An Extended Local Hierarchical Classifier for Prediction of Protein and Gene Functions

Luiz H. C. Merschmann[1] and Alex A. Freitas[2]

[1] Federal University of Ouro Preto, Computer Science Department, Brazil
luizhenrique@iceb.ufop.br
[2] University of Kent, School of Computing, UK
a.a.freitas@kent.ac.uk

Abstract. Gene function prediction and protein function prediction are complex classification problems where the functional classes are structured according to a predefined hierarchy. To solve these problems, we propose an extended local hierarchical Naive Bayes classifier, where a binary classifier is built for each class in the hierarchy. The extension to conventional local approaches is that each classifier considers both the parent and child classes of the current class. We have evaluated the proposed approach on eight protein function and ten gene function hierarchical classification datasets. The proposed approach achieved somewhat better predictive accuracies than a global hierarchical Naive Bayes classifier.

Keywords: Hierarchical Classification, naive Bayes, Bioiformatics, Protein Function Prediction.

1 Introduction

Classification is a well-known data mining task, where the algorithm builds, from the training set, a classifier that is used to predict the class labels of instances in the test set. A very active research area in bioinformatics consists of using classification methods to predict protein and gene functions. Although several sequencing genome projects have generated the full genome sequence of many organisms in the last decades, the functions of many proteins and genes still remain unknown. This is because, in general, determining the functions of genes and proteins is much more difficult and time-consuming than finding out their sequences.

A popular approach for biologists to infer new protein/gene functions is to use techniques that perform a similarity search in a protein/gene database containing proteins/genes with known functions. Basically, these techniques compute the similarity between the sequence of a protein/gene with unknown function and the sequences of the proteins/genes in a database. Thus, the new protein/gene is assigned to the class of its most similar protein(s)/gene(s) in the database [1].

Nevertheless, similarity-based protein/gene function prediction methods have some limitations. First, it is known that proteins/genes with similar sequences

can have different functions [2]. Second, function prediction based only on sequence similarity does not consider many relevant biochemical properties of proteins/genes [3].

Aiming at solving these limitations, several works have proposed approaches that consist of inducing classification models from protein/gene data, where each protein/gene is represented by a set of attributes, and the new proteins/genes are classified by the induced model. This approach, which was adopted in this work, give us the opportunity to use a variety of classification methods for the protein/gene function prediction.

Most classification methods can deal only with flat classification problems, where there are no hierarchical relationships among the classes. However, in many problems the classes are naturally organized into hierarchies. Hierarchical classification problems are particularly common in the prediction of protein and gene functions, where the classes (functions) to be predicted are arranged in a tree or DAG (Direct Acyclic Graph) structure.

The prediction of protein and gene functions is challenging, mainly because there are usually hundreds or thousands of classes in the hierarchy and the class distribution is usually highly skewed, i.e., different class labels occur with very different frequencies. To simplify the problem, some works have just ignored the hierarchical class structure and addressed this problem as a traditional flat classification problem [4–6]. However, such works lose valuable information about parent-child class relationships, which is avoided by using a hierarchical classification method. One such method is described in [7], where the authors evaluated two hierarchical classification methods, based on a global and local version of a hierarchical Naive Bayes classifier - with the global version obtaining better predictive accuracy on eight protein function prediction datasets.

In this work, we propose an extended local hierarchical Naive Bayes classifier that (unlike a conventional local approach) exploits parent-child relationships between the classes in order to build a binary classifier for each class in the hierarchy. Then, the results of these binary classifiers are combined to produce the final classification for an instance. We evaluated our proposal on the same eight protein datasets used in [7] and on other ten gene function datasets, and compared it against the global-model approach proposed in [7].

The remaining of this paper is organized as follows. Section 2 presents an overview on hierarchical classification. In Section 3, we describe the hierarchical classifier proposed in this work. Section 4 presents the experimental setup and reports the results obtained in the comparative experiment. Finally, the conclusion and directions for future work are described in Section 5.

2 Hierarchical Classification

Hierarchical classification methods can be analyzed according to different aspects. The first one regards the type of hierarchical structure (tree or DAG) the method is able to deal with. This structure represents the relationships between the classes of the problem to be solved. Basically, in a tree each class is associated

with at most one parent class, while in a DAG a child class can have multiple parent classes.

The second aspect is how deep in the hierarchy the classification is done. A method can either perform mandatory leaf node predictions, where each instance must be assigned classes at leaf nodes in the class hierarchy; or non-mandatory (optional) leaf node predictions, where the most specific class assigned to an instance can be any (internal of leaf) class node in the hierarchy.

The third aspect refers to the number of different path labels in the hierarchy a method can assign an instance to. A method can be able to predict, to each instance, multiple paths of labels in the class hierarchy, or be restricted to predict just a single path of labels.

The fourth aspect concerns how the hierarchical structure is handled by the method. Three different approaches are presented in the literature: flat classification, which ignores the class hierarchy and performs predictions considering only the leaf node classes; local model approaches, when a set of local models are employed; and global model approaches, when a single classification model is built considering the class hierarchy as a whole during a single run of the classification algorithm. Since flat classification is out of the scope of this work, only local model approaches and global model approach are discussed next.

2.1 Local Classification Approach

In this approach, multiple classifiers are built, each one with a local view of the problem. Note that the class hierarchy is taken into account through a local perspective. Based on the different ways of using the local information, the classifiers can be grouped into three different categories [8]: local per node approach, local per parent node approach and local per level approach.

The local per node approach creates one binary classifier for each class node in the hierarchy (except the root node). Each classifier predicts whether or not an instance belongs to its corresponding class. The dashed rectangles in Figure 1(a) represent the classifiers. Note that this approach allows an instance to be assigned to classes in distinct branches in the hierarchy, which can lead to a class-membership inconsistency. To avoid that, several inconsistency removal methods are available [9–11].

In the local per parent node approach, a multi-class classifier is trained for each parent node in the hierarchy aiming at distinguishing between its child nodes. This approach is often used with a top-down prediction strategy when classifying new test instances. To illustrate this strategy, consider the hierarchy in Figure 1(b), where each dashed rectangle represents a classifier used to predict one of the child class nodes related to that classification node. Suppose a new test instance is assigned the class 1 by the root node classifier. Then, at the first hierarchy level, the classifier related to class node 1 will assign to this instance one of the child classes (1.1 or 1.2) of that node, and so on, until the instance is classified at the deepest appropriated level.

Finally, the local per level approach consists of training a multi-class classifier for each level of the class hierarchy. This is the hierarchical approach least used

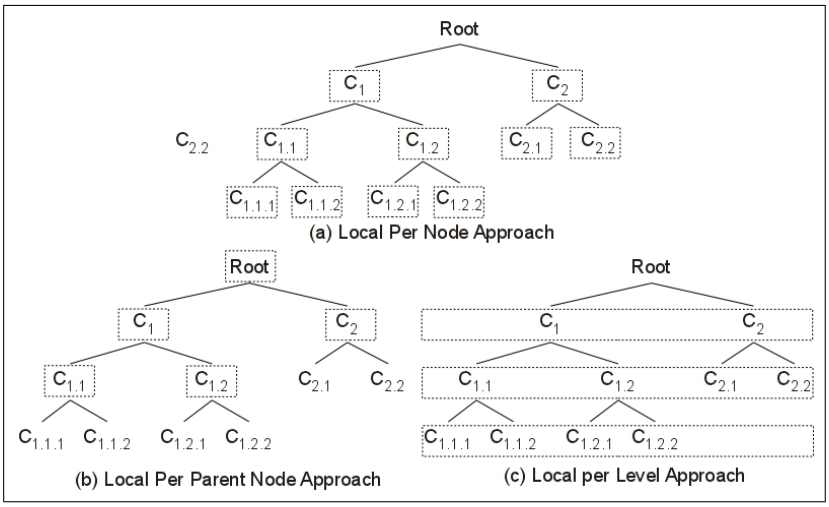

Fig. 1. Types of hierarchical classification approaches.

in the literature [12]. Its major disadvantage is to be prone to class-membership inconsistency. For example, using the class hierarchy shown in Figure 1(c), three classifiers would be trained, one for each hierarchy level (represented by dashed rectangles). Then, given a instance to be classified, it is possible to have the following predictions: class 1 at level 1, class 2.1 at level 2 and class 1.1.2 at level 3. Clearly, the predicted class 2.1 is not consistent with the classes 1 and 1.1.2. Hence, this kind of approach requires a post-processing procedure to correct inconsistent predictions.

2.2 Global Classification Approach

Instead of creating a set of classifiers, the global approach involves the training of a single classifier taking into account the class hierarchy as a whole. Then, given a new instance to be classified, the induced classifier is able to assign it a class from any level of the hierarchy.

While the local approach with the top-down class prediction strategy has the disadvantage of propagating a classification mistake at a given level of the hierarchy through all its deeper levels, the global approach avoids that drawback by performing the classification in a single step using a single classifier.

It is worth noting that the global approach lacks the modular nature of the local approach, i.e., the characteristic of dividing the training phase in different processes, each of them considering part of the class hierarchy. Therefore, the single classifier built by the global approach tends to be more complex than each individual classifier produced by local approaches. However, this modular nature of the local approach does not imply that they will have better predictive accuracy than global approaches.

In this work we propose an extended local hierarchical Naive Bayes classifier based on the local per node approach (as described in the next section) and compare it against a global classification approach.

3 The Proposed Hierarchical Classifier

The proposed classifier deals with hierarchical classification problems where the classes to be predicted are disposed in a tree-based structure, in the scenarios of mandatory leaf node prediction and prediction of a single path in the class hierarchy. The main goal is to exploit parent-child relationships between the classes when building a binary classifier for each class in the hierarchy. Then, the predictions made by the set of binary classifiers are combined in order to produce a consistent prediction.

The proposed classifier, named Extended Local Hierarchical Naive Bayes (ELHNB), is based on the local per node approach, creating one binary classifier for each node of the class hierarchy. The training of each classifier considers not only the local information related to each classification node as usual, but also information about the relationships between each class node and its parent and child nodes – where the latter type of information is the proposed extension. Note that, since our method is based on Naive Bayes, we make the assumption of class conditional independence, that is, the attributes are conditionally independent of one another given the class attribute.

Let $D = \{\boldsymbol{d^1}, \ldots, \boldsymbol{d^t}\}$ be a set of training instances. Each instance $\mathbf{d^j}$, $j = 1, \ldots, t$, is represented by its attribute vector $\boldsymbol{X^j} = \{x_1^j, x_2^j, \ldots, x_n^j\}$ and is associated with a binary class vector $\boldsymbol{C^j} = \{c_1^j, c_2^j, \ldots, c_m^j\}$, where n is the number of predictor attributes and m is the number of classes in the hierarchy. Each c_i^j is assigned the value 1 if the instance d^j is associated with the class C_i, and 0 otherwise.

We train a Naive Bayes classifier to predict the class label vector $\boldsymbol{C} = \{c_1, c_2, \ldots, c_m\}$ for a new instance $\boldsymbol{X} = \{x_1, x_2, \ldots, x_n\}$, i.e., to learn a predictor $f : \boldsymbol{X} \mapsto [c_1, c_2, \ldots, c_m]$. The proposed approach has two phases. The first one is a local classification phase, where we train a binary classifier for each class. Each binary classifier performs a probabilistic classification for each class C_i, i.e., it computes $P(C_i = 1|\boldsymbol{X})$ and $P(C_i = 0|\boldsymbol{X})$. In the second phase we use the probabilities $P(C_i|\boldsymbol{X})$ calculated in the first phase to generate a consistent class vector prediction for the instance $\boldsymbol{X}$.

Recall that the classes are structured into a tree. Given a class C_i, the set of nodes formed by its parent and child nodes, termed neighbors of C_i, is represented by $\boldsymbol{N_i}$. The labels of the nodes (classes) contained in $\boldsymbol{N_i}$ are coded through a vector $\boldsymbol{Y_i} = \{y_1, y_2, \ldots, y_k\} \in \{0,1\}^{k_i}$, where k_i is the number of neighbors of C_i. In the local classification phase, for each C_i, we compute $P(C_i = c_i)$, $c_i \in \{0, 1\}$, taking into account the relationships between the class C_i and its parent and child nodes in the hierarchy. It can be determined by computing the

following marginal probability:

$$P(C_i = c_i) = \sum_{\boldsymbol{Y_i} \in \{0,1\}^{k_i}} P(C_i = c_i|\boldsymbol{Y_i}) \times P(\boldsymbol{Y_i}) \tag{1}$$

Thus, $P(C_i = c_i|\boldsymbol{X})$ is obtained conditioning Equation 1 on instance $\boldsymbol{X}$ as follows:

$$P(C_i = c_i|\boldsymbol{X}) = \sum_{\boldsymbol{Y_i} \in \{0,1\}^{k_i}} P(C_i = c_i|\boldsymbol{X}, \boldsymbol{Y_i}) \times P(\boldsymbol{Y_i}|\boldsymbol{X}) \tag{2}$$

Applying Bayes' theorem on each term of the product in Equation 2:

$$P(C_i = c_i|\boldsymbol{X}, \boldsymbol{Y_i}) = \frac{P(\boldsymbol{X}|C_i = c_i, \boldsymbol{Y_i}) \times P(C_i = c_i|\boldsymbol{Y_i})}{\sum_{c_i \in \{0,1\}} P(\boldsymbol{X}|C_i = c_i, \boldsymbol{Y_i}) \times P(C_i = c_i|\boldsymbol{Y_i})} \tag{3}$$

and

$$P(\boldsymbol{Y_i}|\boldsymbol{X}) = \frac{P(\boldsymbol{X}|\boldsymbol{Y_i}) \times P(\boldsymbol{Y_i})}{\sum_{\boldsymbol{Y_i} \in \{0,1\}^{k_i}} P(\boldsymbol{X}|\boldsymbol{Y_i}) \times P(\boldsymbol{Y_i})} \tag{4}$$

Substituting Equations 3 and 4 into Equation 2, we have:

$$P(C_i = c_i|\boldsymbol{X}) = \sum_{\boldsymbol{Y_i} \in \{0,1\}^{k_i}} \left(\frac{P(\boldsymbol{X}|C_i = c_i, \boldsymbol{Y_i}) \times P(C_i = c_i|\boldsymbol{Y_i})}{\sum_{c_i \in \{0,1\}} P(\boldsymbol{X}|C_i = c_i, \boldsymbol{Y_i}) \times P(C_i = c_i|\boldsymbol{Y_i})} \times \frac{P(\boldsymbol{X}|\boldsymbol{Y_i}) \times P(\boldsymbol{Y_i})}{\sum_{\boldsymbol{Y_i} \in \{0,1\}^{k_i}} P(\boldsymbol{X}|\boldsymbol{Y_i}) \times P(\boldsymbol{Y_i})} \right) \tag{5}$$

As $\sum_{\boldsymbol{Y_i} \in \{0,1\}^{k_i}} P(\boldsymbol{X}|\boldsymbol{Y_i}) \times P(\boldsymbol{Y_i}) = P(\boldsymbol{X})$, we can rewrite Equation 5 as:

$$P(C_i = c_i|\boldsymbol{X}) = \sum_{\boldsymbol{Y_i} \in \{0,1\}^{k_i}} \left(\frac{P(\boldsymbol{X}|C_i = c_i, \boldsymbol{Y_i}) \times P(C_i = c_i|\boldsymbol{Y_i})}{\sum_{c_i \in \{0,1\}} P(\boldsymbol{X}|C_i = c_i, \boldsymbol{Y_i}) \times P(C_i = c_i|\boldsymbol{Y_i})} \times \frac{P(\boldsymbol{X}|\boldsymbol{Y_i}) \times P(\boldsymbol{Y_i})}{P(\boldsymbol{X})} \right) \tag{6}$$

Given that $P(\boldsymbol{X})$ is constant for all $\boldsymbol{Y_i}$ vector configurations, rearranging Equation 6 we get:

$$P(C_i = c_i|\boldsymbol{X}) = \frac{\sum_{\boldsymbol{Y_i} \in \{0,1\}^k} \left(\frac{P(\boldsymbol{X}|C_i = c_i, \boldsymbol{Y_i}) \times P(C_i = c_i|\boldsymbol{Y_i}) \times P(\boldsymbol{X}|\boldsymbol{Y_i}) \times P(\boldsymbol{Y_i})}{\sum_{c_i \in \{0,1\}} P(\boldsymbol{X}|C_i = c_i, \boldsymbol{Y_i}) \times P(C_i = c_i|\boldsymbol{Y_i})} \right)}{P(\boldsymbol{X})} \tag{7}$$

In Equation 7, aiming at reducing the number of parameters in evaluating $P(\boldsymbol{X}|C_i = c_i, \boldsymbol{Y_i})$ and $P(\boldsymbol{X}|\boldsymbol{Y_i})$, we use the Naive Bayes assumption that there are no dependence relationships among the attributes given the class. Then, these probabilities are computed as $P(\boldsymbol{X}|C_i = c_i, \boldsymbol{Y_i}) = \prod_{k=1}^{n} P(x_k|C_i = c_i, \boldsymbol{Y_i})$ and $P(\boldsymbol{X}|\boldsymbol{Y_i}) = \prod_{k=1}^{n} P(x_k|\boldsymbol{Y_i})$.

In the second phase, named global classification phase, we enforce hierarchical consistency of class labels using the probabilities $P(C_i = c_i|\boldsymbol{X})$ computed in the first phase. In order to obtain a consistent classification, for each possible path p in the hierarchy from *Root* node to node i, we compute the geometric average of probabilities $P(C_i = 1|\boldsymbol{X})$ along the path as follows:

$$GA_p = \sqrt[|L_p|]{\prod_{C_i \in L_p} P(C_i = 1|\boldsymbol{X})}, \tag{8}$$

where L_p is the set of classes in the path p.

As final solution, the instance $\boldsymbol{X}$ is assigned to the class vector $\boldsymbol{C}$ where the classes C_i contained in the path with the highest GA are set to 1 and the remaining to 0.

4 Computational Experiments

4.1 Baseline Method

Since we are proposing an extended local hierarchical Naive Bayes classifier, we use as a baseline method the global hierarchical Naive Bayes classifier proposed in [7], which achieved promising predictive performance when evaluated on eight protein datasets and outperformed a conventional local per parent node hierarchical classifier. In this conventional local hierarchical classifier, during the training phase, for each non-leaf node, a Naive Bayes multi-class classifier is trained to discriminate among the child class nodes of the classifier's corresponding node. Next, to implement the test phase, the top-down class prediction strategy is adopted.

The hierarchical classifier in [7] is an extension of the flat classification algorithm Naive Bayes to deal with hierarchical classification problems.

Given a new instance $X = \{x_1, x_2, \ldots, x_n\}$ to be classified, where each x_k refers to the value of attribute A_k, the flat Naive Bayes classifier simply assigns to it the class C_i associated with the maximum value of the posterior probability calculated as $P(C_i|X) \propto \prod_{k=1}^{n} P(x_k|C_i) \times P(C_i)$.

To explain how the hierarchical Naive Bayes [7] works, consider a tree-based hierarchy containing these class nodes: C_1, C_2, $C_{1.1}$, $C_{1.2}$, $C_{2.1}$ and $C_{2.2}$. To classify a new instance, the prior probabilities $P(C_1)$, $P(C_2)$, $P(C_{1.1})$, $P(C_{1.2})$, $P(C_{2.1})$ and $P(C_{2.2})$, and the likelihoods $P(x_k|C_1)$, $P(x_k|C_2)$, $P(x_k|C_{1.1})$, $P(x_k|C_{1.2})$, $P(x_k|C_{2.1})$ and $P(x_k|C_{2.2})$ are computed taking into account the class hierarchy, as follows. More precisely, to compute the prior probabilities and likelihoods during the training phase, the method takes into account that any instance which belongs to class C_i also belongs to all its ancestor classes. For example, if a training instance belongs to class $C_{1.2}$, that instance will be taken into account to compute the prior probabilities of both that class ($C_{1.2}$) and its ancestor classes (in this case, C_1). In addition, the attribute values of a training instance will be taken into account to compute the likelihoods associated with that instance's class and all its ancestor classes. These modification make the global hierarchical Naive Bayes able to predict classes at any level of the class hierarchy. For more details about this method, see [7].

4.2 Datasets

Experiments were conducted by running both the proposed and the baseline methods on 18 bioinformatics datasets, where eight are protein function and ten are gene function hierarchical classification datasets. As these datasets were obtained from different sources, we organized them into two groups.

Group A contains eight protein function datasets, referring to two different protein families: Enzymes and G-Protein-Coupled Receptors (GPCRs). Enzymes are proteins that catalyze chemical reactions [13] while GPCRs are transmembrane proteins that are the targets of many medical drugs [14]. We used four enzyme datasets (whose names start with EC – Enzyme Commission) and four GPCR datasets, where the predictor attributes correspond to protein properties and the classes to be predicted are hierarchical protein functions. Most predictor attributes are binary, indicating whether or not a protein signature (or motif) is present in a protein, but there are also two numeric attributes: the amino acid sequence length and molecular weight. The names of the datasets are also related to the type of motifs used: Interpro Entries, FingerPrints, Prosite Patterns and Pfam. These datasets[3] have also been used in previous hierarchical classification works [15], [16] and [7].

Group B contains ten gene function datasets, referring to the yeast genome. The predictor attributes include five types of bioinformatics data: secondary structure, phenotype, homology, sequence statistics, and expression. The classes to be predicted are taken from FunCat[4], a scheme for classifying the function of gene products developed by MIPS [17]. These datasets[5], initially presented in [18] and after updated and used in [19] were multi-label data, i.e., each instance was associated with one or more class paths in the hierarchy. Since in this work we are dealing with a single path label scenario, these datasets were converted into single label data by randomly choosing one class path for each instance.

Before running the classification algorithms, all datasets were preprocessed as follows: (a) All numeric attributes were converted into discrete ones by using an unsupervised discretization algorithm based on equal frequency binning (using 20 bins); (b) Every class with fewer than 10 instances was merged with its parent class. This process was repeated until every class in the hierarchy had at least 10 instances. If during this process the most specific class of an instance became the Root class, then that instance was removed; (c) Since in this work we are dealing with a mandatory leaf node prediction problem, after the previous step (b), we removed from the datasets all instances whose most specific class was not a leaf class node. Table 1 presents the main characteristics of the datasets after these pre-processing steps. This table shows, for each dataset, its number of attributes, number of instances and number of classes at each hierarchy level (1st/2nd/3rd/...). The pre-processed datasets used in our experiments are available at: http://www.decom.ufop.br/luiz/resources/.

4.3 Predictive Accuracy Evaluation Metrics

In order to evaluate the predictive accuracy of the hierarchical classifiers, we used the hierarchical F-measure, which is an adaptation of the flat F-measure tailored for hierarchical classification problems [20]. The hierarchical F-measure is computed as $hF = \frac{2 \times hP \times hR}{hP + hR}$, where hP and hR stand for hierarchical Precision

[3] https://sites.google.com/site/carlossillajr/resources
[4] http://mips.helmholtz-muenchen.de/proj/funcatDB/
[5] http://dtai.cs.kuleuven.be/clus/hmcdatasets/

Table 1. Characteristics of the Datasets

Group	Datasets	# Attributes	# Instances	# Classes/Level
A	GPCR-Pfam	75	6,524	12/52/79/49
	GPCR-Prosite	129	5,728	9/50/79/49
	GPCR-Prints	283	4,880	8/46/76/49
	GPCR-Interpro	450	6,935	12/54/82/50
	EC-Prints	382	11,048	6/45/92/208
	EC-Prosite	585	11,328	6/42/89/187
	EC-Pfam	708	11,057	6/41/96/190
	EC-Interpro	1,216	11,101	6/41/96/187
B	CellCycle	78	2,486	16/47/69/32/8
	Church	28	2,499	16/49/67/34/6
	Derisi	64	2,497	16/48/70/31/7
	Eisen	80	1,641	16/43/55/23/2
	Expr	552	2,554	16/49/68/28/5
	Gasch1	174	2,595	16/48/71/32/7
	Gash2	53	2,631	17/49/68/34/6
	Phenotype	70	1,023	15/43/40/15/1
	Sequence	479	2,689	17/48/65/29/5
	SPO	81	2,463	16/48/68/31/8

and hierarchical Recall, respectively. Considering that P_i is the set composed by the most specific class predicted for a test instance i and all its ancestor classes and T_i is the set composed by the true most specific class for a test instance i and all its ancestor classes, the hP and hR were defined in [20] as follows: $hP = \frac{\sum_i |P_i \bigcap T_i|}{\sum_i |P_i|}$ and $hR = \frac{\sum_i |P_i \bigcap T_i|}{\sum_i |T_i|}$.

Although these measures are recommended to evaluate hierarchical classification scenarios [8], there is a situation where their application faces a problem. Basically, hierarchical precision and hierarchical recall are measures related to the concepts of specialization and generalization errors, respectively. To illustrate these concepts, let us consider the following examples. Let C_1 be the most specific predicted class for an instance whose true most specific known class is $C_{1.3}$. In this case we have a generalization error, since the most specific class predicted is more generic than the true most specific known class for that instance. This generalization error is captured by the hierarchical recall measure, which for this example is $hR = 1/2$. Observe that in this case the hierarchical precision assumes the maximum value, i.e, $hP = 1$. On the other hand, if $C_{1.3}$ is the most specific predicted class for an instance whose true known class is C_1, we have a specialization error, as the predicted class is more specific than the true known class for that instance. Now, the hierarchical precision measure indicates this error ($hP = 1/2$), whilst the hierarchical recall measure assumes the maximum value $hR = 1$.

At first glance, hierarchical precision and hierarchical recall measures seem to penalize specialization and generalization errors appropriately. However, con-

sidering an over-specialization as an error (penalized by hP) can be unfair in some kinds of applications, given that, if the true most specific known class for an instance is a more generic class like C_1, this does not mean that the prediction of the more specific class $C_{1.3}$ is an error. This may just mean that at present the more specific class of that instance is unknown, only its more generic class is currently known. Indeed, this kind of situation is relatively common in protein and gene function prediction, where more specific functional classes are often unknown and will be discovered later, with continuing advances in biological experiments that determine gene and protein functions.

Hence, we modified the definition of hP in order not to penalize over-specialized predictions. It is important to mention that even in mandatory leaf node prediction problems (the scenario considered here) over-specialized predictions can be made, since we can have leaf node classes at different levels in the hierarchy. Then, the adapted hP measure used in this work (which was used to measure the predictive accuracy of both hierarchical classification methods in our experiments) is defined as $hP = \frac{\sum_i |P_i \bigcap T_i|}{\sum_i min(|P_i|,|T_i|)}$, where $min(|P_i|, |T_i|)$ is the minimum value between $|P_i|$ and $|T_i|$.

4.4 Computational Results

As mentioned earlier, the Extended Local Hierarchical Naive Bayes classifier (ELHNB) was compared with the global-model Naive Bayes approach (GMNB) proposed in [7] on 18 bioinformatics datasets.

The performance of the hierarchical classifiers was measured by using the 10-fold cross validation [21] and the hierarchical F-measure (described in Section 4.3). The same ten folds in each dataset were used to evaluate the classifiers. In addition, for each dataset, in order to determine if there is a statistically significant difference between the hierarchical F-measure of the two hierarchical classifiers being compared, we used the Wilcoxon's Signed-Rank Test (two-sided test) as recommended by [22].

The results comparing the baseline GMNB with the proposed ELHNB are shown in Table 2. For each dataset, the third and fourth columns present the hierarchical F-measure values (hF) obtained by 10-fold cross validation (with the standard error in parentheses). The largest hierarchical F-measure value (hF) between those obtained by the two methods is in bold font. The last column presents the name of the method that achieved the best hF value when there is a statistically significant difference between the hF values of the two classifiers, or the symbol (–) to indicate that the difference between the hF values was not statistically significant.

In the eight datasets of Group A, the proposed ELHNB obtained significantly better results than GMNB for four datasets. In the remaining four datasets there was no statistically significant difference between the hF values of the two classifiers. In Group B, ELHNB outperformed GMNB in one dataset and GMNB outperformed ELHNB in another one. In the remaining datasets of this group, the difference of hF values between the classifiers was not statistically significant.

Table 2. Results of Comparative Experiment

Group	Data Sets	GMNB hF(std. error)	ELHNB hF(std. error)	Result of Statistical Test ($\alpha = 0.05$)
A	GPCR-Pfam	62.48 (0.31)	**62.99** (0.32)	–
	GPCR-Prosite	**61.58** (0.54)	61.44 (0.44)	–
	GPCR-Prints	79.66 (0.56)	**79.96** (0.51)	–
	GPCR-Interpro	80.44 (0.31)	**80.64** (0.36)	–
	EC-Prints	93.97 (0.24)	**94.99** (0.15)	**ELHNB**
	EC-Prosite	94.46 (0.08)	**96.84** (0.06)	**ELHNB**
	EC-Pfam	94.81 (0.14)	**95.42** (0.14)	**ELHNB**
	EC-Interpro	95.71 (0.12)	**96.33** (0.15)	**ELHNB**
B	CellCycle	12.45 (0.54)	**12.83** (0.57)	–
	Church	**10.65** (0.46)	10.61 (0.44)	–
	Derisi	10.42 (0.31)	**10.88** (0.58)	–
	Eisen	15.52 (0.87)	**16.52** (0.81)	**ELHNB**
	Expr	14.09 (0.73)	**14.63** (0.5)	–
	Gasch1	**15.31** (0.55)	14.20 (0.46)	**GMNB**
	Gash2	**15.70** (0.51)	15.42 (0.38)	–
	Phenotype	**8.08** (0.37)	7.22 (0.48)	–
	Sequence	**16.20** (0.79)	15.80 (0.76)	–
	SPO	**10.18** (0.52)	9.66 (0.59)	–

Overall, the proposed ELHNB reached results statistically equivalent or better than GMNB in 17 out of 18 datasets.

5 Conclusion

In this work, we proposed an extended local hierarchical Naive Bayes classifier based on a local per node approach, where a binary classifier is built for each class node in the hierarchy by exploiting the relationships between that class node and its parent and child nodes. The term "extended" is used to indicate that the proposed classifier extends the conventional local hierarchical approach by training each classifier with classes predicted for that classifier's neighbor (parent and child) nodes.

Different scenarios can be considered when dealing with hierarchical classification problems. In this paper we dealt with mandatory leaf node prediction problems, where the algorithm has to predict one of the leaf class nodes for each test instance. In addition, we focused on problems in which the classes to be predicted are disposed in a tree-based hierarchy and each data instance has a class label associated with a single path in this class hierarchy.

The evaluation of the proposed classifier was conducted on 18 bioinformatics datasets, where eight are protein function and ten are gene function hierarchical classification datasets.

Given the Bayesian nature of the proposed classifier, aiming at comparing it against a method of the same broad type, we used as a baseline method the global-model Naive Bayes approach proposed in [7]. In our experiments the proposed ELHNB classifier achieved predictive performance (measured by hierarchical F-measure) significantly better than the baseline GMNB method in 5 datasets, was significantly worse in only 1 dataset, and statistically equivalent in the remaining ones. Therefore, we conclude that the proposed extended local hierarchical Naive Bayes classifier has shown good predictive performance in the bioinformatics datasets used in this work, being somewhat more accurate than a global hierarchical classifier.

As future work, we intend to evaluate the ELHNB method in other hierarchical scenarios and compare it against other hierarchical classification approaches.

Acknowledgements

The first author is financially supported by CNPq - a Brazilian research-support agency (processes number 202120/2011-2 and 307711/2010-2). The authors also would like to thank the anonymous reviewers for their helpful comments.

References

1. Sleator, R.D., Walsh, P.: An overview of in silico protein function prediction. Archives of Microbiology **192**(3) (2010) 151–155
2. Gerlt, J.A., Babbitt, P.C.: Can sequence determine function? **1** (2000)
3. Syed, U., Yona, G.: Using a mixture of probabilistic decision trees for direct prediction of protein function. In: Proceedings of the seventh annual international conference on Research in computational molecular biology. RECOMB '03, New York, NY, USA, ACM (2003) 289–300
4. Pavlidis, P., Cai, J., Weston, J., Noble, W.S.: Learning gene functional classifications from multiple data types. Journal of Computational Biology **9** (2002) 401–411
5. Suhai, S., Glatting, K.H., Eils, R., Schubert, F., Moormann, J., König, R., Vinayagam, A.: Applying support vector machines for gene ontology based gene function prediction. BMC Bioinformatics **5** (2004)
6. Jung, J., Thon, M.R.: Automatic annotation of protein functional class from sparse and imbalanced data sets. In: Proc. of the First International Conference on Data Mining and Bioinformatics. (2006) 65–77
7. Silla Jr., C.N., Freitas, A.A.: A global-model naive bayes approach to the hierarchical prediction of protein functions. In: Proc. of the 2009 Ninth IEEE International Conference on Data Mining, IEEE Computer Society (2009) 992–997
8. Silla Jr., C.N., Freitas, A.A.: A survey of hierarchical classification across different application domains. Data Mining and Knowledge Discovery **22**(1-2) (2011) 31–72
9. Wu, F., 0002, J.Z., Honavar, V.: Learning classifiers using hierarchically structured class taxonomies. In: Proc. of the International Symposium on Abstraction, Reformulation and Approximation. (2005) 313–320
10. Barutcuoglu, Z., DeCoro, C.: Hierarchical shape classification using bayesian aggregation. In: Proc. of the IEEE International Conference on Shape Modeling and Applications 2006. SMI '06 (2006) 44

11. Valentini, G.: True path rule hierarchical ensembles for genome-wide gene function prediction. IEEE/ACM Transactions on Computational Biology and Bioinformatics **8**(3) (2011) 832–847
12. Silla Jr., C.N.: Novel Approaches for Hierarchical Classification with Case Studies in Protein Function Prediction. PhD thesis, University of Kent (2011)
13. Grisham, C.M., Garrett, R.H.: Biochemistry. Saunders College Publishers, Philadelphia (1999)
14. Venkatakrishnan, A.J., Deupi, X., Lebon, G., Tate, C.G., Schertler, G.F., Babu, M.M.: Molecular signatures of g-protein-coupled receptors. Nature **494** (2013) 185–194
15. Costa, E.P., Lorena, A.C., Carvalho, A.C.P.L.F., Freitas, A.A., Holden, N.: Comparing several approaches for hierarchical classification of proteins with decision trees. In: Proc. of the 2nd Brazilian Conference on Advances in Bioinformatics and Computational Biology, Lecture Notes in Bioinformatics 4643. BSB'07, Angra dos Reis, Brazil, Springer-Verlag (2007) 126–137
16. Holden, N., Freitas, A.A.: Improving the performance of hierarchical classification with swarm intelligence. In: Proc. of the 6th European Conference on Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics. (2008) 48–60
17. Mewes, H.W., Heumann, K., Kaps, A., Mayer, K.F.X., Pfeiffer, F., Stocker, S., Frishman, D.: Mips: a database for genomes and protein sequences. Nucleic Acids Research **27**(1) (1999) 44–48
18. Clare, A., King, R.D.: Predicting gene function in saccharomyces cerevisiae. In: Proc. of the European Conference on Computational Biology. (2003) 42–49
19. Vens, C., Struyf, J., Schietgat, L., Džeroski, S., Blockeel, H.: Decision trees for hierarchical multi-label classification. Machine Learning **73**(2) (2008) 185–214
20. Kiritchenko, S., Matwin, S., Famili, A.F.: Functional annotation of genes using hierarchical text categorization. In: Proc. of the BioLINK SIG: Linking Literature, Information and Knowledge for Biology. (2005)
21. Han, J., Kamber, M., Pei, J.: Data Mining: Concepts and Techniques. 3rd edn. Morgan Kaufmann Publishers, USA (2011)
22. Japkowicz, N., Shah, M.: Evaluating Learning Algorithms: A Classification Perspective. Cambridge University Press, New York, USA (2011)