An Efficient Algorithm for Hierarchical Classification of Protein and Gene Functions

Fabio Fabris and Alex A. Freitas
School of Computing, University of Kent
Canterbury, Kent, CT2 7NF, UK
{ff79,A.A.Freitas}@kent.ac.uk

***Abstract*—The classification of protein and gene functions is a complex problem that is becoming more relevant as the number of sequenced genes and proteins increases. This work presents a modified version of the Extended Local Hierarchical Naive Bayes algorithm, which exploits the requirements of the original algorithm (single-path, mandatory-leaf-prediction hierarchical classification problems in tree-structured class hierarchies) to greatly improve classification run-time. We show that, considering 18 hierarchical classification datasets, the modified algorithm yields equivalent predictive performance and significantly improves run-time in the training and prediction phases.**

I. Introduction

In the recent years, the cost of extracting genomic and proteomic data from organisms has decreased significantly, to the extent that researchers now have free access to vast public collections of biological data. Generally speaking, these datasets comprise the sequence information (amino-acids, base-pairs, and other secondary information extracted from gene or protein sequences) and the classification of the biological processes that gene or protein is involved in a curated ontology. An example of such ontology is the Gene Ontology [7] and an example of a protein dataset is the Universal Protein Resource [8].

To make sense of this ever-increasing flow of complex information, data mining techniques such as clustering, classification and data visualization are, more than ever, required to assist biologists. In this paper we shall focus on the classification task of data mining. Broadly speaking, classification is the computational task of deriving a function $\mathcal{F} : \mathbf{x} \rightarrow C$ from a set of instances in the form $(\mathbf{x_j}, \mathbf{c_j})$, where $\mathbf{x_j}$ are the predictive attributes of the j-th instance and $\mathbf{c_j}$ is (are) the classe(s) of the j-th instance. We call the derivation of $\mathcal{F}$ as the "training phase". After the training, we use $\mathcal{F}$ to predict the classes of a set of previously unseen instances given only their predictive attributes (prediction phase). Because there are many approaches to build different $\mathcal{F}$ functions and none is better than all the others in every occasion, it is wise to estimate the predictive accuracy using some measure, such as the F-Measure [6], to select the best classifier for a given classification task.

The objective of using classification algorithms in this work is to infer protein functions using only attributes describing the protein and gene sequence information. That is, to build a function $\mathcal{F}$ that maps the gene and protein sequences to classes, namely, their functional classes, their location, and the biological processes that they are involved in. The inferred classes may be used as a starting point to select the most promising wet-lab experiments to be performed, which are much more time-consuming and expensive than their computational counterparts.

However, unlike the "flat" classes of standard classification problems, ontologies that define the classification of genes and proteins are normally organized as trees or Directed Acyclic Graphs (DAGs). This complicates the use of out-of-shelf classification algorithm and justifies the development of specialized algorithms for hierarchical classification [5].

Additionally, due to the ever-increasing size of biological datasets [10], it is important to develop efficient algorithms that scale up well both in the training and prediction phase. This factor has been neglected in the literature, where works containing complexity and/or run-time analysis of hierarchical classification algorithm are uncommon. To help fill this niche, this work proposes a modified version of the Extended Local Hierarchical Naive Bayes (ELHNB) [9]. The modification (M-ELHNB) speeds up the training and prediction phases of the algorithm, maintaining an equivalent predictive performance.

The remainder of this work is organized as follows: Section II formally defines the hierarchical classification problem and the sub-problem that we shall focus on. Section III presents the original ELHNB classifier and our proposed modified version. In Section IV we present and discuss the experimental results and finally, in Section V we draw our conclusion and lay ground for future work.

II. Hierarchical Classification

Hierarchical Classification, sometimes called Structured Classification, is a special type of classification problem in which the classes of the instances have a pre-established taxonomy, generally specified as a DAG or a tree. We shall consider that this taxonomy defines an "Is-a" relationship between classes, that is, if an instance is classified as c_a, it is also implicitly classified as all the classes in the set $Anc(c_a)$, where $Anc(c)$ is defined as the set of all ancestors of class c.

Following the notation proposed in [5], hierarchical classification problems may be described by the tuple $\langle \Upsilon, \Psi, \Phi \rangle$, where $\Upsilon \in \{T, DAG\}$ defines the arrangement of the class taxonomy (Tree or DAG, respectively), $\Psi \in \{MPL, SPL\}$ indicates, respectively, whether at least one instance has Multiple Paths of Labels (MPL) or every instance has only a Single Path of Labels (SPL) in the class taxonomy. This distinction is analogous to single-label and multi-label classification problems in "flat" classification. $\Phi \in \{PD, FD\}$ dictates if at least one instance has Partial Depth (PD) labeling or every instance

has Full Depth (FD) labeling. FD labeling requires the most specific class of all instances to be a leaf node, while in PD labeling an instance may have a non-leaf node as its most specific class.

Hierarchical classification algorithms may be classified in two broad types: global or local. Local hierarchical classification algorithms train several traditional (flat) classification algorithms using some approach to transform the hierarchical classification problem to a flat one during the training phase and then recover the hierarchical structure from the flat classifications during the prediction phase. On the other hand, Global approaches build a single classification model for the whole hierarchy and do not require any problem transformation.

We shall focus on the Extended Local Hierarchical Naive Bayes (ELHNB) classification algorithm, recently proposed in [9]. In that work, a Naive Bayes (NB) local classifier was trained for each class considering the classes of the neighbouring nodes (parent and children) in the class hierarchy as extended features. In the prediction phase, the algorithm marginalises out the extended features by summing up the probabilities of all possible combinations of neighbouring classes. This algorithm is classified as "extended local" because although they use local NB classifiers, the class hierarchy is taken into consideration by using the extended features.

III. Extended Local Hierarchical Naive Bayes

Applying traditional classifiers in hierarchical classification problems is not straight-forward. Although the literature has multiple works on the subject of applying traditional algorithms in the hierarchical classification setting [5], algorithms that explicitly take the class hierarchy into account have the potential of improving the classification performance, reducing model size, and improving the interpretability of the classifier [1].

For these reasons, one of the successful efforts towards creating specific classification algorithms for hierarchical classification is the work of [9], which is an algorithm specially tailored for hierarchical classification that takes into account the class hierarchy explicitly. This algorithm, however, has a specially lengthy prediction phase due to the necessity of various probability calculations. We propose a modification of the algorithm that significantly improves its run time, specially in the prediction phase.

A. Original Algorithm

The ELHNB algorithm, specific for $\langle T, SPL, FD\rangle$ hierarchical classification problems, estimates the probability of an instance belonging to each class C_i given its features $\mathbf{x}$ using eq. (1) (readers interested in the derivation may refer to the original paper.) Eq. (1) is used for each class $C_i, 1 \leq i \leq N$, where N is the number of classes in the hierarchy.

$$P(C_i = c_i|\mathbf{x}) = \frac{1}{P(\mathbf{x})} \times \sum_{\mathbf{y_i} \in \{0,1\}^{k(i)}} \left(\frac{P(\mathbf{x}|C_i = c_i, \mathbf{y_i}) P(C_i = c_i|\mathbf{y_i}) P(\mathbf{x}|\mathbf{y_i}) P(\mathbf{y_i})}{\sum_{c' \in \{c_i, \tilde{c_i}\}} P(\mathbf{x}|C_i = c', \mathbf{y_i}) P(C_i = c'|\mathbf{y_i})} \right) \quad (1)$$

Where $k(i)$ represents the number of neighbours of the i-th class (the set containing the children and parent of the i-th class node, given by the hierarchy), $\mathbf{y_i}$ is a binary vector that iterates over all possible classifications of the neighbourhood of the i-th class, C_i is a random variable that may take the values c_i and $\tilde{c_i}$, $C_i = c_i$ is the event of the current instance being classified as belonging to the i-th class and $C_i = \tilde{c_i}$ the event of the current instance not being classified as belonging to the i-th class.

To estimate $P(\mathbf{x}|C_i, \mathbf{y_i})$ and $P(\mathbf{x}|\mathbf{y_i})$, the authors use the Naive Bayes assumption that the predictive attributes are independent given the class, reducing the estimation expressions to $P(\mathbf{x}|C_i, \mathbf{y_i}) = \prod_{k=1}^{n} P(x_k|C_i, \mathbf{y_i})$ and $P(\mathbf{x}|\mathbf{y_i}) = \prod_{k=1}^{n} P(x_k|\mathbf{y_i})$. The probabilities $P(C_i|\mathbf{y_i})$ and $P(\mathbf{y_i})$ may be estimated by simple counting and $P(\mathbf{x})$ by using the fact that $P(C_i = c_i|\mathbf{x}) + P(C_i = \tilde{c_i}|\mathbf{x}) = 1$.

The probabilities calculated by the algorithm are not guaranteed to be consistent, that is, the probability of an instance x_k belonging to class i, $P(C_i = c_i|\mathbf{x})$, may be higher than the probability the instance belonging to the parent of i, which is inconsistent to the fact that if an instance belongs to class i it implicitly belongs to the parent of i. Therefore, to tackle this problem, the authors calculate the geometric average of the probabilities of all possible paths from the root to the leaves and choose the path with greatest probability as the final classification, avoiding the inconsistency problem.

The overall time complexity of the algorithm's training phase considering the number of probability distributions that need to be estimated is $O(S_{mean} \times N)$, where S_{mean} is the mean size of the neighborhood of all nodes. Naturally, the complexity of the prediction phase considering the number of evaluations of probability functions is also $O(S_{mean} \times N)$.

B. Modified Algorithm

We shall demonstrate that for problems classified as $\langle T, SPL, FD\rangle$, the training phase of the Modified Extended Local Hierarchical Naive Bayes (M-ELHNB) algorithm may be executed faster by reducing the number of probability distributions that must be estimated. Likewise, we shall demonstrate that the prediction phase (the estimation of $P(C_i = c_i|\mathbf{x})$) for a given class may be executed in constant time regarding the number of its child nodes in the class hierarchy. Since C_i is a binary variable, it is enough to compute either $P(C_i = c_i|\mathbf{x})$ or $P(C_i = \tilde{c_i}|\mathbf{x})$. Next we show how to compute $P(C_i = \tilde{c_i}|\mathbf{x})$ since it allows for more simplifications in equation (1).

Theorem 1. *Estimating $P(\mathbf{x})P(C_i = \tilde{c_i}|\mathbf{x})$ requires constant time (that is, the estimation time complexity is independent of the number of neighbours of C_i in the class hierarchy).*

First, lets us consider non-leaf nodes, we may decompose eq. (1) *as follows:*

$$P(C_i = \tilde{c_i}|\mathbf{x}) = \frac{1}{P(\mathbf{x})} \times \left[\sum_{\mathbf{y_i} \in S_1} \left(\frac{P(\mathbf{x}|C_i = \tilde{c_i}, \mathbf{y_i}) P(C_i = \tilde{c_i}|\mathbf{y_i}) P(\mathbf{x}|\mathbf{y_i}) P(\mathbf{y_i})}{\sum_{c' \in \{c_i, \tilde{c_i}\}} P(\mathbf{x}|C_i = c', \mathbf{y_i}) P(C_i = c'|\mathbf{y_i})} \right) + \quad (2)$$

$$\sum_{\mathbf{y_i} \in S_2} \left(\frac{P(\mathbf{x}|C_i = \tilde{c}_i, \mathbf{y_i}) P(C_i = \tilde{c}_i|\mathbf{y_i}) P(\mathbf{x}|\mathbf{y_i}) P(\mathbf{y_i})}{\sum_{C_i \in \{c_i, \tilde{c}_i\}} P(\mathbf{x}|C_i = c', \mathbf{y_i}) P(C_i = c'|\mathbf{y_i})} \right) + \tag{3}$$

$$\sum_{\mathbf{y_i} \in S_3} \left(\frac{P(\mathbf{x}|C_i = \tilde{c}_i, \mathbf{y_i}) P(C_i = \tilde{c}_i|\mathbf{y_i}) P(\mathbf{x}|\mathbf{y_i}) P(\mathbf{y_i})}{\sum_{C_i \in \{c_i, \tilde{c}_i\}} P(\mathbf{x}|C_i = c', \mathbf{y_i}) P(C_i = c'|\mathbf{y_i})} \right) \Bigg]. \tag{4}$$

Class sets S_1, S_2 and S_3 are the only three possible types of label configuration for the neighbourhood of the i-th class node: 1) one parent and one child node, 2) one parent and no children and, 3) no parents and no children. Notice that these are the only possibilities because we are dealing with single-path predictions in a tree setting. Figure 1 displays a graphical representation of the three possibilities.

We may eliminate expression (2) from the summation since, when $\mathbf{y_i} \in S_1$, $P(C_i = \tilde{c}_i|\mathbf{y_i})$ equals to 0 because there is no case where $\tilde{c}_i$ happens and both its parent and some child node are active (have a positive class label). This is because, due to the "Is-a" hierarchy, if a child of C_i is active, C_i must be active too. Similarly we may simplify expressions (3) and (4) considering that $P(C_i = c_i|\mathbf{y_i})$ is equal to 0 when $\mathbf{y_i} \in S_2$ or $\mathbf{y_i} \in S_3$. Equation (7) *presents the simplifications.*

$$P(C_i = \tilde{c}_i|\mathbf{x}) = \frac{1}{P(\mathbf{x})} \times \tag{5}$$

$$\Bigg[\sum_{\mathbf{y_i} \in S_2} \left(\frac{\cancel{P(\mathbf{x}|C_i = \tilde{c}_i, \mathbf{y_i}) P(C_i = \tilde{c}_i|\mathbf{y_i})} P(\mathbf{x}|\mathbf{y_i}) P(\mathbf{y_i})}{\cancel{P(\mathbf{x}|C_i = \tilde{c}_i, \mathbf{y_i}) P(C_i = \tilde{c}_i|\mathbf{y_i})}} \right) + \sum_{\mathbf{y_i} \in S_3} \left(\frac{\cancel{P(\mathbf{x}|C_i = \tilde{c}_i, \mathbf{y_i}) P(C_i = \tilde{c}_i|\mathbf{y_i})} P(\mathbf{x}|\mathbf{y_i}) P(\mathbf{y_i})}{\cancel{P(\mathbf{x}|C_i = \tilde{c}_i, \mathbf{y_i}) P(C_i = \tilde{c}_i|\mathbf{y_i})}} \right) \Bigg] \tag{6}$$

$$P(\mathbf{x}) P(C_i = \tilde{c}_i|\mathbf{x}) = \sum_{\mathbf{y_i} \in S_2} P(\mathbf{x}|\mathbf{y_i}) P(\mathbf{y_i}) + \sum_{\mathbf{y_i} \in S_3} P(\mathbf{x}|\mathbf{y_i}) P(\mathbf{y_i}). \tag{7}$$

Therefore, because sets S_2 and S_3 are of unitary cardinality, containing only one possible configuration of labels for the neighbour (parent or child) classes of C_i, calculating $P(\boldsymbol{x}) P(C_i = \tilde{c}_i|\boldsymbol{x})$ requires constant time complexity with respect to the number of neighbours of C_i in the class hierarchy.

To calculate the probability $P(\boldsymbol{x}) P(C_i = \tilde{c}_i|\boldsymbol{x})$ for leaf nodes we may follow the same steps considering that the set S_1 is empty (the leaf nodes have no children), the set S_2 contains only one element (the parent node of the leaf) and the set S_3 is empty, meaning that the time complexity for the probability estimation for leaf nodes is also independent of the number of neighbors. ■

To estimate the normalizing constant $P(\mathbf{x})$ we would need the value of $P(C_i = c_i|\mathbf{x})$, which is the original probability estimated by [9], the very value that we would like to avoid calculating because of the time complexity dependency on the number of neighbours of the class node.

Thus, we use a heuristic to get rid of the normalizing constant $P(\mathbf{x})$: we calculate all non-normalized probabilities and assume that the largest one, $Z_{pseudo} = \max_{C_i}(P(\mathbf{x}) P(C_i = \tilde{c}_i|\mathbf{x}))$, is the pseudo-normalization constant. Additionally, we do not use the geometric average approach proposed by [9] to find the most probable path in the class hierarchy, instead, we use a top-down strategy: first find the most probable class that is a child of the root node, then, recursively find the most probable child of this node, and so on, until a leaf node is reached. This approach had better results than the original strategy of using the geometric average in our experiments.

The overall time complexity of the training phase of the modified algorithm, considering the number of probability distributions that need to be estimated, is $O(N)$, where N is the number of class nodes in the hierarchy. The time complexity of the prediction phase, considering the number of evaluations of probability functions is also $O(N)$.

Both time complexities are independent of the number of neighbours of the class nodes and are strictly smaller than $O(S_{mean} \times N)$, where S_{mean} is the mean number of neighbors of all class nodes. S_{mean} must be larger than one, since the size of the neighborhood set S is at least 1 for all nodes if the graph is connected.

IV. Experiments

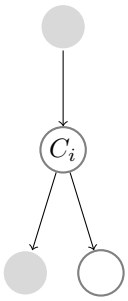
In this section we present the effects of the modification made in the original ELHNB algorithm with respect to predictive performance, training time, and prediction time. To test the algorithm, we use 18 bioinformatics datasets, eight encoding protein functions and ten encoding gene functions.

A. Datasets

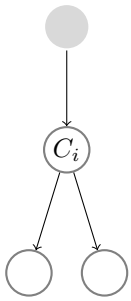
We conducted our experiments in the same 18 datasets made available by [9] [1]. These datasets are commonly used in works about hierarchical classification. For the sake of organization, these datasets are divided in two groups: Group A contains eight protein function datasets, four related to G Protein-Coupled Receptors (GPCRs) and four related to enzymes. GPCRs are transmembrane proteins that are common targets of many medical drugs. Enzymes are large molecules that speed up certain biochemical reactions. The names of the datasets related to enzymes start with EC (Enzyme Commission) and the names of the datasets related to GPCR proteins start with GPCR. The predictor attributes of datasets in Group A are the binary values that represent the existence (or not) of a particular protein signature (motif) in a protein amino acid sequence, the amino acid sequence length, and the molecular weight. The second part of the dataset name represents the type of motif that was used to create the dataset (Interpro , FigerPrints, Prosite and Pfam). The creation of these datasets is described in [4].

Group B contains different types of datasets related to the Yeast genome, namely: secondary structure, phenotype, homology, sequence statistics, and expression. The classes to be predicted were extracted from FunCat. The creation of these datasets is described in [2]. Table I presents the main characteristics of the datasets used in this work, where "#" means "number of".

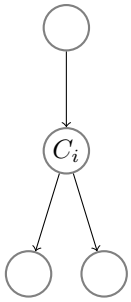
[1] The datasets and algorithms used in this work are available at http://www.cs.kent.ac.uk/people/rpg/ff79/biokdd14 after the publication of this paper.



(a) Case 1 - C_i's parent is active and exactly one child node of C_i is active. In this case, the number of possible configurations of labels for the set y_i of neighbours of C_i is equal to the number of children of C_i.



(b) Case 2 - C_i's parent is active, C_i's children are not active. This case contains only one possible configuration of labels for the set y_i of neighbours of C_i.



(c) Case 3 - None of the neighbours of C_i is active. This case contains only one possible configuration of labels for the set y_i of neighbours of C_i.

Fig. 1: Simplified examples of the three possible cases, shaded nodes represent the active neighbourhood of the node C_i, i.e., neighbours whose class label is present in an instance.

TABLE I: Main characteristics of the datasets

Group	Datasets	# Attributes	# Instances	# Classes per level Level
A	GPCR-Interpro	450	6935	12/54/82/50
	GPCR-Pfam	75	6524	12/52/79/49
	GPCR-Prints	283	4880	8/46/76/49
	GPCR-Prosite	129	5728	9/50/79/49
	EC-Interpro	1216	11101	6/41/96/187
	EC-Pfam	708	11057	6/41/96/190
	EC-Prosite	585	11328	6/42/89/187
	EC-Prints	382	11048	6/45/92/208
B	CellCycle	77	2486	16/47/69/32/8
	Church	27	2499	16/49/67/34/6
	Derisi	63	2497	16/48/70/31/7
	Eisen	79	1641	16/43/55/23/2
	Expr	551	2554	16/49/68/28/5
	Gasch1	173	2595	16/48/71/32/7
	Gasch2	52	2631	17/49/68/34/6
	Phenotype	69	1023	15/43/40/15/1
	Sequence	478	2689	17/48/65/29/5
	SPO	80	2463	16/48/68/31/8

B. Predictive Performance

In this section we measure the impact of the proposed modification in regards to predictive performance. We use the well-known hierarchical F-Measure to measure predictive performance. The hierarchical F-Measure (hF) is defined as $hF \equiv \frac{2*hP*hR}{hP+hR}$, [6] where hP is the hierarchical precision and hR is the hierarchical recall, defined as $hP \equiv \frac{\sum_j |P_j \cap T_j|}{\sum_j |P_j|}$ and $hR \equiv \frac{\sum_j |P_j \cap T_j|}{\sum_j |T_j|}$, where P_j is the set of predicted classes of the j-th instance and T_j is the set of true classes of the j-th instance.

Table II presents the hierarchical F-Measure of the modified and original algorithm in 18 datasets. To test whether the hF value of the modified algorithm (M-ELHNB) is statistically equivalent to the hF value of the standard ELHNB, considering the combined results of all datasets, we used the two-sided Wilcoxon Signed Rank test [3]. According to the statistical test, it is not possible to reject the null hypothesis that the algorithms are equivalent (p-value of 0.673 for $\alpha = 0.05$).

TABLE II: Comparison between the Standard (ELHNB) and the Modified Algorithm (M-ELHNB) in regards to the hierarchical F-Measure using 10-fold cross validation. Numbers in brackets are the standard errors. Numbers in bold face indicate the best performing algorithm.

Group	Dataset	ELHNB hF	M-ELHNB hF
A	GPCR-Pfam	**0.6087** (0.0042)	0.5926 (0.0037)
	GPCR-Prosite	**0.5893** (0.0068)	0.5597 (0.0070)
	GPCR-Prints	**0.7689** (0.0048)	0.7567 (0.0054)
	GPCR-Interpro	**0.7693** (0.0021)	0.7573 (0.0042)
	EC-Prints	0.9360 (0.0022)	**0.9451** (0.0016)
	EC-Prosite	0.9490 (0.0022)	**0.9686** (0.0010)
	EC-Pfam	0.9604 (0.0016)	**0.9748** (0.0012)
	EC-Interpro	0.9605 (0.0021)	**0.9772** (0.0013)
B	CellCyle	0.0898 (0.0084)	**0.1382** (0.0064)
	Church	0.0881 (0.0049)	**0.0961** (0.0045)
	Derisi	0.0751 (0.0045)	**0.0825** (0.0033)
	Eisen	0.0489 (0.0023)	**0.1482** (0.0038)
	Expr	**0.0536** (0.0053)	0.0347 (0.0025)
	Gasch1	**0.0674** (0.0059)	0.0178 (0.0027)
	Gash2	0.1153 (0.0061)	**0.1307** (0.0056)
	Phenotype	0.0761 (0.0062)	**0.0809** (0.0072)
	Sequence	**0.0434** (0.0025)	0.0274 (0.0021)
	SPO	0.0741 (0.0043)	**0.1091** (0.0060)

C. Running time

To test the running time of the two algorithms we measured the total running times of the training and prediction phases of all 10 steps of the 10-fold cross-validation. All algorithms were executed in a cluster computer with 24 Xeon E5520 processors with 12 GB of RAM memory, running Ubuntu 12.04. We used the Oracle Grid Engine to distribute the jobs. Both algorithms were implemented in the Python programming language.

Figures 2 to 4 present the running times of the algorithms in the selected datasets. We divided the datasets in three groups, according to their overall running time. Group I contains the smallest running times and Group III the largest. It is clear from the figures that the modified algorithm has a better running time in both the training and in the prediction phases, as expected. The difference in running times is specially significant in Group III: e.g., in the dataset "EC-Pfam" the

ELHNB algorithm took 88.2 hours to run its training and prediction phases, while the modified algorithm took only 37.2 hours. A difference of more than 2 days, and a reduction of about 58% in the time taken by ELHNB.

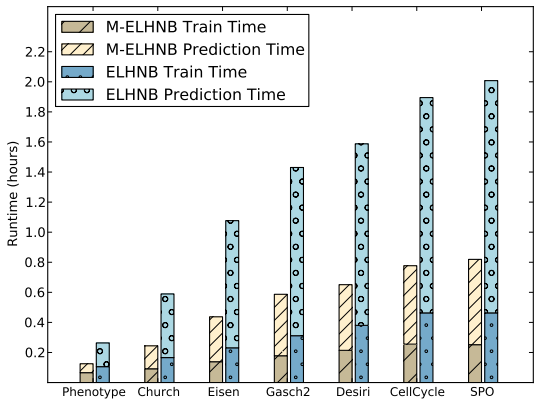


Fig. 2: Running time of the classifications algorithms, Group I

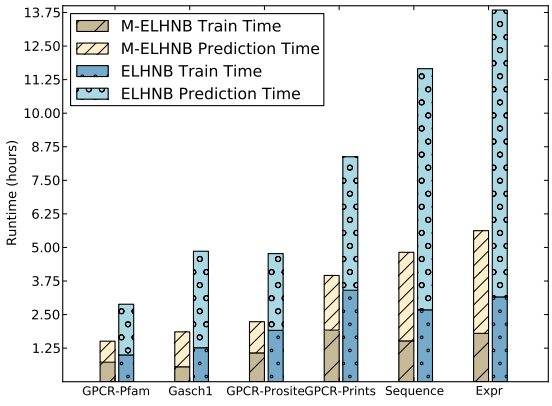


Fig. 3: Running time of the classifications algorithms, Group II

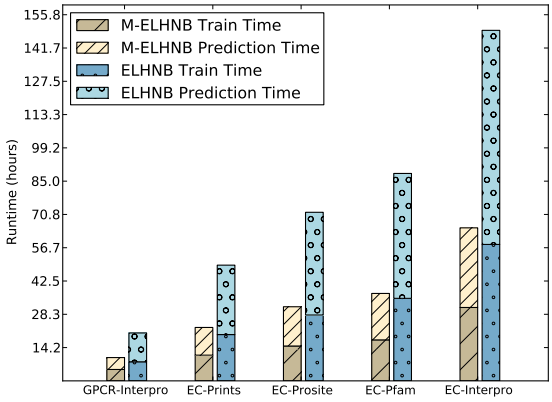


Fig. 4: Running time of the classifications algorithms, Group III

V. Conclusion and Future Work

In this paper we presented a modified version of the ELHNB algorithm specialized for hierarchical classification problems where each instance is assigned a single path from the root node to a leaf node in the class hierarchy - called single path, mandatory leaf class prediction problems. We showed that our algorithm is statistically equivalent to the standard one in terms of predictive performance, but significantly faster.

An extension of this work may be the application of the modified algorithm in other domains and the extension of the algorithm for other types of hierarchical structures such as DAGs, multiple paths, non-mandatory leaf class prediction classification problems.

Acknowledgment

The first author is financially supported by CAPES, a Brazilian research-support agency (process number 0653/13-6).

References

[1] H. Blockeel, L. Schietgat, J. Struyf, S. Džeroski, and A. Clare, "Decision Trees for Hierarchical Multilabel Classification: A Case Study in Functional Genomics," in *Knowledge Discovery in Databases: PKDD 2006*, ser. Lecture Notes in Computer Science, vol. 4213. Springer Berlin, 2006, pp. 18–29.

[2] A. Clare and R. D. King, "Predicting gene function in Saccharomyces cerevisiae," *Bioinformatics*, vol. 19, no. 2, pp. ii42–ii49, Oct. 2003.

[3] J. Demsar, "Statistical Comparisons of Classifiers over Multiple Data Sets," *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

[4] N. Holden and A. A. Freitas, "Improving the Performance of Hierarchical Classification with Swarm Intelligence," in *Proc. of the 6th European Conf. on Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics (EvoBIO)*. Springer, 2008, pp. 48–60.

[5] C. N. S. Jr and A. A. Freitas, "A survey of hierarchical classification across different application domains," *Data Mining and Knowledge Discovery*, vol. 44, no. 1-2, pp. 31–72, 2011.

[6] S. Kiritchenko, S. Matwin, and F. Famili, "Functional annotation of genes using hierarchical text categorization," in *BioLINK SIG: Linking Literature, Information and Knowledge for Biology*, 2005.

[7] The Gene Ontology Consortium, "The Gene Ontology (GO) database and informatics resource." *Nucleic acids research*, vol. 32, pp. D258–61, 2004.

[8] The Uniprot Consortium, "The Universal Protein Resource (UniProt) in 2010." *Nucleic acids research*, vol. 38, pp. D142–8, Jan. 2010.

[9] L. d. C. Merschmann and A. A. Freitas, "An Extended Local Hierarchical Classifier for Prediction of Protein and Gene Functions," ser. Lecture Notes in Computer Science, vol. 8057. Springer Berlin, 2013, pp. 159–171.

[10] L. D. Stein, "Integrating biological databases." *Nature Reviews Genetics*, vol. 4, no. 5, p. 337, 2003.