Computers and Mathematics with Applications 66 (2013) 1991–2002

Contents lists available at ScienceDirect

Computers and Mathematics with Applications

journal homepage: www.elsevier.com/locate/camwa

An evaluation of global-model hierarchical classification algorithms for hierarchical classification problems with single path of labels

Helyane Bronoski Borges [a,b,*], Carlos N. Silla Jr. [a], Júlio Cesar Nievola [b]

[a] UTFPR – Universidade Tecnológica Federal do Paraná, Brazil

[b] PPGIa - Pontifícia Universidade Católica do Paraná (PUCPR), Brazil

ARTICLE INFO

Keywords:
Hierarchical classification
Global approach

ABSTRACT

Several classification tasks in different application domains can be seen as hierarchical classification problems. In order to deal with hierarchical classification problems, the use of existing flat classification approaches is not appropriate. For these reason, there has been a growing number of studies focusing on the development of novel algorithms able to induce classification models for hierarchical classification problems. In this paper we study the performance of a novel algorithm called Hierarchical Classification using a Competitive Neural Network (HC-CNN) and compare its performance against the Global-Model Naïve Bayes (GMNB) on eight protein function prediction datasets. Interestingly enough, the comparison of two global-model hierarchical classification algorithms for single path of labels hierarchical classification problems has never been done before.

1. Introduction

In the hierarchical classification literature, there are several tasks that can benefit from the use of hierarchical classification algorithms, such as protein function prediction [1,2], text categorization [3,4], music genre classification [5], among others [6].

Although the hierarchical classification field is receiving increasing attention by the scientific community, there are still many open questions. One of them is: Which hierarchical classification algorithm is better in different types of hierarchical classification problems?

In the context of flat classification, there are several algorithms available within different software packages. Unfortunately this is not the case for the hierarchical classification research area, where only very few resources are available. As we shall see in Section 2, hierarchical classification problems and algorithms can be classified according to their core characteristics. By understanding the different core characteristics of the hierarchical classification problems and algorithms, it becomes clearer that there has been no comparison of different types of global model hierarchical classification algorithms for single path of labels problems in the literature, which is the main contribution of this paper.

This paper is an extended version of [7]. The main novelty of this paper is that we compare the HC-CNN algorithm (proposed in [7] and presented in Section 3) with two versions of the Global Model Naïve Bayes (proposed in [8] and presented in Section 4) on eight hierarchical protein function prediction datasets with single path of labels. The experimental settings, computational results and their analysis are presented in Section 5. Finally in Section 6, we present the conclusions and future research directions.

* Corresponding author at: UTFPR – Universidade Tecnológica Federal do Paraná, Brazil. Tel.: +55 4232204827.
E-mail addresses: helyane@utfpr.edu.br (H.B. Borges), carlosjunior@utfpr.edu.br (C.N. Silla Jr.), nievola@ppgia.pucpr.br (J.C. Nievola).

0898-1221/$ – see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.camwa.2013.06.027

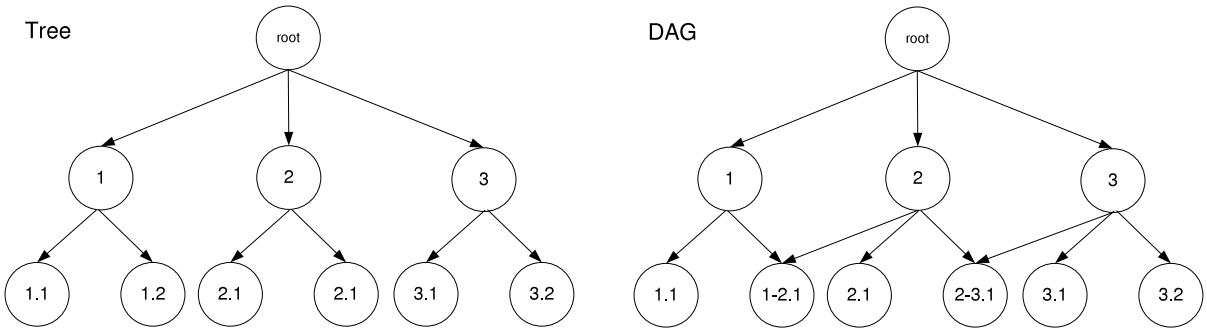

Fig. 1. Example of the different types of structure for hierarchical classification problems.

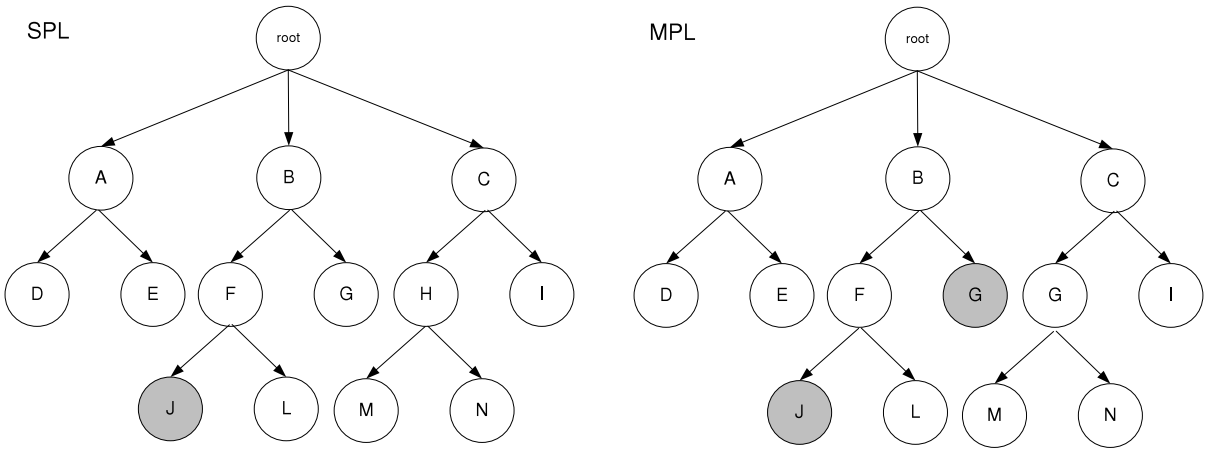

Fig. 2. Example of the difference between a hierarchical classification SPL and MPL problems.

2. Hierarchical classification

The task of automatic classification is a process that consists in associating a particular example to one or more classes, among a set of predefined classes according to the characteristics (attributes) of the example.

In general, the task of automatic classification can be divided into two types: flat (conventional) classification and hierarchical classification. Most of the problems cited in the literature involve the flat classification approach where each data instance belongs to one specific class out of a set of classes, and where each class is independent of the others.

It is important, therefore, to better understand the main characteristics of hierarchical classification problems and hierarchical classification approaches.

2.1. Hierarchical classification problems

The hierarchical classification task differs from the flat classification task because the classes are organized in a hierarchy structured as either a tree or a directed acyclic graph (DAG) where the nodes of this hierarchy represent the classes that are involved in the classification process.

The main difference between the tree structure and the DAG structure is that in the tree structure each node (each class), except the root node, has only one ancestor (parent), while in the DAG structure each node (class) may have one or more ancestor nodes.

However, as pointed out in [6] the hierarchical class structure is not the only core characteristic of hierarchical problems. In [6] the authors define the core characteristics a hierarchical classification problem by using a 3-tuple $\langle \boldsymbol{\Upsilon}; \boldsymbol{\Psi}; \boldsymbol{\Phi} \rangle$, where:

- $\boldsymbol{\Upsilon}$ specifies the problem structure representing the hierarchical classes (nodes in the graph) and their interrelationships (edges in the graph). This attribute can have as value either a tree (T) or a Directed Acyclic Graph (DAG). Fig. 1 shows an example to illustrate the difference between the two types of structure for hierarchical classification problems.
- $\boldsymbol{\Psi}$ indicates whether a data instance is allowed to have class labels associated with single or multiple paths in the class hierarchy; this attribute can have as value either SPL (Single Path of Labels) or MPL (Multiple Path of Labels). In the case of SPL the problem only has one path of labels in the hierarchy while in the case of MPL the problem has some examples (instances) that have more than one path of labels in the hierarchy. The MPL is also often referred in the literature as

a Multi-Label Hierarchical Classification problem. Fig. 2 shows an example to illustrate the difference between the two scenarios.

- Φ describes the label depth of the data instances. If all the examples (instances) of the dataset are labeled up to the leaf nodes, this attribute is said to have full depth (FD) labeling. Otherwise the dataset is said to have partial depth (PD) labeling. In the case of PD labeling, Silla and Freitas recommend that the percentage of partial depth labeled examples is informed by using $PD_{\%}$.

2.2. Hierarchical classification approaches

According to [6] the existing approaches to hierarchical classification can be classified as flat classification approaches, local-model hierarchical classification approaches and global-model hierarchical classification approaches. In this section we briefly review this approaches.

2.2.1. Flat classification approach

The flat classification approach has the same behavior of a conventional classification algorithm in the training and testing phases. This approach considers that a hierarchical classification problem can be transformed into a flat classification problem disregarding the concept of ancestors and descendants, i.e. it ignores the class hierarchy, predicting only the leaf nodes. This approach is similar to conventional flat classification and can be applied to tree and DAG structures. Fig. 3 illustrates this approach.

2.2.2. Local-model hierarchical classification approaches

As Silla and Freitas [6] explain, this approach can be divided into three types of local-model hierarchical classification approaches, namely: Local Classifier per Node Approach, Local Classifier per Parent Node Approach and Local Classifier per Level Approach.

2.2.2.1. Local classifier per node approach. This approach is the most used in the literature [4,9,11,10]. It trains a binary classifier for each node of the class hierarchy, i.e. it uses M local independent classifiers, one for each class (M is the total number of nodes in the class hierarchy). Consequently, the number of classifiers to be trained could be huge in situations where there are many classes. Moreover, a crucial problem that can arise is that the results can be inconsistent, because there is no guarantee that the class hierarchy is respected. Fig. 4 illustrates this approach.

2.2.2.2. Local classifier per parent node approach. This approach consists in training each parent node of the class hierarchy as a multi-class classifier [6]. This approach can't be used with hierarchical classification problems that have DAG structure without a strategy for creating classifiers. Fig. 5 illustrates this approach.

2.2.2.3. Local classification per level approach. This approach consists in creating a multi-class classifier for each level of the hierarchy [6]. This approach could be used in tree and DAG structures. In DAG structures, the application of this technique is more complex, since there may be more than one path in a DAG structure. Thus, a class can belong to more than one level in the hierarchy, which can bring redundancy among the classifiers. The same problem can occur if a node class has been wrongly propagated to the following levels of hierarchy. Fig. 6 illustrates this approach.

The standard testing approach for local-model hierarchical classification is to use the Top-Down approach. The top-down approach starts by predicting one of the classes among the classes at first level of the hierarchy, then it proceeds by predicting one of the children classes of the predicted class. This process is repeated until a leaf node is reached [1].

2.3. Global or big-bang classifier

The global or big-bang hierarchical classification approach builds a single classification model considering the class hierarchy, based on the training set, as the work of [12,22,23]. In this approach, the prediction can occur at any level of hierarchy. Thus, none of the approaches used for flat classification can be used, without changing the classifier.

The main advantage of this approach is that there is no need to train a large number of classifiers and to deal with the inconsistency in the prediction of classes. Its main disadvantage is the increased complexity on developing the global classifier. Fig. 7 illustrates this approach.

2.4. Formalization of the hierarchical classification approaches

The classification of the hierarchical classification approaches presented so far identifies the main core characteristic of the existing hierarchical classification approaches. However, as pointed out in [6], the hierarchical classification approaches should also be formally described using a 4-tuple $\langle \Delta; \Xi; \Omega; \Theta \rangle$, where:

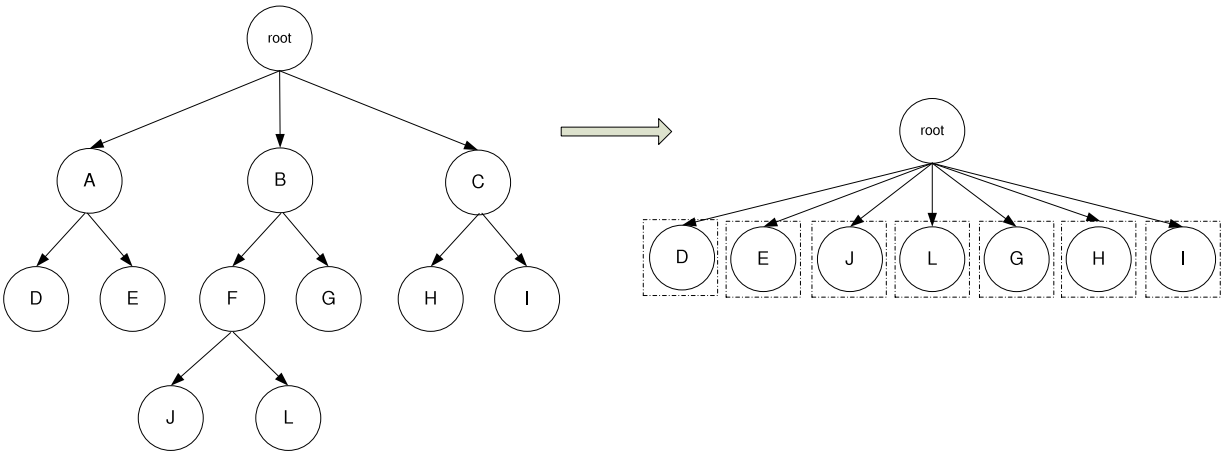

Fig. 3. Example of the flat classification approach for a hierarchical classification problem.

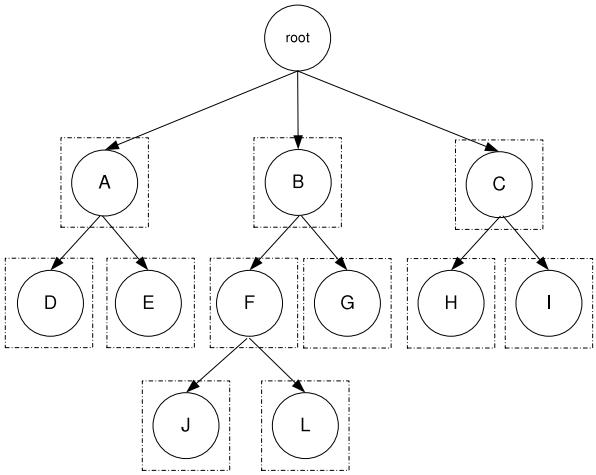

Fig. 4. Example of the Local Classification per Node (LCN) approach.

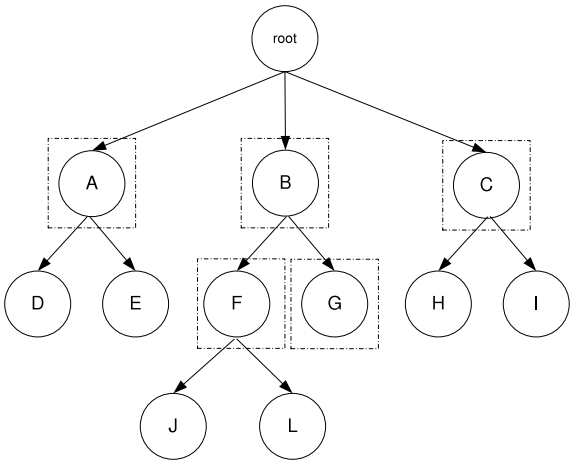

Fig. 5. Example of the Local Classification per Parent Node (LCPN) approach.

- Δ specifies if the algorithm can predict labels in just one or multiple different paths in the hierarchy. The possible values for this attribute are *SPP* (Single Path Prediction), if the algorithm can only deal with problems which have Ψ = single

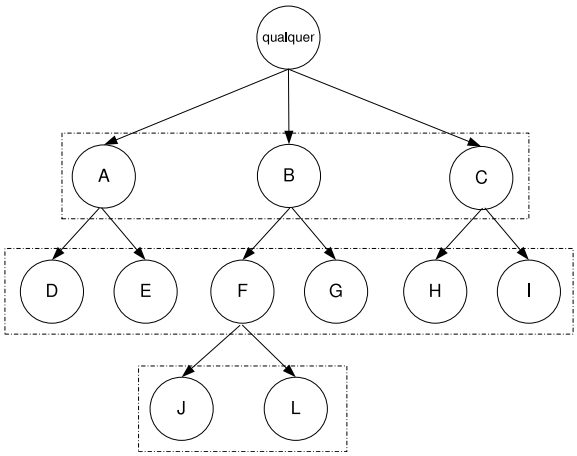

Fig. 6. Example of the Local Classification per Level (LCL).

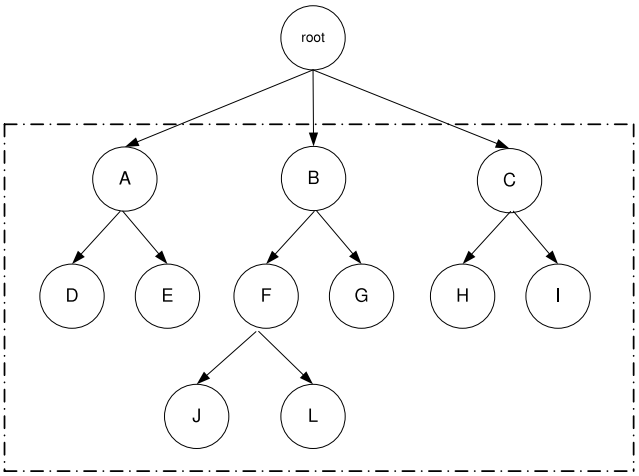

Fig. 7. Example of the global hierarchical classification (GC) approach.

path of labels; or *MPP* (Multiple Path Prediction) if the algorithm can deal with problems which have Ψ = multiple path of labels.

- Ξ specifies the prediction depth of the algorithm. It can have two values: *MLNP* (Mandatory Leaf-Node Prediction), which means the algorithm assigns leaf class(es) to every data instance; *NMLNP* (Non-Mandatory Leaf-Node Prediction), which means the algorithm does not need to assign a leaf class to every instance (i.e., for some instances the most specific predicted class can be a non-leaf class).
- Ω specifies the type of hierarchical classification structure that the algorithm can handle. It has two values: *T* (tree), *D* (DAG), indicating that the algorithm can predict classes arranged into a tree or DAG structure, respectively.
- Θ is the categorization of the algorithm under the proposed taxonomy proposed in [6]. It can have the following values: LCN (Local Classifier per Node); LCPN (Local Classifier per Parent Node); LCL (Local Classifier per Level); GC (Global Classifier).

The main advantage of identifying the existing and proposed algorithms using this formalism is to better identify if a given problem is being handled by an adequate algorithm, i.e. if the problem has partial depth labeling, it would make sense to use a hierarchical classification approach which is capable of non-mandatory leaf node prediction. Furthermore, by comparing the existing studies in hierarchical classification [6], it seems (to the best of the authors knowledge) that no comparison between $\langle SPP, NMLP, T, GC\rangle$ approaches have been made. Most of the existing papers in the literature compare their $\langle MPP, NMLP, D, GC\rangle$ approaches against the ClusHMC (Hierarchical multi-label classification) algorithm [12], such as the hm-Ant-Miner [13]. Unfortunately both the ClusHMC and the hm-Ant-Miner algorithms work by not giving one answer to the user, but rather a list with all the classes and their respective probabilities. The user must then select a threshold in order to have an answer. Because of this fact, in this work we employed two $\langle SPP, NMLP, T, GC\rangle$ algorithms that are able to give the user a set of classes as output.

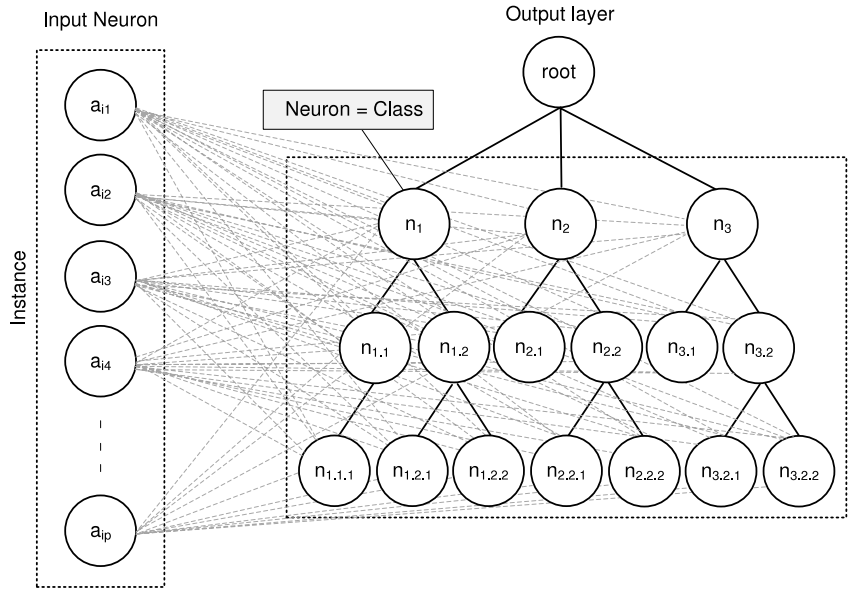

Fig. 8. Example of the HC-CNN.

3. Hierarchical classification using a competitive neural network

One of the characteristics of a competitive network is its ability to realize mappings that preserve the topology of the input and output spaces. The learning process proposed here is based on competitive learning [14], in which neurons of the output layer compete to be activated so that only one output neuron will be the "winner" of the competition process. The synaptic weight adjustments are made by the neuron that was activated and its neighbors.

The HC-CNN algorithm proposed in this paper is based on a Competitive Artificial Neural Network [14,15]. Fig. 8 shows a neural network model. This network consists of two layers of neurons. The input layer is connected to an input vector dataset. The term "Input neurons" defined in this figure represents all instances of entry, according to the interpretation of the dataset. The processing layer or output layer, which is a competitive network is the output mapping, represents the hierarchy of classes, where each neuron is connected to its ancestors and possibly descendants.

In a traditional competitive network, for example, the Kohonen network, the neurons of the output layer are arranged in a grid network [15], which can be rectangular, hexagonal, among others, and they represent the network topology. In HC-CNN algorithm the topology is a tree, where each neuron is connected with its ancestor (parent) and descendant (children) neurons. These neurons (output layer) are created according to the number of classes in the hierarchy, and each neuron in the output layer is connected to all neurons of the input layer.

Neurons are stimulated by the input examples during the competitive process. In this way, it will be considered the "winner" the neuron who is more similar to the input instance selected. The comparison is made through the use of a distance measure.

Prior to the training, some parameters should be defined, for instance, the amount of epochs to train the neural network and its learning rate (initial and final) which will decrease exponentially during the training, and the synaptic weights are randomly initialized. The training process of the network is divided into three phases, as in a traditional competitive network: Competition, Cooperation and Adaptation.

3.1. Competition

An instance e_i of input dataset $DB_{Train} = [e_1 e_2 e_3 \dots e_q]$ of dimension q is selected. Each element of DB_{Train} consists of attributes $e_i = [a_{1i} a_{2i} a_{3i} \dots a_{li}]$, where a_{jk} is the j-th attribute, $1 \leq j \leq l, l-1$ is the number of input attributes, and a_{li}, $i \in N : 1 \leq i \leq q$, represents the class attribute.

The neural network is created according to information obtained in the input data: the number of input neurons is equal to the number of input attributes, the output layer neurons are represented by $NN = [n_1 n_2 n_3 \dots n_b]$ where b is the number of the class that exists in the class hierarchy. Each neuron of the output layer consists of synaptic weights $n_i = [p_{1i} p_{2i} p_{3i} \dots p_{(l-1)i}]$ such that $i \in N : 1 \leq i \leq (l-1)$.

To find the input vector e_i that is closest to the synaptic weights vector, a distance measure is used. The measure chosen was the Euclidean distance (Eq. (1)).

$$d_{ik} = \sum_{i=1}^{l-1} \sqrt{(e_{ij} - n_{ijk})^2} \tag{1}$$

where e_i is the input instance, n_{ijk} is the k-th neuron of the output layer, l is the number of attributes and k is the amount of output layer neurons.

The next step is to identify the neuron that shows the lowest distance (Eq. (2)). Thus, the winner neuron of the competition process of the cycle is obtained:

$$nv = \arg\min(d_i). \tag{2}$$

Thus, the neuron of network NN that is closest to the selected input instance will be considered the winner and will stimulate their neighborhood according to the network topology.

3.2. Cooperation

The winner tends to excite the ancestor neurons that are closest. Thus, this phase is located in the region of the topological neighborhood that will update the weights of neurons.

In the HC-CNN algorithm, the neighborhood criterion is determined from the relationship between the neuron and its ancestors (parents of the winner neuron) and descendants (children of the winner neuron). This information is obtained through the existing relationship in the class hierarchy.

3.3. Adaptation

The third phase is the adaptation process of synaptic weights. The objective of training is to approximate the weight vectors to the input instances, because the weights of the neurons need to be adjusted so that there is a better classification of the instance e_i. If the class labels of the input instance e_i are equal to the winner neuron, then the neuron weights are adjusted so that they are closer to the instance. This is the case of a correct prediction. If the class labels of the input instance e_i are different from the labels of the predicted ones, weights will be updated to be more distant from this instance, because the class is predicted incorrectly. That is, during training, the algorithm adjusts the weights of the neuron and its ancestors, making the comparison by the identifier of the class of each input instance with the desired output.

Eq. (3) shows the way the neuron weights are updated.

$$n_{ijk}(t+1) = \begin{cases} (n_{ijk} + (e_i(t) - n_{ij}(t)) * Ap(t) * Dist(t)), & \text{if the correct class} \\ (n_{ijk} - (e_i(t) - n_{ij}(t)) * Ap(t) * Dist(t)), & \text{if the incorrect class} \end{cases} \tag{3}$$

where n_{ijk} is the weight of the neuron attribute of the output layer at iteration t between the input neuron (instance) e_i and the neuron k.

Ap is the learning rate for the instant of time $(t+1)$ which is obtained according to Eq. (4).

$$Ap(t) = (\mu_i - \mu_f) * e^{-\frac{t_{current}}{C}} \tag{4}$$

where μ_i is the initial learning rate, μ_f the end learning rate, $t_{current}$ is the current iteration and C is a constant defined for the exponential function decreases slowly.

$Dist(t)$ is the distance of winner neuron and their ancestors neurons that will have its weights adjusted as shown in Eq. (5).

$$Dist(t) = 1/(k+1) \tag{5}$$

where k is the distance between nodes of the winning neuron and the ancestral neuron where the weights will be adjusted.

The updating of the weights of ancestor neurons follows the same method shown in the above equations (Eqs. (3)–(5)).

After the weights updating, a new instance is selected and all the procedure is repeated until all instances are selected. At the end, the first epoch of the training has been completed. The procedure is repeated again until the execution of all epochs.

In the last iteration of this phase the synaptic weights set of neurons is obtained and will be used in the test phase of the algorithm. Table 1 presents a succinct description of the HC-CNN algorithm.

The test of the algorithm is done in a way similar to training. Just as the training database, the testing database consists of instances $DB_{Test} = [x_1 x_2 x_3 \ldots x_g]$ where g the amount of test instances. Each instance x_i is composed of attributes $x_i = [a_{1i} a_{2i} a_{3i} \ldots a_{li}]$, $i \in N : 1 \leq i \leq g$, a_{li} represents the class attribute and l is amount attribute.

Table 2 shows the procedure for testing the HC-CNN. One can observe that it is similar to the training procedure. The main difference is that at this stage the weights are fixed from the last cycle of the training phase.

Table 1
Training algorithm of the HC-CNN.

INPUT:
– Training dataset $DB_{Train} = [e_1 e_2 e_3 \ldots e_q]$ of dimension q.
STEP 1: **INITIALIZE** – The initial learning rate μ_i and the final learning rate μ_f. – Determine the number of cycles ep. – Initialize the synaptic weights of the network: $NN = [n_1 n_2 n_3 \ldots n_b]$ where b is the number of class that exists in the class hierarchy. – Calculate the learning rate $Ap(t) = (\mu_i - \mu_f) * e^{-t_{current}/C}$ that the $t_{current}$ is current iteration and C is a constant defined for the exponential function decreases slowly.
STEP 2: **STOPPING CRITERION** – Number of cycles.
STEP 3: **TRAINING** – Select an instance e_i of the input dataset $DB_{Train} = [e_1 e_2 e_3 \ldots e_q]$. **COMPETITION:** – Calculate the distance between the instance e_i with the neurons of the network $NN = [n_1 n_2 n_3 \ldots n_b]$. – Finding neuron n_j which had the shortest distance, that are considered the winner neurons. **COOPERATION:** – Find the ancestors of the neuron j. **ADAPTATION:** – Update the synaptic weights of neurons and their ancestors. – If the class true equals the predicted class $Dist(t) = 1$.
STEP 4: **UPDATE** – Update the learning rate **Ap(t).** – Start the STEP 2.
OUTPUT ALGORITHM – Adequate weights set.

Table 2
Test algorithm of the HC-CNN.

INPUT
– Test dataset $DB_{Test} = [x_1 x_2 x_3 \ldots x_g]$ of g is amount of test instance.
STEP 1: **STOPPING CRITERION** – Until all instances have been selected and tested.
STEP 2: **TEST** – Select an instance x_i of the input dataset $DB_{Test} = [x_1 x_2 x_3 \ldots x_g]$. – Calculate the distance between the instance x_i with the neurons of the network $NN = [n_1 n_2 n_3 \ldots n_b]$. – Finding neuron n_j which had the shortest distance, that are considered the winning neurons. – Find the ancestors of the neuron j. – Evaluation prediction. – Assign result to the confusion matrix.
OUTPUT ALGORITHM – Accuracy rate obtained by the algorithm.

4. Global-Model Hierarchical Classification Naïve Bayes (GMNB)

The flat Naïve Bayes algorithm is well known in the field of data mining for being a simple approach, with clear semantics, to representing, using and learning probabilistic knowledge [16]. For this reasons in [8] the authors extended the flat classification Naïve Bayes to deal with hierarchical classification problems in a global-model way, creating the Global-Model Hierarchical Classification Naïve Bayes (GMNB).

The training phase of the GMB consists of computing the probabilities by taking into account the available hierarchical information. Therefore, both the prior probability and the likelihoods computations have been modified to deal with the hierarchical classification scenario.

In order to modify the prior probability calculations to take the hierarchy into account, during the training phase, we will assume that any example which belongs to class C_k will also belong to all its ancestor classes. For example, if a training example belongs to a certain class (say class 2.1), this means that the prior probabilities of both that class and its ancestor classes (i.e. classes 2 and 2.1 in this case) are going to be updated. (This is because we are dealing with a "IS-A" class hierarchy, as usual.)

Hence, for the Global-Model Naïve Bayes (GMNB), the prior computation ($P(C_k)$) for each of the classes C_k; $k = 1, \ldots, nC$, is estimated by using the number of examples in the training set with class C_k (as in flat Naïve Bayes) plus the number of examples in the training set which are descendants of C_k divided by the total number of examples in the training set.

In the likelihood computation, using Fig. 4 as an example, the flat Naïve Bayes algorithm would compute the likelihood for only the leaf classes, i.e.: $P(a_{ij}|D)$, $P(a_{ij}|E)$, $P(a_{ij}|J)$, $P(a_{ij}|L)$, $P(a_{ij}|G)$, $P(a_{ij}|H)$, $P(a_{ij}|I)$. This is computed for each attribute A_i, with each value a_{ij} belonging to the domain of A_i, $i = 1, \ldots, n_a$, $j = 1, \ldots, n_{vi}$, where n_a is the number of attributes and n_{vi} is the number of values in the domain of the A_i-th attribute. The likelihood computations need to be modified to compute the likelihood for all classes in the hierarchy, i.e.: $P(a_{ij}|A)$, $P(a_{ij}|B)$, $P(a_{ij}|C)$, $P(a_{ij}|D)$, $P(a_{ij}|E)$, $P(a_{ij}|F)$, $P(a_{ij}|G)$, $P(a_{ij}|H)$, $P(a_{ij}|I)$, $P(a_{ij}|J)$, $P(a_{ij}|L)$.

As with the prior calculation modifications, we will assume that any example which belongs to class C_k will also belong to all its ancestor classes. This way when a training example is processed, its attribute–value pair counts are added to the counts of the given class and its ancestor classes. As in the previous example, if the training example belongs to class 2.1, the attribute–value counts are added to the counts of both classes 2 and 2.1.

These modifications during the training phase, will allow the algorithm to predict classes at any level of the hierarchy during the testing phase. In order to classify a new example, the Global-Model Naïve Bayes simply assigns the class with maximum value of the posterior probability, and outputs as the class label the class with the maximum posterior probability and its ancestor classes.

According to the taxonomy proposed by [6] the GMNB is a $\langle SPP, NMLP, D, GC\rangle$.

4.1. Global-model hierarchical-classification with usefulness (GMBwU)

The GMNB uses the MAP (Maximum a posteriori) rule to predict the final class label. Although the predictions of deeper classes are often less accurate (since deeper classes have fewer examples to support the training of the classifier than shallower classes), deeper class predictions tend to be more useful to the user, since they provide more specific information than shallower class predictions. If we only consider the posterior class probability (the product of likelihood $\times$ prior class probability) we would not take into account the usefulness to the user. It is interesting therefore to select a class label which has a high posterior probability and is also useful to the user.

For this reasons the authors of (GMNB) created a second version of the GMNB which uses a usefulness criterion known as GMNBwU. The GMNBwU has an additional step to predict the class with maximum value of the product of posterior probability $\times$ usefulness. The usefulness of each class is given by Eq. (6).

$$usefulness(C_k) = 1 - \left(\frac{a(C_k)\log_2 treesize(C_k)}{\max}\right) \tag{6}$$

where:

- $treesize(C_k) = 1 +$ number of descendant classes of C_k (1 is added to represent C_k itself).
- $a(C_k) = 0$, if $p(C_k) = 0$; $a(C_k) =$ a user defined constant (default $= 1$) otherwise.
- max is the highest value obtained by computing $a(C_k)\log_2 treesize(C_k)$ for all classes C_k; $k = 1, \ldots, n_c$ (where n_c is the number of classes) and it is used to normalize all the other values into the range [0, 1].

To make the final class prediction, the Global-Model Naïve Bayes with Usefulness (GMNBwU) assigns, to the current test example, the class label which maximizes the product of the posterior probability and usefulness as shown in Eq. (7).

$$classify(X) = \arg\max_{C_k} \prod_{i=1}^{n_a} (P(A_{i=a_{ij}}|C_k) * P(C_k)) * Usefulness(C_k). \tag{7}$$

5. Experiments

5.1. Datasets

The experiments to evaluate the classifiers predictive performance were performed on eight databases, four of them formed by protein G-Protein-Coupled Receptor (GPCR) and the other four formed by Enzyme Commission Codes (EC). These sets were available from the authors of the work [2]. Table 3 shows some characteristics of these databases.

For the all experiments 2/3 of the examples were used for training and 1/3 for testing (hold-out procedure). In addition, all sets were normalized using the Min–Max approach. An observation to be made is that some nodes of the hierarchy have only a few children, which causes a great unbalance in the tree.

Table 3
Characteristics of the datasets.

Datasets	Number of samples	Number of attributes	Number of class	Number of class per level
ECinterproFinal	14 036	1216	331	6/41/96/188
ECpfamFinal	13 995	708	333	6/41/96/191
ECprintsFinal	14 038	382	352	6/45/92/209
ECprositeFinal	14 048	585	324	6/42/89/191
GPCRinterproFinal	7 461	450	198	12/54/82/50
GPCRpfamFinal	7 077	75	192	12/52/79/49
GPCRprintsFinal	5 422	282	179	7/46/76/49
GPCRprositeFinal	6 261	128	187	9/50/79/49

5.2. Evaluation measures

The evaluation measures used to report the predictive performance of the hierarchical algorithms in this work are the ones developed by Kiritchenko et al. [17] and use concepts of ancestral and descendant classes.The authors proposed two evaluation measures: hierarchical precision and hierarchical recall which take into account the hierarchical relationships [17]. These measures are based on conventional measures of precision and recall.

These measures use the common ancestors of the true and predicted classes in the evaluation. To calculate the hierarchical recall (Eq. (8)), the number of common ancestors is divided by the number of ancestors of the true class

$$hR = \frac{\left|A(C_t) \cap A(C_p)\right|}{A(C_t)} \tag{8}$$

where $A(C_t)$ and $A(C_p)$ are classes that are true ancestors and predicted ancestors, respectively.

To calculate the hierarchical precision (Eq. (9)), the number of common ancestors is divided by the number of ancestors of the predicted class.

$$hP = \frac{\left|A(C_t) \cap A(C_p)\right|}{A(C_p)}. \tag{9}$$

These measures can be used to calculate an extension of the F-measure (harmonic mean), named the hierarchical F-measure (Eq. (10)).

$$hF - Measure = \frac{(\beta^2 + 1) * hR * hP}{\beta^2 * hR * hP} \tag{10}$$

where β is the importance given to precision and recall.

5.3. Experimental settings

5.3.1. HC-CNN

Initial learning rate and final learning rate used in the experiments with the HC-CNN were 0.1 and 0.01, respectively. The neural network synaptic weights were generated randomly, according to a uniform distribution. The evaluation of the classification was made taking into account all levels of the hierarchy.

5.3.2. GMNB and GMNBwU

All the datasets presented in Section 5.1 have two numerical attributes. It is a well-known fact that discretization is recommended over trying to estimate an unknown real-world underlying probability [18]. There are many different types of discretization algorithms, but they can be broadly classified (with regard to their class awareness) into supervised and unsupervised discretization methods [18]. An excellent review of discretization methods for Naïve Bayes is presented in [19]. In this work, due to the skewed nature of the data, we discretized all numerical attributes using an unsupervised discretization method known as Equal Frequency Discretization (EFD) [19] with 20 bins.

Another step that is needed to be taken into account when estimating $P(A_i|C_k)$ in the Bayesian classifiers is the case of when in the training data, a particular value of a_{ij} does not occur. In this case this value would be zero (0) and would drastically affect the performance of the classifier. In this work, when such a case happens, we estimate this value to be $1/n_e$ where n_e is the number of examples in the training set.

Table 4
Results for the different global-model hierarchical classifiers.

Datasets	HC-CNN			GMNB			GMNBwU		
	H-Prec.	H-Rec.	HF-Mea.	H-Prec.	H-Rec.	HF-Mea.	H-Prec.	H-Rec.	HF-Mea.
ECinterproFinal	82.5%	84.2%	83.4%	97.1%	90.6%	93.8%	96.6%	97.2%	96.9%
ECpfamFinal	83.2%	84.1%	83.6%	97.2%	89.1%	93.0%	95.7%	95.9%	95.8%
ECprintsFinal	87.4%	85.2%	86.3%	93.7%	88.8%	91.2%	92.7%	93.5%	93.1%
ECprositeFinal	92.3%	89.1%	90.7%	96.6%	89.8%	93.0%	95.2%	95.6%	95.4%
GPCRinterproFinal	67.4%	68.7%	68.0%	89.3%	75.0%	81.5%	88.0%	91.2%	83.3%
GPCRpfamFinal	68.2%	57.3%	62.3%	79.0%	58.5%	67.2%	72.1%	61.8%	66.5%
GPCRprintsFinal	72.8%	70.7%	71.8%	89.0%	72.3%	79.8%	85.3%	76.5%	80.7%
GPCRprositeFinal	50.5%	44.8%	47.5%	78.5%	53.5%	63.6%	68.0%	56.7%	61.9%

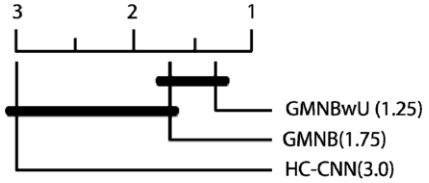

Fig. 9. Statistical tests for the three global-model hierarchical classifiers with $\alpha = 0.05$.

5.4. Computational results

The experiments comparing the predictive performance of the HC-CNN (with 1000 epochs for training), the GMNB and the GMNBwU are presented in Table 4.

To measure if there is any statistically significant difference between the hierarchical classification methods (measured by the hierarchical f-measure) being compared, we have employed the Friedman test with the post-hoc Shaffer's static procedure (with $\alpha = 0.05$) for comparison of multiple classifiers over many datasets as strongly recommended by [20]. The results of the statistical test are presented in a graphical way (as suggested in [21]) in Fig. 9 in order to summarize the pairwise comparisons made by the test. In Fig. 9 the bold horizontal lines connect the representations whose results are not found to be statistically significantly different.

The analysis of the results presented in Table 4 and Fig. 9 show that the GMNBwU is statistically significant better than the HC-CNN. It also shows that there is no significant difference between the GMNB and the HC-CNN and GMNBwU classifiers.

Although the HC-CNN is shown to be statistically worse than the GMBNwU this result should be taken with caution. As mentioned earlier this is (to the best of the authors' knowledge) the first experiment where with the same characteristics $\langle SPP, NMLP, D, GC\rangle$ are compared on a problem with the Single Path of Labels characteristic.

Considering that there is no statistically significant difference between the GMNB and the HC-CNN it would be interesting to verify in future research if, by adapting the usefulness criterion with the HC-CNN algorithm it would also make better predictions (as it is the same with GMNB and GMNBwU).

Furthermore, the authors are going to make the datasets, as they were used in the experiments, available to further motivate research in the field.

6. Conclusion

This paper presented a new global hierarchical classifier based on a competitive neural network, called HC-CNN, for tree-structured hierarchical classification problems. This classification approach has the advantage of evaluating the predictive performance of the entire class hierarchy, reporting a single result.

This novel algorithm was evaluated for the first time (to the best of the authors' knowledge) against two other hierarchical classification algorithms with the same core characteristics, namely the Global-Model Naïve Bayes (GMNB) and the Global-Model Naïve Bayes with Usefulness (GMNBwU).

The three algorithms were compared using eight protein function prediction datasets and their performance was evaluated with metrics tailored to hierarchical classification problems.

Our computational results have brought some new insights for future research, such as adapting the notion of usefulness of the GMNBwU to other global model hierarchical classification algorithms such as the HC-CNN. Another research direction is to perform different experiments with and without the usefulness criterion with the HC-CNN monitoring the learning rates of the neural network and the training amount of cycles in order to compare their predictive performances. In addition, use other evaluation measures for evaluating performance of algorithms.

As future research we intend to compare the performance of the methods present in this paper in different application domains, as text categorization, music genre classification, among others, and also to extend them (both the HC-CNN and the GMNB) to deal with hierarchical classification problems that have a DAG structure.

References

[1] A.A. Freitas, A.C.P.F. Carvalho, A tutorial on hierarchical classification with applications in bioinformatics, in: D. Taniar (Ed.), Research and Trends in Data Mining Technologies and Applications, in: Advances in Data Warehousing and Mining, IGI Publishing, Hershey, PA, USA, 2007, pp. 179–209 (Cap. 7).
[2] N. Holden, A.A. Freitas, A hybrid particle swarm/ant colony algorithm for the classification of hierarchical biological data, in: Proceedings of IEEE Swarm Intelligence Symposium, 2005, 2005, pp. 100–107.
[3] A. Sum, E. Lim, Hierarchical text classification and evaluation, in: Proc. of the International Conference on Data Mining (ICDM), California, USA, Nov. 2001, pp. 521–528.
[4] G. Xue, et al. Deep classification in large-scale text hierarchies, in: Proc. of the 31st Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR), Singapore, July 2008, pp. 619–626.
[5] J.J. Burred, A. Lerch, A hierarchical approach to automatic musical genre classification, in: Proc. of the 6th International Conference on Digital Audio Effects, 2003, pp. 8–11.
[6] C.N. Silla Jr, A.A. Freitas, A survey of hierarchical classification across different application domains, Data Mining and Knowledge Discovery 22 (1–2) (2011) 31–72.
[7] H.B. Borges, J.C. Nievola, Hierarchical classification using a competitive neural network, in: Proc. of the 8th International Conference on Natural Computation (ICNC), Vol. 1, IEEE Press, Piscataway, NJ, 2012, pp. 1–6.
[8] C.N. Silla Jr, A.A. Freitas, A global-model Naïve Bayes approach to the hierarchical prediction of protein functions, in: IEEE International Conference on Data Mining (ICDM), Miami, FL, USA, 2009, pp. 992–997.
[9] G. Valentini, True path rule hierarchical ensembles, in: J. Kittler, J. Benediktsson, F. Roli (Eds.), Proc. of the Eighth Int. Workshop on Multiple Classifier Systems, in: Lecture Notes in Computer Science, vol. 5519, Springer, 2009, pp. 232–241.
[10] Z. Barutcuoglu, et al., Hierarchical multi-label prediction of gene function, Bioinformatics 22 (7) (2006) 830–836.
[11] Y. Guan, et al., Predicting gene function in a hierarchical context with an ensemble of classifiers, Genome Biology 9 (2008) 1–18.
[12] C. Vens, et al., Decision trees for hierarchical multi-label classification, Machine Learning 73 (2) (2008) 185–214.
[13] F.E.B. Otero, A.A. Freitas, C.G. Johnson, A hierarchical multi-label classification ant colony algorithm for protein function prediction, Memetic Computing 2 (3) (2010) 182–196.
[14] S. Haykin, Neural Networks: A Comprehensive Foundation, 2nd ed., Prentice-Hall, 1999.
[15] T. Kohonen, The self-organizing map, Proceedings of IEEE 78 (9) (1990) 1464–1480.
[16] I.H. Witten, E. Frank, Data Mining: Practical Machine Learning Tools and Technique, second ed., Morgan Kaufmann, San Francisco, 2005.
[17] S. Kiritchenko, et al., Learning and evaluation in the presence of class hierarchies: application to text categorization, in: Proc. of the 19th Canadian Conf. on Artificial Intelligence, in: Lecture Notes in Artificial Intelligence, vol. 4013, 2006, pp. 395–406.
[18] J. Dougherty, R. Kohavi, M. Sahami, Supervised and unsupervised discretization of continuous features, in: Proc. of the 12th Int. Conf. on Machine Learning, 1995, pp. 194–202.
[19] Y. Yang, G.I. Webb, Discretization for Naïve-Bayes learning: managing discretization bias and variance, Machine Learning 74 (1) (2009) 39–74.
[20] S. García, F. Herrera, An extension on statistical comparisons of classifiers over multiple data sets for all pairwise comparisons, Journal of Machine Learning Research 9 (2008) 2677–2694.
[21] J. Demsar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine Learning Research 7 (2006) 1–30.
[22] H. Blockeel, et al. Hierarchical multi-classification, in: Proc. of the First SIGKDD Workshop on Multi-Relational Data Mining (MRDM), Edmonton, Canada, July, 2002. pp. 21–35.
[23] R.T. Alves, M.R. Delgado, A.A. Freitas, A.A. , Multi-label hierarchical classification of protein functions with artificial immune systems, in: Proc. Advances in Bioinformatics and Computational Biology, vol. 5167, 2008, pp. 1–12.