Multi-Label Hierarchical Classification using a Competitive Neural Network for Protein Function Prediction

Helyane Bronoski Borges  
UTFPR– Universidade Tecnológica Federal do Paraná  
PPG1a - Pontificia Universidade Católica do Paraná (PUCPR)  
Curitiba, Brazil  
Helyane@utfpr.edu.br

Julio Cesar Nievola  
PPG1a - Pontificia Universidade Católica do Paraná (PUCPR)  
Curitiba, Brazil  
nievola@ppgia.pucpr.br

Abstract—Hierarchical classification is a problem with applications in many areas as protein function prediction where the dates are hierarchically structured. Therefore, it is necessary the development of algorithms able to induce hierarchical classification models. This paper presents an algorithm for hierarchical classification using the global approach, called Multilabel Hierarchical Classification using a Competitive Neural Network (MHC-CNN). It was tested on some datasets from the bioinformatics field and its results are promising.

Keywords—Hierarchical Classification; Competitive Neural Network, Global Classifier

I. INTRODUCTION

Proteins are polymers of amino acids resulting from the translation of genetic information contained in DNA of cells. They are biomolecules more abundant and occur in great diversity, which can act as enzymes, antibodies, hormones, structural components, cell receptors among others. Due to this diversity of functions, proteins have great importance in almost all biological phenomena such as energy production, immune defense, activity of muscular contraction, neurochemical and reproduction [18].

We can cite two ways to describe the possible function provides by proteins. The first had the aim is characterize and standardize the description of the various enzymatic reactions that occur at the cellular level through a committee for the Enzymes Classification (EC). The second way is using ontology known as Gene Ontology (GO). The GO is a project that had aim of standardizing the representation of gene and gene product attributes across species and databases. The project provides a controlled vocabulary (ontologies) of terms that describe gene products in terms of their associated biological processes, cellular components and molecular functions.

GO structure is described as a graph where each node represents a GO term and the edges represent relationships between terms. The relationships used in the GO are directed and acyclic forming a directed acyclic graph (DAG). The ontologies resemble a hierarchy, where the conditions of the child node are more specialized than the condition of the parent node and a child node can have more than one father. For this semantics associated the edges is easy to see that this type of structure the general terms are closer to the root and the specific terms are closer to the leaves.

The protein function prediction can be treated as a problem of hierarchical classification where the aim of this area is to assign a biological function to a set of proteins.

Based on this problem, specific algorithms for hierarchical classification are being developed. This paper presents an algorithm for hierarchical multi-label classification based on the global approach using a competitive artificial neural network.

II. HIERARCHICAL CLASSIFICATION

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, classification can be divided into two types: flat classification (conventional) and hierarchical classification. Most of the problems cited in the literature involves the flat classification where each sample is associated with a class, which belongs to a set of classes, where each class is independent of the others.

The hierarchical classification differs from flat classification because the classes are organized in a hierarchy structured as a tree or a 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) can have one or more ancestors nodes.

One characteristic that differs flat classification from hierarchical classification refers to the type of prediction of classes in the hierarchy, which can be distinguished into two categories: mandatory leaf node (possible only in hierarchical classification) prediction and non-mandatory leaf node (possible only in hierarchical classification).

In mandatory leaf node prediction all examples should be associated with classes represented by leaf nodes. In the non-mandatory leaf node prediction there is no requirement that the prediction occurs at leaf nodes. Thus, the examples must be associated with classes that are represented by any
internal node of the class hierarchy along with their ancestors.

To explore hierarchical classification problems some solutions have been proposed, which can be divided into three main approaches: flat hierarchical classification, local hierarchical classification and global hierarchical classification [2]. These approaches describe how the classifiers are built and not a classification method, such as top-down approach that is often cited in literature as being one of the approaches.

1st Approach – Flat Hierarchical Classification

The flat hierarchical classification 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 ancestor and descendant, 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.

2nd Approach – Local Hierarchical Classification

As [4] explain, this approach can be divided into three kinds of classification: Local Classifier per Node Approach, Local Classifier per Parent Node Approach and Local Classifier per Level Approach.

1) Local Classifier per Node Approach

This approach is the most used in the literature [5], [6], [15], [14]. 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.

2) Local Classifier per Parent Node Approach

This approach consists in training each parent node of the class hierarchy a multi-class classifier [4]. This approach can’t be used in DAG structures without any strategy for creating classifiers parents since a child node can have more than one parent node.

3) Local Classifier per Level Approach

This approach consists in creating a multi-class classifier for each level of the hierarchy [4]. 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 that type of structure. Thus, a class can belong to more than one level in the hierarchy, which can bring redundancy among the classifiers. This approach also has the advantage that each classification model is built using a process of modularization as the local classifier per node approach. The same problem can occur if a node class has been wrongly propagated to the following levels of hierarchy.

In general, for local hierarchical classifiers, evaluation uses the top-down method. It starts predicting the class of the first level, and then the class predicted at the next level in the hierarchy is chosen only among the classes which have the previous one as its parent class. This process is repeated for classes at deeper levels [2].

3rd Approach – 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 [7], [9] and [3]. 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 advantages of this approach are that there is no need to train a large number of classifiers and the automatic manipulation of inconsistency in the prediction of classes.

Its main disadvantage is the increased complexity of the global classifier.

III. RELATED WORKS

The hierarchical classification is an approach that has been widely used in text mining since the 90s. Among the works in this context can be cited [24], [1], [12].

The field of bioinformatics presents several problems to be solved by hierarchical classification, but it is, unfortunately, still little explored. Some works have been published using this approach, specifically in the protein function prediction, but using an hierarchical tree structure [20], [21], [7], [22], [8] and [23].

In the work developed by Jensen et al. (2003), Laegreid et al. (2003) and [19] the authors dismissed the class hierarchy, i.e. the concept of ancestor and descendant was not applied. Thus, one can say that the problem of hierarchical classification was transformed into flat classification problem, where the conventional classification algorithms are used.

In [14], [15] and [24], created a local binary classifier for each node. Although the set of binary classifiers represent the hierarchy of classes, there is no guarantee that they produce consistent results. Moreover, if an error occurs in a class at a certain level, this could spread to the classes of the levels below. Thus, one can say that the results presented by the authors are questionable because the approach has these drawbacks.

In [3] developed a hierarchical classification model for the DAG structure using the global or big-bang approach. In this work the authors discuss three kinds of classification: single-label classification (SC), hierarchical single-label classification (HSC) and multi-label hierarchical classification (HMC). For the development of these classifiers the authors used the induction of decision trees
and showed how this model can be modified for use in hierarchical DAG structures.

These approaches are implemented in the CLUS and consist of generating a single decision tree for the whole hierarchy. This induction algorithm of decision tree is based on the framework Predictive Clustering Trees (PCT).

In [26] extended the Clus HMLC developed by [3], using other distance measures. The measures used by the authors were Jaccard distance, and SimGIC ImageCLEF. Such measures have been implemented in CLUS.

In [9] constructed a hierarchical classification model called Hierarchical Multi-Label Classification with an Artificial Immune System (MHCAIS), which uses concepts of an Artificial Immune System (AIS). This hierarchical classifier aims to discover knowledge represented as rules if-then.

The author presents two versions of MHCAIS: global and local. The local version builds a classifier for each class, while in the global version a single classifier is generated to distinguish all classes of the application.

In [27] develop a new Ant Colony Optimisation algorithm, named hAnt-Miner, for the hierarchical classification problem of predicting protein functions using the GO. The algorithm proposed discovers a single global classification model in the form of an ordered list of IF-THEN classification rules which can predict GO terms at all levels of the GO hierarchy, satisfying the parent-child relationships between GO terms.

IV. MULTI-LABEL HIERARCHICAL CLASSIFICATION USING A COMPETITIVE NEURAL NETWORK (MHC-CNN)

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 [10], 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 MHC-CNN algorithm proposed in this paper is based on a Competitive Artificial Neural Network [10], [13]. Figure 1 shows a neural network model. This network consists of two layers of neurons. The input layer is connected to an input vector data set. The term “Input neurons” defined in this figure represents all instances of entry, according to the interpretation of the data set. The processing layer or output layer, which in a competitive network is the output mapping, represents the hierarchy of classes, where each neuron is connected to its ancestors and possibly descendants.

In the traditional competitive network, for example, the Kohonen network, the neurons of the output layer are arranged in a grid network [13], which can be rectangular, hexagonal, among others, and they represent the network topology. In MHC-CNN algorithm the topology is a DAG, where each neuron is connected with its ancestors (parents) and descendent (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.

IV. MULTI-LABEL HIERARCHICAL CLASSIFICATION USING A COMPETITIVE NEURAL NETWORK (MHC-CNN)

Figure 1. Example of the MHC-CNN.

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 distance measures.

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 weight are randomly initialized. The training process of the network is divided into three phases, as in a traditional competitive network: Competition, Cooperation and Adaptation

A. Competition

An instance \( e_i \) of input data set \( DB_{\text{Train}}=\{e_1, e_2, \ldots, e_q\} \) of dimension \( q \) is selected. Each element of \( DB_{\text{Train}} \) consists of attributes \( e_i=\{a_{i1}, a_{i2}, a_{i3}, \ldots, a_{in}\} \), where \( a_{ij} \) is the \( j \)-th attribute, \( 1 \leq j \leq l-1 \) is the number of input attributes, and \( i \in N: 1 \leq i \leq q \). \( a_{ij} \) represents the class attribute.

The neural network is created according to information obtained in the input data: the number of input neurons is equals the number of input attributes, the output layer neurons are represented by \( NN=\{n_1, n_2, n_3, \ldots, n_b\} \), where \( b \) is the number of neuron of each class that exists in the class hierarchy. Each neuron of the output layer consists of synaptic weights such as \( n_i=\{p_{1i}, p_{2i}, p_{3i}, \ldots, p_{l(1)i}\} \) 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, distance measures are used. The measure chosen was the Euclidean distance (Equation 1).

\[
d_{ik} = \sqrt{\sum_{i=1}^{l(1)} (e_{ij} - n_{ijk})^2}
\]
$e_i$ is the input instance, $n_{ijk}$ is the $k$-th neuron of output layer, $l$ is the number of attributes and $k$ is amount of output layer neurons.

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

$$nv = \arg \min(d_i)$$

(2)

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

When an instance has more than one class, ie multi-label prediction, the classifier selects more winners. For example, if the instance $e_i$ has three classes the algorithm selected three winners.

B. Cooperation

The winner tends to excite their ancestors 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 MHC-CNN algorithm, the neighborhood criterion is determined from the relationship between the neuron and their 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.

C. Adaptation

The third phase is the adaptation process of synaptic weights. Since 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 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 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 of this instance, because the class is predicted incorrectly. That is, during training, the algorithm adjusts the weights of the neuron and their ancestors, making the comparison by the identifier of the class of each input instance with the desired output.

Equation 3 shows the equation for updating the weights neurons.

$$n_{ijk}(t+1) = \begin{cases} 
(n_{ijk} + \epsilon_i(t) A_p(t) Dist(t)), & \text{if the true class} \\
(n_{ijk} - \epsilon_i(t) A_p(t) Dist(t)), & \text{if the incorrect class}
\end{cases}$$

(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 $k$.

$A_p$ is the learning rate for the instant of time $(t+1)$ which is obtained by the formula represented in Equation 4.

$$A_p(t) = (\mu_i - \mu_f) \cdot e^{\frac{t_{current}}{C}}$$

(4)

where $\mu_i$ is the initial learning rate, $\mu_f$ 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 and is obtained by the formula shown in Equation 5.

$$Dist(t) = 1/(k+1)$$

(5)

where $k$ is the distance in node of winner neuron and the ancestral neuron that will the weights adjusted.

The updating of the weights of ancestor neurons follows the same method shown in the above equations (Equations 3, 4 and 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. Again, the procedure is repeated until the execution of all epochs.

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

The test of the algorithm is done similarly 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 $x_i=[a_{i1}, a_{i2}, a_{i3}, \ldots, a_{ig}]$, $\forall i \in \{1, \ldots, g\}$. $a_i$ represents the class attribute and $l$ is amount attributes.

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

V. EVALUATION MEASURES

Two evaluation measures were used to report the predictive performance of the samples: distance-based depth-dependent measure [7] and hierarchy based measures [11], [12]. The choice of these measures was made to assess the performance of the classification in different ways.

A. Distance-based Depth-Dependent Measures

When evaluating the result of a hierarchical prediction three situations may occur: correct prediction, partially correct prediction and incorrect prediction. To better understand these situations, each one will be exemplified.
TABLE I. TRAINING OF ALGORITHM MHC-CNN

<table>
<thead>
<tr>
<th>INPUT ALGORITHM</th>
<th>- Training data set $DB_{Tren}={e_1 \ e_2 \ e_3 \ ... \ e_q}$ of dimension $q$.</th>
</tr>
</thead>
<tbody>
<tr>
<td>STEP 1: INITIALIZE</td>
<td>- The initial learning rate $\mu_i$ and the final learning rate $\mu_f$.</td>
</tr>
<tr>
<td></td>
<td>- Determine the number of cycles $ep$.</td>
</tr>
<tr>
<td></td>
<td>- Initialize the synaptic weights of the network: $NN=[n_1 \ n_2 \ n_3 \ ... \ n_b]$ where $b$ is the number of class that exists in the class hierarchy.</td>
</tr>
<tr>
<td></td>
<td>- 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.</td>
</tr>
<tr>
<td>STEP 2: STOPPING CRITERION</td>
<td>- Number of cycles.</td>
</tr>
<tr>
<td>STEP 3: TRAINING</td>
<td>- Select an instance $e_i$ of the input data set $DB_{Tren}={e_1 \ e_2 \ e_3 \ ... \ e_q}$.</td>
</tr>
<tr>
<td></td>
<td>COMPETITION:</td>
</tr>
<tr>
<td></td>
<td>- Calculate the distance between the instance $e_i$ with the neurons of the network $NN=[n_1 \ n_2 \ n_3 \ ... \ n_b]$.</td>
</tr>
<tr>
<td></td>
<td>- Finding neuron $n_j$ who had a shortest distance, which are considered winners neurons.</td>
</tr>
<tr>
<td></td>
<td>COOPERATION:</td>
</tr>
<tr>
<td></td>
<td>- Find the ancestors of the neuron $j$.</td>
</tr>
<tr>
<td></td>
<td>ADAPTATION:</td>
</tr>
<tr>
<td></td>
<td>- Update the synaptic weights of neurons and their ancestors.</td>
</tr>
<tr>
<td></td>
<td>- If the class true equal the predict class $Dist(t)=1$.</td>
</tr>
<tr>
<td>STEP 4: UPDATE</td>
<td>- Update the learning rate $Ap(t)$.</td>
</tr>
<tr>
<td></td>
<td>Start the STEP 1.</td>
</tr>
<tr>
<td>OUTPUT ALGORITHM</td>
<td>- Adequate weights set.</td>
</tr>
</tbody>
</table>

TABLE II. TEST OF ALGORITHM MHC-CNN

<table>
<thead>
<tr>
<th>INPUT ALGORITHM</th>
<th>- Test data set $DB_{Test}={x_1 \ x_2 \ x_3 \ ... \ x_g}$ of $g$ is amount of test instance.</th>
</tr>
</thead>
<tbody>
<tr>
<td>STEP 1: STOPPING CRITERION</td>
<td>- Until all instances have been selected and tested.</td>
</tr>
<tr>
<td>STEP 2: TEST</td>
<td>- Select an instance $x_i$ of the input data set $DB_{Test}={x_1 \ x_2 \ x_3 \ ... \ x_g}$.</td>
</tr>
<tr>
<td></td>
<td>- Calculate the distance between the instance $x_i$ with the neurons of the network $NN=[n_1 \ n_2 \ n_3 \ ... \ n_b]$.</td>
</tr>
<tr>
<td></td>
<td>- Finding neuron $n_j$ who had a shortest distance, which are considered winners neurons.</td>
</tr>
<tr>
<td></td>
<td>- Find the ancestors of the neuron $j$.</td>
</tr>
<tr>
<td></td>
<td>- Evaluation prediction.</td>
</tr>
<tr>
<td></td>
<td>- Assign result to the confusion matrix.</td>
</tr>
<tr>
<td>OUTPUT ALGORITHM</td>
<td>- Accuracy rate obtained by the algorithm.</td>
</tr>
</tbody>
</table>

1) First Possibility: Correct Prediction

There are two types of possible correct prediction. The first one occurs when the algorithm hits the full path, being the predicted class equal to the true class as shown in Figure 3.

![Figure 2. Example of Correct Prediction – 1st Possibility.](image)

The second case occurs when the predicted class is in the full path of the correct one, but it is more specific. Figure 3 shows this possibility: the true class is represented by the node "3" in the tree, and the algorithm predicts the node "3.1.1". This case is considered a correct prediction.

![Figure 3. Example of Correct Prediction – 2nd Possibility.](image)

2) Second Possibility: Partially Correct prediction

An example of a partially correct prediction is shown in Figure 4. In this case, the true class is represented by the node "2.1.1" but the algorithm predicts the class represented by the node "2.1". Observe that the node's parent node is predicted true. Although the predicted class is in the correct path it stops before finding the true class in the tree, not providing the full specificity of it. Therefore, one can say that the prediction was partially correct, because the algorithm was on the correct path of prediction, it just occurred before hitting the full specification.

An instance whose class is predicted at higher levels tends to be more easily classified than a class in deeper levels. Thus, the algorithm considers it a partial prediction, being based on the level of class, that means, classes at levels closer to the root have higher importance than classes at deeper levels.

In this example, the class is predicted on the second level and true class is at the third level. Then, indices of
importance are assigned inversely proportional to the level of the classes, i.e., the class "2.1" is replaced by an index two times larger than the class "2.1.1". Equation 6 shows the formula for this calculation.

\[ 1p + 2p + ... + np = 1 \]  

(6)

where \( p \) is the index and \( n \) is the level in the hierarchy. The correct prediction rate is the sum of weights of classes correctly predicted, i.e. the predicted class and its ancestor classes. Applying the formula to this example, one obtains \( p=0.16 \). Thus, the weight of class "2.1.1" is 0.16, and the class "2" is 0.5. Then the hit rate of this sample is 83%.

![Figure 4](image-url)

**Figure 4. Example of Partially Correct Prediction.**

3) **Third Possibility: Incorrect Prediction**

There is an incorrect prediction when the predicted class totally misses the path prediction as shown in Figure 5. It is observed that the true class is represented by the node "2.1.1", however, the algorithm predicts incorrectly the class as "3.2".

![Figure 5](image-url)

**Figure 5. Example of Incorrect Prediction.**

**B. Hierarchy based Measure**

This kind of measure was developed by Kiritchenko et al. (2004) and uses concepts of ancestral and descendant classes. The author proposes two evaluation measures: hierarchical precision and hierarchical recall which take into account the hierarchical relationships [11], [12]. 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 recall (Equation 7), the number of common ancestors is divided by the number of ancestors of the true class.

\[ hR = \frac{|A(C_t) \cap A(C_p)|}{A(C_t)} \]  

(7)

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

To calculate the precision (Equation 8), the number of common ancestors is divided by the number of ancestors of the predict class.

\[ hP = \frac{|A(C_t) \cap A(C_p)|}{A(C_p)} \]  

(8)

These measures can be used to calculate an extension of the F-measure, named the hierarchical F-measure (Equation 9).

\[ hF = \text{Measure} = \frac{(\beta^2 + 1) * hR * hP}{\beta^2 * hR + hP} \]  

(9)

where \( \beta \) is the importance given to precision and recall.

**VI. EXPERIMENTERS AND RESULTSSSS**

The experimenters are realized in five databases were the same used by [3] in their experiments. The Table 3 shows the characteristic the databases selected. For the all experiments 2/3 of the examples were used for training and 1/3 for testing (hold-out procedure).

**TABLE III. CHARACTERISTICS OF DATABASES**

<table>
<thead>
<tr>
<th>Database</th>
<th>Amount Samples</th>
<th>Amount Attributes</th>
<th>Amount Class</th>
<th>Amount Min/Max Class per Sample</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cellcycle</td>
<td>3751</td>
<td>77</td>
<td>4125</td>
<td>3/28</td>
</tr>
<tr>
<td>Church</td>
<td>3749</td>
<td>27</td>
<td>4125</td>
<td>3/28</td>
</tr>
<tr>
<td>Derisi</td>
<td>3719</td>
<td>63</td>
<td>4119</td>
<td>3/28</td>
</tr>
<tr>
<td>Einsen</td>
<td>2418</td>
<td>79</td>
<td>3573</td>
<td>3/28</td>
</tr>
<tr>
<td>Spo</td>
<td>3697</td>
<td>80</td>
<td>4119</td>
<td>3/28</td>
</tr>
</tbody>
</table>

In addition, the attributes missing in the database were imputed. The criterion used for imputation of missing attribute values was to calculate the arithmetic average of the closer ancestor classes of class to which belongs the sample that has the missing attribute. After, all datasets were normalized using the approach Min-Max.

The initial learning rate and final learning rate used in the experiments were 0.1 and 0.01, respectively. The neural network synaptic weights were generated randomly, according to an uniform distribution. Evaluation of the classification was made taking into account all levels of the hierarchy.
The results at the MHC-CNN are compared with the Clus-HMC and Clus-HSC using distance-based depth dependent measure and hierarchy-based measures.

For the Clus experiments were selected some thresholds to compare the performance the algorithms. Table 4 presents the results obtained with 500 epochs for training the neural network and the thresholds 50 and 100. The term $DTi$ indicates the databases in the order show of the Table 3, $Dist$ indicates distance measure and $hF$ indicate $hF$-Measure.

The results were statistically compared using the Friedman and Nemenyi test [16] [17] to verify whether there is statistical higher that others algorithms. Clus-HMC and Clus-HSC using performances of the algorithms Friedman and Nemenyi test [16] [17] to verify whether there is statistical higher that others algorithms.

Based on the results of this test, it is observed that the Clus-HMC and Clus-HSM using $hF$ with threshold equal 50 is statistical higher that others algorithms.

### VII. CONCLUSION

This paper presented a new global hierarchical classifier based on a competitive neural network, called MHC-CNN, for prediction of structured data in DAG.

This classification approach has the advantage of evaluating the predictive performance of the entire class hierarchy, reporting a single result. The results of the predictions were assessed using two approaches to hierarchical classification measures: distance-based depth-dependent measure and measured based on hierarchical.

Although the results with the MHC-CNN present results statistically lower in some cases, the classifier can be a promising alternative to deal with hierarchical multi-label classification of biological data.

Other experiments can be done using other learning rates of the neural network and the amount of cycles to train the neural network in order to compare the result of prediction.

In addition, as future work, we intend to using others measures to a updating the synaptic weights of the neural network and adapt the classifier to use others approaches for the neural network train.

### REFERENCES


