FL and PPTD, when the flat classifiers 1-NN and 5-NN were adopted. On the other hand, the PPTD classifier achieved a better performance when the 3-NN flat classifier was used. In addition, the FL classifier was superior when the flat classifiers 7-NN, 9-NN and C4.5 were adopted. We also observe that all best results for each data set (underlined) were obtained when the 1-NN flat classifier was used and, in this case, the three strategies obtained similar performances. We conclude that, for the data sets in Group A, the local classifier per level approach using the proposed SWV strategy showed a competitive behavior in comparison with the traditional classifiers FL and PPTD.

Table III. Comparison of HF values (%) between classifiers SWV, FL and PPTD - Group A

Databases	1-NN						3-NN					
	FL		PPTD		SWV		FL		PPTD		SWV	
GPCRpfam	70.31	(1.51)	-	70.32	(1.52)	- 70.31 (1.52)	69.34	(1.14)	▲ 70.07	(0.88)	▼ 69.72	(1.08)
GPCRprints	82.98	(1.21)	-	82.97	(1.19)	- 83.00 (1.20)	82.00	(0.79)	▲ 82.06	(0.70)	- 82.09	(0.82)
GPCRprosite	69.28	(1.19)	-	69.25	(1.23)	- 69.26 (1.19)	67.81	(0.99)	▲ 68.39	(1.11)	- 68.05	(0.86)
GPCRinterpro	83.09	(0.71)	-	83.09	(0.70)	- 83.09 (0.70)	82.63	(0.83)	- 82.85	(0.94)	- 82.73	(0.91)
ECpfam	98.77	(0.17)	-	98.77	(0.17)	- 98.77 (0.17)	98.44	(0.33)	- 98.42	(0.31)	- 98.44	(0.3)
ECprints	98.19	(0.24)	-	98.19	(0.23)	- 98.19 (0.24)	97.87	(0.27)	- 97.87	(0.25)	- 97.86	(0.26)
ECprosite	98.80	(0.26)	-	98.80	(0.26)	- 98.80 (0.26)	98.70	(0.19)	- 98.66	(0.21)	- 98.67	(0.20)
ECinterpro	99.08	(0.29)	-	99.07	(0.29)	- 99.08 (0.29)	98.79	(0.42)	▲ 98.85	(0.37)	- 98.83	(0.40)
Total A	6		5		6		3		5		2	

Databases	5-NN						7-NN					
	FL		PPTD		SWV		FL		PPTD		SWV	
GPCRpfam	69.24	(1.21)	-	69.53	(0.93)	- 69.20 (1.32)	69.33	(1.26)	▼ 69.04	(0.90)	- 68.68	(1.16)
GPCRprints	81.55	(0.84)	▼ 81.03	(0.89)	- 81.05 (0.87)	81.28	(1.06)	▼ 80.89	(1.08)	- 80.86	(1.12)	
GPCRprosite	67.97	(0.58)	-	68.34	(0.64)	▼ 67.83 (0.58)	67.66	(0.91)	▼ 67.31	(0.98)	▼ 66.83	(1.00)
GPCRinterpro	82.43	(0.75)	▼ 82.20	(0.78)	- 82.15 (0.73)	82.03	(0.58)	- 81.95	(0.69)	▼ 81.68	(0.53)	
ECpfam	98.26	(0.42)	▲ 98.24	(0.38)	▲ 98.37	(0.33)	98.27	(0.41)	- 98.16	(0.36)	▲ 98.30	(0.29)
ECprints	97.63	(0.30)	-	97.60	(0.32)	- 97.64 (0.30)	97.49	(0.28)	- 97.37	(0.28)	▲ 97.45	(0.26)
ECprosite	98.47	(0.25)	-	98.45	(0.27)	- 98.47 (0.27)	98.37	(0.31)	▼ 98.29	(0.32)	- 98.33	(0.33)
ECinterpro	98.70	(0.40)	▲ 98.76	(0.41)	- 98.83	(0.41)	98.73	(0.30)	▼ 98.62	(0.32)	▼ 98.82	(0.34)
Total A	3		2		4		6		0		2	

Databases	9-NN						C4.5						
	FL		PPTD		SWV		FL		PPTD		SWV		
GPCRpfam	68.62	(1.34)	-	68.47	(0.83)	- 68.07 (1.10)	69.32	(1.23)	- 68.84	(1.14)	- 68.70	(1.21)	
GPCRprints	80.54	(1.06)	-	80.41	(0.96)	- 80.28 (0.89)	81.76	(0.52)	▼ 79.19	(0.94)	- 79.33	(0.85)	
GPCRprosite	66.67	(0.87)	▼ 66.57	(0.70)	▼ 66.00 (0.87)	68.03	(1.01)	▼ 67.63	(1.17)	▼ 67.08	(1.00)		
GPCRinterpro	81.99	(0.90)	▼ 81.29	(1.05)	- 81.31 (0.91)	82.66	(0.72)	▼ 81.54	(0.57)	- 81.80	(1.03)		
ECpfam	98.18	(0.33)	-	97.88	(0.33)	▲ 98.19	(0.31)	98.33	(0.24)	▲ 98.39	(0.24)	- 98.43	(0.23)
ECprints	97.27	(0.38)	-	97.05	(0.34)	▲ 97.22 (0.41)	97.56	(0.30)	- 97.35	(0.27)	▲ 97.51	(0.31)	
ECprosite	98.16	(0.29)	▼ 98.03	(0.31)	▲ 98.08 (0.30)	98.50	(0.19)	- 98.46	(0.17)	▲ 98.57	(0.18)		
ECinterpro	98.72	(0.35)	-	98.32	(0.40)	▲ 98.70 (0.32)	98.74	(0.38)	▲ 98.73	(0.34)	- 98.79	(0.38)	
Total A	7		0		1		5		0		3		

In Table IV we note that, for data sets in Group B, the proposed SWV strategy obtained a better predictive performance than the other two strategies. We observe, from the total rows, that the

number of better results associated with the SWV strategy (in bold) was greater than the number of better results associated with the classifiers FL and PPTD when the flat classifiers 5-NN, 7-NN, 9-NN and C4.5 were used, many times with statistical significance. We note that nine out of the ten best results per data set (underlined results) were obtained by the SWV strategy. We concluded that, for the data sets of Group B, the local classifier per level approach using the proposed SWV strategy has shown a superior behavior when compared to the traditional classifiers FL and PPTD.

Table IV. Comparison of HF values (%) between classifiers SWV, FL and PPTD - Group B

Databases	1-NN			3-NN		
	FL	PPTD	SWV	FL	PPTD	SWV
Church	21.26 (1.58) ▼	19.38 (1.90) -	19.70 (1.97)	20.44 (1.72) -	20.20 (2.24) -	20.27 (1.94)
CellCycle	24.83 (2.55) -	24.56 (2.10) -	24.82 (2.34)	23.00 (3.27) -	24.79 (3.62) -	24.52 (3.59)
Derisi	18.87 (2.50) -	18.89 (2.23) -	18.73 (2.33)	15.84 (2.05) ▲	19.05 (2.50) ▼	17.94 (2.57)
Eisen	24.72 (2.88) -	24.70 (2.02) -	24.54 (2.32)	23.22 (2.60) -	25.31 (3.97) -	25.19 (3.40)
Expr	25.05 (3.40) ▲	25.35 (3.28) -	25.62 (3.56)	21.65 (1.16) ▲	24.11 (2.72) -	23.44 (1.92)
Gasch1	28.84 (1.70) -	28.29 (1.47) -	28.98 (2.01)	25.85 (2.56) ▲	29.25 (2.45) -	28.62 (1.77)
Gasch2	24.58 (3.31) -	25.23 (2.97) -	25.03 (3.04)	21.51 (2.86) ▲	23.98 (2.54) -	23.26 (2.71)
Phenotype	20.67 (3.75) ▲	20.27 (3.66) -	22.46 (2.91)	22.29 (3.80) -	21.51 (2.76) -	23.66 (3.24)
Sequence	22.43 (2.97) -	24.02 (3.42) ▼	23.08 (3.14)	20.44 (1.94) ▲	23.01 (2.70) -	22.38 (2.56)
SPO	18.23 (2.26) -	18.86 (2.41) -	18.50 (2.68)	16.89 (2.11) ▲	19.94 (1.74) -	19.15 (1.89)
Total B	3	4	3	1	8	1

Databases	5-NN			7-NN		
	FL	PPTD	SWV	FL	PPTD	SWV
Church	21.82 (1.53) -	19.53 (2.84) -	20.53 (2.18)	21.72 (1.11) ▲	19.66 (2.27) ▲	20.19 (2.10)
CellCycle	25.52 (2.44) ▲	27.36 (2.78) -	28.15 (2.98)	27.47 (2.16) -	28.38 (3.33) -	28.60 (3.28)
Derisi	17.11 (2.53) ▲	18.83 (2.51) -	20.06 (2.54)	18.88 (2.83) ▲	20.21 (2.61) -	21.16 (2.60)
Eisen	25.24 (2.63) ▲	26.40 (3.36) ▲	28.31 (2.95)	28.28 (2.49) -	29.25 (2.47) -	29.56 (2.80)
Expr	23.87 (2.33) ▲	26.47 (1.99) -	27.13 (2.63)	25.92 (1.84) ▲	27.21 (3.11) -	28.12 (3.08)
Gasch1	26.62 (2.39) ▲	29.76 (2.55) -	30.07 (1.88)	29.16 (2.86) -	30.97 (2.67) -	31.24 (2.22)
Gasch2	22.56 (3.11) ▲	24.79 (3.52) -	23.92 (3.16)	24.37 (2.67) -	26.12 (2.66) -	25.55 (2.00)
Phenotype	23.10 (4.06) -	21.33 (3.13) -	23.23 (2.11)	24.57 (3.51) -	22.52 (4.06) -	24.42 (3.01)
Sequence	21.85 (2.93) -	23.60 (2.24) -	22.82 (2.69)	23.59 (2.30) -	23.73 (3.39) -	22.76 (2.94)
SPO	17.35 (2.16) ▲	20.41 (2.54) -	19.92 (2.73)	19.14 (1.79) ▲	21.10 (3.19) -	21.59 (2.25)
Total B	1	3	6	2	2	6

Databases	9-NN			C4.5		
	FL	PPTD	SWV	FL	PPTD	SWV
Church	22.10 (1.41) -	19.82 (2.66) -	20.66 (2.04)	21.74 (1.76) -	21.53 (1.93) -	21.29 (2.47)
CellCycle	29.08 (2.30) -	29.40 (2.95) -	29.87 (3.86)	21.86 (3.75) ▲	22.19 (2.41) ▲	24.83 (3.10)
Derisi	20.17 (3.14) -	20.89 (2.36) -	21.20 (2.66)	18.28 (2.77) ▲	20.47 (2.70) -	21.78 (1.97)
Eisen	28.73 (3.02) ▲	29.16 (2.63) -	30.77 (1.83)	22.96 (3.14) -	24.24 (3.19) -	25.79 (2.67)
Expr	27.58 (2.69) ▲	28.29 (2.19) -	29.26 (2.52)	24.98 (2.93) -	24.74 (2.22) ▲	26.61 (2.53)
Gasch1	30.40 (2.41) -	30.08 (2.90) -	30.80 (2.56)	23.09 (2.95) ▲	22.90 (2.05) ▲	25.86 (2.28)
Gasch2	25.89 (2.24) -	26.00 (2.24) -	26.21 (1.34)	20.76 (3.11) ▲	22.52 (3.58) -	23.39 (2.21)
Phenotype	25.77 (3.38) -	22.95 (4.75) -	25.45 (3.32)	22.86 (3.86) ▲	21.39 (3.94) ▲	26.37 (3.35)
Sequence	23.64 (2.41) -	23.90 (2.58) -	23.75 (2.65)	19.68 (2.31) ▲	22.86 (2.59) ▲	25.71 (2.87)
SPO	20.47 (2.08) ▲	22.29 (2.49) -	22.31 (1.64)	19.86 (2.45) -	19.36 (2.13) ▲	21.50 (2.58)
Total B	2	1	7	1	0	9

Table V summarizes the previous results presented in Tables III and IV for the local classifier per level approach with SWV strategy, FL and PPTD hierarchical classifiers. For each data set, it is shown the best result for each hierarchical classifier and the respective flat classifier used to find it.

For the data sets in Group A, we note that the results of the three hierarchical approaches are very close. However, we observe that the local classifier per level with the SWV strategy presents a competitive behavior, since it obtained, in total, six best results for the eight data sets. For the data sets in Group B, the local classifier per level with the SWV strategy showed a better performance than the other hierarchical classifiers in nine out of ten data sets.

From these results, we conclude that the local classifier per level with the proposed SWV strategy has shown good predictive performance, achieving results equal to or better than two other traditional hierarchical classification approaches in 15 out of 18 data sets used in the experiments.

Table V. Best HF results found for each hierarchical strategy

Databases	FL		PPTD		SWV	
	HF(%)	Flat Classifier	HF(%)	Flat Classifier	HF(%)	Flat Classifier
GPCRpfam	70.31	1-NN	70.32	1-NN	70.31	1-NN
GPCRprints	82.98	1-NN	82.97	1-NN	83.00	1-NN
GPCRprosite	69.28	1-NN	69.25	1-NN	69.26	1-NN
GPCRinterpro	83.09	1-NN	83.09	1-NN	83.09	1-NN
ECpfam	98.77	1-NN	98.77	1-NN	98.77	1-NN
ECprints	98.19	1-NN	98.19	1-NN	98.19	1-NN
ECprosite	98.80	1-NN	98.80	1-NN	98.80	1-NN
ECinterpro	99.08	1-NN	99.07	1-NN	99.08	1-NN
Total A	6		5		6	
Church	22.10	9-NN	21.53	C4.5	21.29	C4.5
CellCycle	29.08	9-NN	29.40	9-NN	29.87	9-NN
Derisi	20.17	9-NN	20.89	9-NN	21.78	C4.5
Eisen	28.73	9-NN	29.25	7-NN	30.77	9-NN
Expr	27.58	9-NN	28.29	9-NN	29.26	9-NN
Gasch1	30.40	9-NN	30.97	7-NN	31.24	7-NN
Gasch2	25.89	9-NN	26.12	7-NN	26.21	9-NN
Phenotype	25.77	9-NN	22.95	9-NN	26.37	C4.5
Sequence	23.64	9-NN	24.02	1-NN	25.71	C4.5
SPO	20.47	9-NN	22.29	9-NN	22.31	9-NN
Total B	1		0		9	

Table VI shows the best results achieved for each data set and the hierarchical approaches that have achieved it. Again, we observe a balanced behavior among the three hierarchical approaches for data sets in Group A, and a better performance of the local classifier per level approach using the SWV strategy for the data sets in Group B.

Table VI. Best HF results found for each database

Group	Databases	HF(%)	Strategies	Group	Databases	HF(%)	Strategies
A	GPCRpfam	70.32	PPTD/1-NN	B	Church	22.10	FL/9-NN
	GPCRprints	83.00	SWV/1-NN		CellCycle	29.87	SWV/9-NN
	GPCRprosite	69.28	FL/1-NN		Derisi	21.78	SWV/C4.5
	GPCRinterpro	83.09	FL/1-NN, PPTD/1-NN, SWV/1-NN		Eisen	30.77	SWV/9-NN
	ECpfam	98.77	FL/1-NN, PPTD/1-NN, SWV/1-NN		Expr	29.26	SWV/9-NN
	ECprints	98.19	FL/1-NN, PPTD/1-NN, SWV/1-NN		Gasch1	31.24	SWV/7-NN
	ECprosite	98.80	FL/1-NN, PPTD/1-NN, SWV/1-NN		Gasch2	26.21	SWV/9-NN
	ECinterpro	99.08	FL/1-NN, SWV/1-NN		Phenotype	26.37	SWV/C4.5
					Sequence	25.71	SWV/C4.5
					SPO	22.31	SWV/9-NN

6. CONCLUSIONS

Hierarchical classification problems have been explored with strategies pertaining to different categories. The local classification approach is one of the most used and efficient paradigms for hierarchical classification, where the algorithm is aware of the class hierarchy through a local (rather than global) perspective. In spite of requiring the training of a reduced number of classifiers, one per hierarchical level, the local classifier per level approach has not been properly addressed in the literature yet. Probably, this is due to inconsistent class predictions, common to this hierarchical paradigm.

In this work, we proposed two local classifier per level hierarchical classification approaches, called Sum of Votes (SV) and Sum of Weighted Votes (SWV). Each of these approaches uses a distinct strategy to solve inconsistent predictions. Using 18 hierarchical classification data sets, we evaluated both strategies and concluded that SWV was able to reach higher predictive accuracies. Next, using the same data sets, we compared the SWV strategy with two traditional hierarchical classification approaches: the flat classification and the local classifier per parent approach. The obtained results showed that the Sum of Weighted Votes strategy presented again a competitive and robust performance in terms of prediction accuracy, evidencing the importance of the local classifier per level approach.

The next step of this research is to couple attribute selection procedures with the hierarchical classifiers. We believe the proposed strategy could benefit from considering only the important features in each level of the hierarchy when building the local classifiers in the local classifier per level approach.

REFERENCES

- BARBEDO, J. G. A. AND LOPES, A. Automatic genre classification of musical signals. *EURASIP Journal on Applied Signal Processing* 2007 (1): 157–157, 2007.
- BARUTCUOGLU, Z. AND DECORO, C. Hierarchical shape classification using Bayesian aggregation. In *Shape Modeling International*. Matsushima, Japan, pp. 44, 2006.
- BURRED, J. J. AND LERCH, A. A hierarchical approach to automatic musical genre classification. In *Procs. of the 6th Int. Conf. on Digital Audio Effects (DAFx)*. London, England, pp. 8–11, 2003.
- CECI, M. AND MALERBA, D. Classifying web documents in a hierarchy of categories: a comprehensive study. *Journal of Intelligent Information Systems* 28 (1): 37–78, 2007.
- CLARE, A. AND KING, R. D. Predicting gene function in *saccharomyces cerevisiae*. In *Proceedings of the European Conference on Computational Biology (ECCB 2003)*. Paris, France, pp. 42–49, 2003.
- COSTA, E. P., LORENA, A. C., CARVALHO, A. C. P. L. F., AND FREITAS, A. A. Top-down hierarchical ensembles of classifiers for predicting g-protein-coupled-receptor functions. In *Proceedings of the 3rd Brazilian Symp. on Bioinformatics, Lecture Notes in Bioinformatics 5167*. Santo André, Brazil, pp. 35–46, 2008.
- COSTA, E. P., LORENA, A. C., CARVALHO, A. C. P. L. F., FREITAS, A. A., AND HOLDEN, H. Comparing several approaches for hierarchical classification of proteins with decision trees. In *Proceedings of the 2nd Brazilian Symp. on Bioinformatics, Lecture Notes in Bioinformatics 4643*. Angra dos Reis, Brazil, pp. 126–137, 2007.
- DUMAIS, S. AND CHEN, H. Hierarchical classification of web content. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. Athens, Greece, pp. 256–263, 2000.
- EISNER, R., POULIN, B., SZAFRON, D., LU, P., AND GREINER, R. Improving protein function prediction using the hierarchical structure of the gene ontology. In *Proceedings of the IEEE CIBCB*. La Jolla, USA, pp. 354–363, 2005.
- FAGNI, T. AND SEBASTIANI, F. On the selection of negative examples for hierarchical text categorization. In *Proceedings of the 3rd Language Technology*. Poznan, Poland, pp. 24–28, 2007.
- FREITAS, A. AND DE CARVALHO, A. C. VII. In D. Taniar (Ed.), *A Tutorial on Hierarchical Classification with Applications in Bioinformatics*. Vol. Research and Trends in Data Mining Technologies and Applications. Idea Group, pp. 175–208, 2007.
- HOLDEN, N. AND FREITAS, A. A. Improving the performance of hierarchical classification with swarm intelligence. In *Proceedings of the 6th European Conference on Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics, LNCS 4973*. Valencia, Spain, pp. 48–60, 2008.
- JAIN, R. *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling*. Wiley, 1991.
- KIRITCHENKO, S., MATWIN, S., AND FAMILI, A. F. Functional annotation of genes using hierarchical text categorization. In *Proceedings of the BioLINK SIG: Linking Literature, Information and Knowledge for Biology*. Detroit, USA, 2005.
- KOLLER, D. AND SAHAMI, M. Hierarchically classifying documents using very few words. In *Proceedings of the 14th International Conference on Machine Learning*. Nashville, USA, pp. 170–178, 1997.
- LABROU, Y. AND FININ, T. Yahoo! as an ontology: using yahoo! categories to describe documents. In *Proceedings of the 8th International Conference on Information and Knowledge Management*. Kansas, USA, pp. 180–187, 1999.
- ROCCHIO, J. J. Relevance feedback in information retrieval. In *The Smart retrieval system - experiments in automatic document processing*, G. Salton (Ed.). Prentice-Hall, pp. 313–323, 1971.
- SECKER, A., DAVIES, M. N., FREITAS, A. A., CLARK, E. B., TIMMIS, J., AND FLOWER, D. R. Hierarchical classification of g-protein-coupled receptors with data-driven selection of attributes and classifiers. *International Journal of Data Mining and Bioinformatics* 4 (2): 191–210, 2010.
- SECKER, A., DAVIES, M. N., FREITAS, A. A., TIMMIS, J., MENDAO, M., AND FLOWER, D. R. An experimental comparison of classification algorithms for the hierarchical prediction of protein function. In *3rd UK Data mining and Knowledge Discovery Symposium (UKKDD 2007)*. Kent, England, pp. 13–18, 2007.
- SILLA, C. AND FREITAS, A. A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery* 22 (1-2): 31–72, 2011.
- SILLA, C. N. AND FREITAS, A. A global-model naive bayes approach to the hierarchical prediction of protein functions. In *Proceedings of the 9th IEEE International Conference on Data Mining*. Miami, USA, pp. 992–997, 2009.
- VALENTINI, G. True path rule hierarchical ensembles. In *Proceedings of the 8th International Workshop on Multiple Classifier Systems*. Reykjavik, Iceland, pp. 232–241, 2009.
- WITTEN, I. H. AND FRANK, E. *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition*. Morgan Kaufmann Publishers Inc., 2005.
- WU, F., ZHANG, J., AND HONAVAR, V. Learning classifiers using hierarchically structured class taxonomies. In *SARA*. Vol. 3607. Airth Castle, Scotland, pp. 313–320, 2005.
- XIAO, Z., DELLANDREA, E., DOU, W., AND CHEN, L. Automatic hierarchical classification of emotional speech. In *Proceedings of the 9th IEEE International Symposium on Multimedia Workshops*. Taichung, Taiwan, pp. 291–296, 2007.