Hierarchical Classification of Gene Ontology-based Protein Functions with Neural Networks

Ricardo Cerri
Department of Computing
Federal University of São Carlos
Rodovia Washington Luís, km 235
São Carlos, SP, Brazil
Email: cerri@dc.ufscar.br

Rodrigo C. Barros
Faculdade de Informática
Pontifícia Universidade Católica do RS
Av. Ipiranga, 6681
Porto Alegre, RS, Brazil
Email: rodrigo.barros@pucrs.br

André C. P. L. F. de Carvalho
Departamento de Ciência da Computação
Universidade de São Paulo
Av. Trabalhador Sao-Carlense, 400
São Carlos, SP, Brazil
Email: andre@icmc.usp.br

***Abstract*—Hierarchical Multi-label Classification (HMC) is a classification task where classes are organized in a hierarchical taxonomy, and instances can be simultaneously classified in more than one class. This paper investigates the HMC problem of classifying proteins in functions organized according to the Gene Ontology hierarchical taxonomy. This is a complex task, since the Gene Ontology hierarchy is organized as a Directed Acyclic Graph with thousands of classes hierarchically represented. We propose a neural network-based method to incorporate label-dependency during learning. The experimental results show that the proposed method achieves competitive results when compared to the state-of-the-art methods from the literature.**

I. Introduction

In traditional classification problems, an instance $\mathbf{x}_i \in X$ can be classified in only one class $c_j \in C$. Nonetheless, there are more complex classification tasks where an instance can be simultaneously classified into a set of classes $C_j \in C$. One of this tasks is Hierarchical Multi-Label Classification (HMC), in which an instance can be classified into a set of classes that are previously organized as a hierarchy, with subclasses and superclasses. In this hierarchy, superclass relationships are represented by a partial order $\prec_h$, *i.e.*, for all $c_1, c_2 \in C, c_1 \prec_h c_2$ if and only if c_1 is a superclass of c_2.

Protein Function Prediction (PFP) is a common HMC application, once proteins perform many important functions within an organism. Proteins are related to biochemical reactions, cell signaling, structural, and mechanical functions [1], to name a few. Given that protein functions are hierarchically related, PFP can be considered a typical HMC problem.

This study uses the Gene Ontology (GO) hierarchy to investigate the PFP problem. In the GO, classes are organized as a Directed Acyclic Graph (DAG) hierarchy of terms, in which each term corresponds to a protein function. The hierarchy comprises three ontologies covering different domains, each one with thousands of classes: *cellular components*, *biological processes*, and *molecular functions* [2]. Figure 1 illustrates a small part of the GO taxonomy.

The PFP task using the GO taxonomy is a very challenging problem. As we traverse the hierarchy down to the leaves, making accurate predictions becomes more difficult since terms have fewer and fewer positive instances. In addition, instances can be classified simultaneously into two or more paths of the hierarchy, and a term can have more than one parent node. We work with the "is-a" relation, which forms the basic structure of the GO. Thus, A is a B means that term A is a subtype of term B. Also, classifying an instance into class c_j means that we are classifying it into all superclasses of class c_j. This is the multiple inheritance interpretation, which is the correct interpretation when working with the GO [2].

This paper has the following original contributions. First, it proposes a method called **H**ierarchical **M**ulti-Label **C**lassification with **L**ocal **M**ulti-**L**ayer **P**erceptron (HMC-LMLP), which associates an MLP network to each level of the hierarchy, where each MLP is responsible for the predictions in its associated level. HMC-LMLP employs the Local Classifier per Level (LCL) [3] strategy to use local information within each level of the hierarchy. By using one classifier per level, the classification problem is not decomposed into a large number of sub-problems. The use of many classifiers per level may result in the application of over-specific local information, loss of important information, and loss of label dependency during training [3]. A preliminary version of HMC-LMLP was reported in [4]. Second, differently from the preliminary version, the labels of the training instances are used here as part of the input to train each MLP. Therefore, when training an MLP for level l, the feature vector of an instance is augmented with its classes for the level $l-1$. This modification aims to enforce the use of label dependencies between classes. We also propose a second version which ignores the labels associated with the classes to augment the size of the feature vectors, a baseline version that allows us to examine whether the use of the labels to augment the feature vectors results in performance improvement. Finally, we analyze the application of the LCL strategy to DAG-structured class hierarchies, adapting the DAG structure datasets for these experiments.

The remainder of this paper is organized as follows. Section II presents related work regarding HMC for PFP. The proposed HMC-LMLP method is detailed in Section III, whereas a thorough empirical analysis is carried out in Section IV. A deeper discussion on the results is performed in Section V, and the final considerations and future research directions are presented in Section VI.

II. Related Work

This section discusses recent HMC methods reported in the literature that employ machine learning for protein and gene function prediction.

978-1-4799-1959-8/15/$31.00

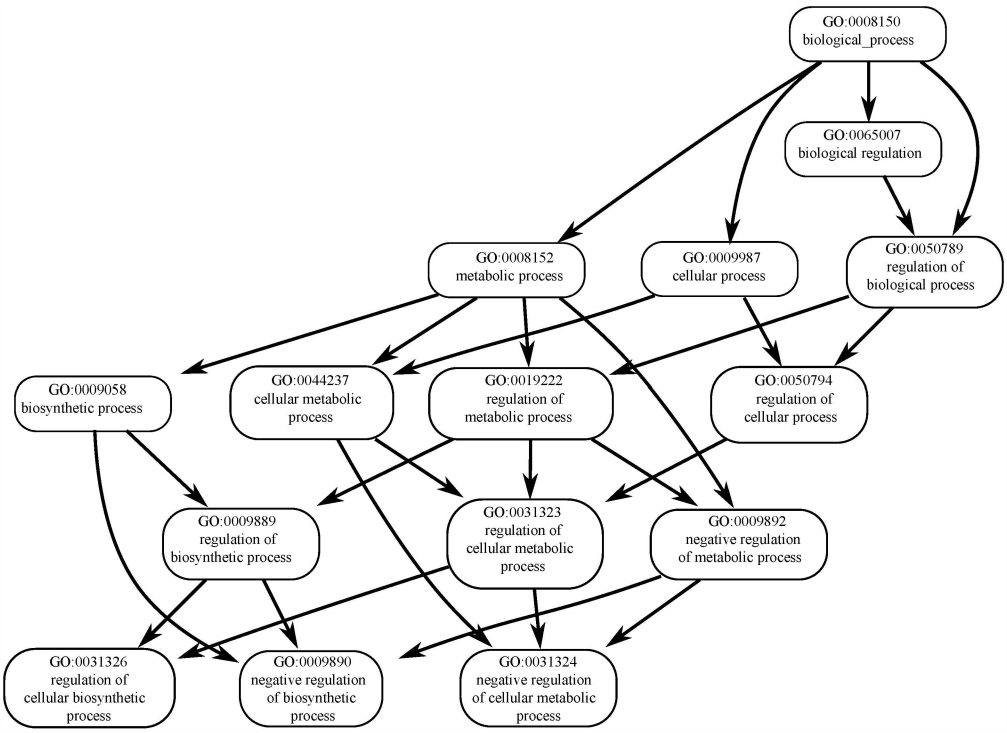


Fig. 1: Part of the Gene Ontology Hierarchical Taxonomy. (Adapted from Ashburner et. al. [2])

In Vens et al. [5], three methods based on the concept of Predictive Clustering Trees (PCT) were investigated. The authors proposed the Clus-HMC method that induces a single decision tree to cope with the entire classification problem. They compared its performance with two methods. The first one, Clus-SC, induces an independent decision tree for each class, ignoring the relationships between classes. The second one, Clus-HSC, explores the hierarchical relationships between the classes to induce a decision tree for each class.

Alves et al. [6] proposed a global method using Artificial Immune Systems (AIS) for the generation of HMC rules. The method is divided into two basic procedures: Sequential Covering (SC) and Rule Evolution (RE). The SC procedure iteratively calls the RE procedure until every (or almost every) training instance (antigens) is covered by the discovered rules. The RE procedure evolves classification rules (antibodies) that are employed to classify the instances. The best antibody is added to the set of discovered rules.

In the work of Otero et al. [7], the authors proposed a method using Ant Colony Optimization (ACO). The method discovers classification rules, where an ACO algorithm is employed to optimize the antecedents of the rules. A sequential covering procedure is applied to create classification rules that cover most of the training instances. The method is initialized with an empty set of rules, and a new rule is added to the set while the number of instances not covered by any rule is higher than a given threshold.

Cesa-Bianchi and Valentini [8] investigated the synergy between different LCN-based strategies related to gene function prediction task in FunCat annotated genes. They integrated kernel-based data fusion tools and ensemble algorithms with cost sensitive HMC methods [9], [10]. The authors defined synergy as the improvement in the prediction accuracy, considering any evaluation measure, due to the use of concurrent learning strategies. The synergy is detected when the combined action of two strategies achieves better correct classification rates than the average of the correct classification of the two strategies used separately [8].

Kordmahalleh et. al. [11] proposed CAM-HMC, an evolutionary algorithm which applies evolutionary crowding niching and adaptive mutation to evolve antecedentes of HMC rules. During the evolutionary process, the authors defined a new distance measure d for the competition between parents p and offspring c. If $|d(p_1, c_1) + d(p_2, c_2)| \leq |d(p_1, c_2) + d(p_2, c_1)|$ the competition is between (p_1, c_1) and (p_2, c_2). Otherwise, the competition is between (p_1, c_2) and (p_2, c_1). The individuals with highest fitness are kept in the population.

The work of Stojanova et. al. [12] proposed a method which considers autocorrelation in HMC, *i.e.*, the statistical relationships between the same variable on different but related instances. The method is called Network Hierarchical Multi-label Classification (NHMC), and builds a generalized form of decision trees using the PCT framework, just like Clus-HMC. During training, NHMC uses both the features of the instances, and the autocorrelations between instances. The autocorrelations were modeled as a network, which is exploited by the method during the learning phase.

A genetic algorithm was proposed by Cerri et. al. [13]. The method, called HMC-GA, evolves the antecedents of HMC rules, containing both propositional and relational tests. The consequents of the rules are deterministically obtained based on the classes of the training instances covered by the

antecedents. Each generated rule is able to classify instances into two or more paths of the GO taxonomy.

Bi and Kwok [14] proposed a method that uses the Mandatory Leaf Node Prediction strategy (MLNP) [3]. The method uses hierarchy information, and the problem is formulated as finding the multiple labels with the largest posterior probability over all the labels. The authors extended the nested approximation property [15] to deal with HMC problems structured as DAG, which was solved using a greedy algorithm (MAS).

In this paper, we make use of four methods reviewed in this section: Clus-HMC (which is considered to be the state-of-the-art method in the literature), Clus-HSC and Clus-SC. We also make use of the Ant Colony Optimization-based method *hm*Ant-Miner, which obtained competitive results with Clus-HMC. We chose these methods because they were all applied to the same datasets used in our experiments. In addition, they produce the same type of output provided by our proposed approach, and they have their code available for downloading, providing a fair base for comparison.

III. HMC-LMLP

HMC-LMLP divides the learning process into a number of steps, combining MLPs individually trained for each level of the class hierarchy. The rationale is that each MLP learns something different from each other, breaking down the complex learning process into simpler processes.

In HMC-LMLP, the MLPs extract local information from the instances at each level, which can be useful in the classification of unlabeled instances. Our hypothesis is that different patterns can be extracted from instances in different hierarchical levels. Note that, whereas many different classifiers could be employed, we decided to choose neural networks because of the simplicity in associating a class per output neuron. Therefore, generating a multi-label prediction for a given instance is done in a straightforward fashion.

In this section, we present the proposed HMC-LMLP version, called HMC-LMLP-Comp. This version employs, at each level, the true labels of the instances from the previous level to augment the feature vectors. The baseline version is named HMC-LMLP-NoComp, since it only uses the original feature vectors to train an MLP at each level. For simplicity, all networks used in this study have a single hidden layer.

A. HMC-LMLP-Comp

Figure 2a illustrates the architecture of HMC-LMLP-Comp and its training process for a hierarchy with three levels. Level 1 is the first level below the root, having two classes. Also in the figure, T_l are the true class labels associated to the instances at the level l; $\mathbf{X}^l$ represents the instances assigned to classes from the level l; h_l and O_l are, respectively, the hidden layer and output layer of the MLP network associated with level l. The matrices $\mathbf{W}_{1l}$ and $\mathbf{W}_{2l}$ represent, respectively, the weights connecting the input attributes and the neurons in the hidden layer, and the neurons in the hidden and output layers of the MLP associated with level l.

The neural network associated with the first level is trained with all training instances ($\mathbf{X}^1$), since all instances are assigned to the classes from the first hierarchical level. At the second level, the MLP input is now the training instances that are assigned to the classes belonging to level 2 ($\mathbf{X}^2$), combined with their true assigned classes in the first level. The advantage of using the augmented feature vector for training each MLP is the incorporation of label dependency. This process is repeated for each level of the hierarchy.

As can be observed in the figure, the training process of HMC-LMLP-Comp, for each hierarchical level, can be performed in parallel, which can speed up the training process.

B. HMC-LMLP-NoComp

In HMC-LMLP-NoComp, an individual MLP is trained for each hierarchical level without employing the class labels to augment the feature vectors of the training instances. Figure 2b illustrates the HMC-LMLP-NoLabels architecture and the training process. Similarly to HMC-LMLP-Comp, the training process in each level can be performed in parallel.

C. Obtaining final predictions

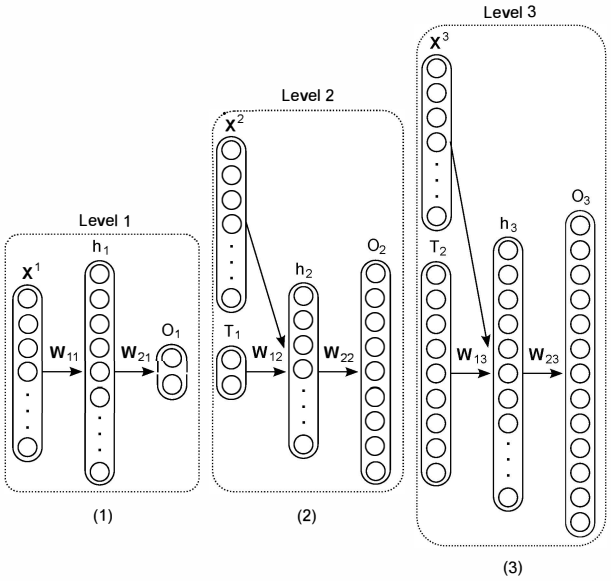
In the test phase of HMC-LMLP-Comp (i.e., when predicting a test instance), the true labels are not available. Thus, a top-down strategy is employed, in which the feature vectors that are used for training the MLP at level l are augmented with the output provided by the MLP in the level $l-1$. Due to this network dependency, the testing process of HMC-LMLP-Comp cannot be performed in parallel. Instead, the predictions for each level have to be obtained sequentially. In HMC-LMLP-NoComp, the testing phase is performed by feeding all instances into all MLPs at every level. Each MLP then provides independent predictions for the instances at each level. Thus, both training and testing phases, for each level, can be performed in parallel.

After obtaining the MLP outputs values, they are tested against thresholds in order to define the predictions for each level. If the output of a given neuron j is greater than or equal to a given threshold, the instance that is being classified is assigned to class c_j. The final classification from HMC-LMLP is given by a binary vector $\mathbf{v}$ of size $|C|$, where C is the set of all classes in the hierarchy. If the output value of neuron j is greater than or equal to a given threshold, the value 1 is assigned to position $\mathbf{v}_j$. Otherwise, the position is set to 0. Since the activation function that is used in the neurons is the logistic sigmoid function, the output values range between 0 and 1. Thus, we can make use of threshold values also ranging from 0 to 1. The larger the threshold, the lower the number of predicted classes. Conversely, the lower the threshold, the larger the number of predicted classes.

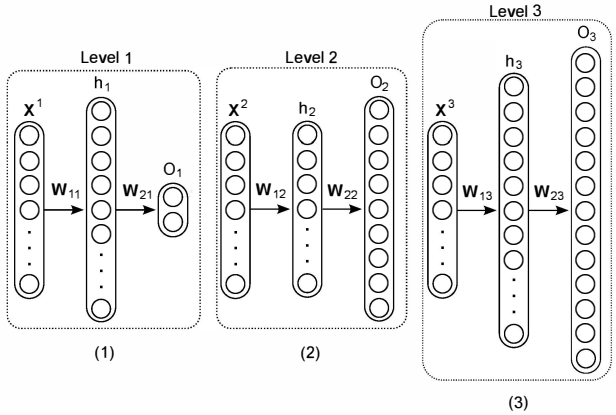
After testing the predictions against the thresholds, there could be classification inconsistencies, *i.e.*, when a subclass is predicted but its superclass is not. This problem is intrinsic to the LCL strategy [3], and for addressing this matter we employ a post-processing phase that removes all predicted classes whose superclasses were not predicted as well.

D. Computational complexity

Each MLP used in HMC-LMLP-Comp has a complexity of $\mathcal{O}(W_l)$, with W_l being the number of weights and biases of the MLP associated with level l. Let A be the number of attributes



(a) Example of the HMC-LMLP-Comp architecture. (1) Training an MLP at the first level; (2) Using the true classes of the training instances in level 1 to augment the feature vector of the instances responsible for training the MLP at the second level; (3) Using the true classes of the training instances in level 2 to augment the feature vectors which are used for training the MLP at the third level.



(b) Example of the HMC-LMLP-NoComp architecture. (1) Training an MLP at the first level; (2) Training an MLP at the second level; (3) Training an MLP at the third level.

Fig. 2: Example of HMC-LMLP-Comp and HMC-LMLP-NoComp architectures for a three-level hierarchy.

in the dataset, H_l be the number of hidden neurons of the MLP associated with level l, and O_l be the number of output neurons of the MLP associated with level l. We can thus define W_1 as $(A+1) \times H_1 + (H_1+1) \times O_1$. From the second level onwards, W_l is defined as $(O_{l-1} + A + 1) \times H_l + (H_l + 1) \times O_l$. The training cost of each MLP associated with level l in HMC-LMLP-Comp is then $\mathcal{O}(W_l \times m_l \times n)$, with m_l being the number of training instances assigned to classes belonging to level l, and n the number of training epochs. In HMC-LMLP-NoComp, the computational cost is smaller, since the class labels are not used to augment the feature vectors.

IV. Experiments

In this section, we present the experiments that were carried out to compare the prediction performance achieved by HMC-LMLP's versions and the state-of-the-art HMC algorithms. We also present the datasets, parameters, and the evaluation measure that were employed in the experiments.

A. Datasets

We make use of ten freely available[1] datasets related to protein function prediction. These datasets are related to issues like phenotype data and gene expression levels. Table I presents the main characteristics of the training, validation, and test datasets. A description of each dataset can be found in [5].

Considering that there is no level definition in DAG structures (a class can be located at different levels depending on which hierarchical path is chosen from the root node to the class), we defined the depth of a class in a DAG structure as the deepest path from the class to the root node. This is necessary for the application of HMC-LMLP, since the method requires a clear separation of classes in levels. We chose the deepest path as the definition of depth because it guarantees that when a class is located in a level l, all its superclasses will be located in levels shallower than l. With this depth definition, each hierarchy ended up with 13 levels.

We performed a pre-processing step before running HMC-LMLP over these datasets, in which all nominal attribute values were transformed into numeric values using the one-attribute-per-value approach. In this paper, instead of using 0s and 1s, the nominal attributes were assigned -1 (absence) and 1 (presence), which are better suited for training neural networks [16]. The attributes were then standardized (mean 0 and variance 1). Additionally, all missing values for nominal and numeric attributes were replaced, respectively, by their mode and mean values.

B. Evaluation Measure

The outputs of HMC-LMLP, for each class, are real values between 0 and 1. The same is true for the literature methods. Thus, in order to obtain the final predictions, a threshold value was further employed. When classifying an instance, if the corresponding output value for a given class is greater than or equal to the threshold, the instance is assigned to the class, otherwise it is not.

The choice of the "optimal" threshold value is a difficult task, since low threshold values lead to many classes being assigned to each instance, resulting in high recall and low

[1] http://www.cs.kuleuven.be/~dtai/clus/hmcdatasets.html

TABLE I: Summary of the datasets: number of attributes ($|A|$), number of classes ($|C|$), number of classes per level (Classes per level), total number of instances (Total) and number of multi-label instances (Multi).

Dataset	$\|A\|$	$\|C\|$	Classes per level	Training		Valid		Test	
				Total	Multi	Total	Multi	Total	Multi
Cellcycle	77	4122	33/155/394/597/929/779/631/335/171/63/21/5/9	1625	1625	848	848	1278	1278
Church	27	4122	33/155/394/597/929/779/631/335/171/63/21/5/9	1627	1627	844	844	1278	1278
Derisi	63	4116	33/155/394/596/927/778/630/334/171/63/21/5/9	1605	1605	842	842	1272	1272
Eisen	79	3570	33/149/360/524/786/679/539/271/141/55/19/5/9	1055	1055	528	528	835	835
Expr	551	4128	33/155/394/599/932/780/631/335/171/63/21/5/9	1636	1636	849	849	1288	1288
Gaschl	173	4122	33/155/394/597/929/779/631/335/171/63/21/5/9	1631	1631	846	846	1281	1281
Gasch2	52	4128	33/155/394/599/932/780/631/335/171/63/21/5/9	1636	1636	849	849	1288	1288
Pheno	69	3124	33/145/332/489/670/568/460/236/114/49/18/4/6	653	653	352	352	581	581
Seq	478	4130	33/155/394/599/932/780/633/335/171/63/21/5/9	1692	1692	876	876	1332	1332
Spo	80	4116	33/155/394/596/927/778/630/334/171/63/21/5/9	1597	1597	837	837	1263	1263

precision. On the other hand, large threshold values lead to very few instances being classified, resulting in high precision and low recall. To deal with this problem, we make use of precision-recall curves (PR-curves) as the evaluation measure for the experiments. To obtain a PR-curve, different thresholds between [0,1] are applied to the outputs of the methods, and thus different values of precision and recall are obtained, one for each threshold value. Each threshold then represents a point within the PR-space. The union of these points form a PR-curve, and the area under the curve is calculated. All methods are thus compared based on their areas under the PR-curves.

More specifically, we employed the area under the average PR-curve ($AU(\overline{PRC})$). Given a threshold value, a precision-recall point ($\overline{Prec}, \overline{Rec}$) in the PR-space can be obtained through Equations (1) and (2). They correspond to the micro-average of precision and recall.

$$\overline{Prec} = \frac{\sum_i TP_i}{\sum_i TP_i + \sum_i FP_i} \quad (1) \qquad \overline{Rec} = \frac{\sum_i TP_i}{\sum_i TP_i + \sum_i FN_i} \quad (2)$$

To verify the significance of the results, we employed the Friedman and Nemenyi statistical tests, recommended for comparisons involving many classifiers and several datasets [17]. We adopted a confidence level of 95% in the statistical tests. As in [5] and [7], 2/3 of each dataset were used for inducing the classification models and 1/3 for test. We used the same data partitions suggested in [5].

C. Parameters

We investigate the performance of HMC-LMLP using the conventional Back-propagation algorithm (Bp) [18], and the Resilient Back-propagation (Rprop) [19].

The HMC-LMLP parameters were optimized using the Eisen validation dataset. This dataset was selected because it was one of the datasets where Clus-HMC obtained its best performance (0.380), and also because it has a relatively small number of attributes, which allows to run several experiments in a reasonable amount of time. The following parameters were optimized: (i) number of neurons in each hidden layer (beginning with the hidden layer of the MLP network associated with the first hierarchical level, and finishing with the hidden layer of the MLP network associated with the last level), (ii) the learning rate and momentum constant used in the Back-propagation algorithm, and (iii) the range of values used to initialize the neural network's weights. We executed HMC-LMLP over the validation dataset using different sets of parameter values. We employed different initial weight values, number of hidden neurons, learning rates, and momentum constants. We did not use all possible sets of values due to the large number of possibilities.

For the initial weights, we noticed that the larger their values, the more likely the occurrence of overfitting. We varied the initial weights by randomly selecting them initially from [-0.1,0.1], but gradually increasing the range to [-1,1]. We tested a limited number of neurons for each hidden layer, beginning with 1.0/1.0/0.95/0.9/0.85/0.8/0.75/0.7/0.65/0.6/0.55/0.5/0.45 neurons in each layer and gradually decreasing these values. These numbers represent the fraction of the total number of network inputs. Thus, if a neural network has 100 inputs, the value 0.6 means 60 hidden neurons. Considering the learning rate and momentum, we started our experiments with the same default values used in the Weka toolkit [20] (learning rate equal to 0.3 and momentum equal to 0.2). We gradually decreased these values and noticed that the neural networks became less prone to overfitting as these values decreased. The final parameters obtained for HMC-LMLP after the preliminary experiments are listed next.

- Number of hidden neurons per level (fraction of the total number of network inputs): 0.65/0.65/0.6/0.55/0.5/0.45/0.4/0.35/0.3/0.25/0.2/0.15/0.1;
- Learning rate and momentum constant used in Back-propagation for hidden and output layers: $\{0.05, 0.03\}$ and $\{0.03, 0.01\}$, respectively;
- Initial weights of the neural networks: within [-0.1,0.1];
- Parameter values of the Rprop algorithm: initial Delta (Δ_0) = 0.1, maximum Delta (Δ_{max}) = 50.0, minimum Delta (Δ_{min}) = $1e^{-6}$, increase factor (η^+) = 1.2, and decrease factor (η^-) = 0.5.

We would like to point out that we decreased the number of hidden neurons of the neural networks as the hierarchical levels became deeper. Our intention was to avoid overfitting, since the number of training instances is smaller for the networks associated with deeper hierarchical levels. The first two values

(0.65/0.65) are the same because all instances are classified in classes belonging to the first and second levels. The Rprop parameter values were the ones suggested in [19].

D. Results

Table II presents the PR-curves for the HMC-LMLP's versions and the literature methods. We refer to the HMC-LMLP versions as Bp-Comp (Back-propagation with classes augmenting the feature vectors), Bp-NoComp (Back-propagation with no augmentation), Rprop-Comp (Resilient Back-propagation with classes augmenting the feature vectors), and Rprop-NoComp (Resilient Back-propagation with no augmentation).

The results for HMC-LMLP and *hm*Ant-Miner are the mean and standard deviation over 10 executions. Each HMC-LMLP execution was performed with randomly initialized weights. Clus-HMC, Clus-HSC, and Clus-SC are deterministic algorithms, and thus require a single execution. We highlight the best absolute values. We also compared the HMC methods considering specific classes with the goal of examining their behavior when predicting classes in different hierarchical levels. Since Clus-HMC is considered to be the state-of-the-art method, we performed the comparisons in the Seq dataset, in which Clus-HMC showed the best results according to Table II.

We selected, for each level, the three classes where Clus-HMC obtained its best results. We went down until the seventh hierarchical level. Results are shown in Table III. The best absolute values are highlighted.

V. Discussion

According to Table II, the best results were obtained by Rprop-NoComp, Clus-HMC and Clus-HSC. We can see that using the true classes to augment the feature vectors did not improve the classification results when compared to the HMC-LMLP version that does not use the augmentation process. We believe there are two reasons that may have harmed Bp(Rprop)-Comp's performance: (i) adapting the DAG hierarchy, and (ii) using different values to augment the feature vectors during the training and test phases.

Regarding the adaptation made in the DAG hierarchies, Bp-Comp and Rprop-Comp could have achieved better performance if all relationships between classes had been available during training. Recall we had to adapt the DAG hierarchy to define the depth of a class as being the number of edges in the longest path between class and root node. For that reason, many hierarchical relationships between classes were not considered during the training phase. Figure 3 illustrates a scenario that explains this rationale.

Following the scenario presented in Figure 3, consider that a training instance is assigned to paths A.C and A.B.C, and that class C is a direct subclass of both classes A and B. In this scenario, there are two possible depths for class C: 2 (A.C) and 3 (A.B.C). In the adaptation process we have proposed, class C is defined as belonging to the third level. In this case, when training an MLP for the third level, we consider class C as subclass of class B alone. Thus, when training a neural network to predict class C (third level), we are not using the information related to all its superclasses (classes A and B) as inputs. Only class B is considered.

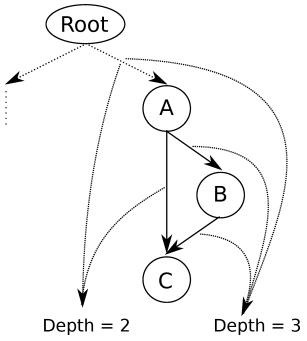


Fig. 3: Illustrative example of different depths for the same class.

The different values that are used when augmenting the feature vectors may also have harmed the performance of Bp-Comp and Rprop-Comp. Recall that when training an MLP for level l, we make use of the true labels of the training instances (values 0 or 1) at level $l-1$ to augment their feature vectors. However, the true labels are not available during test. Thus, the predictions made by the neural network at level $l-1$ (values [0,1]) were used instead. Therefore, each MLP was trained using 0 or 1 values for augmenting the feature vectors during training, but were tested with real values in the interval [0,1].

Comparing the Bp and Rprop algorithms, note that the Rprop versions of HMC-LMLP provided better predictive performance than the Bp versions. These results suggest that Rprop copped better with the greater number of attributes, which increased considerably due to the augmentation process.

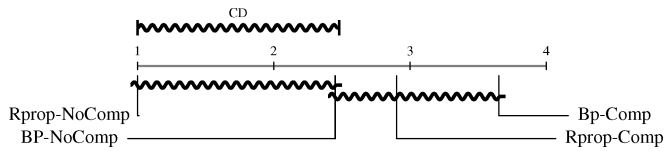


Fig. 4: Critical diagram presenting results of the 4 HMC-LMLP's versions.

Figure 4 shows the results of the statistical tests regarding the four HMC-LMLP's versions. Note that Rprop-NoComp outperforms most versions of HMC-LMLP with statistical significance. The difference for BP-NoComp is within the limit of the critical difference. The tests confirm that Rprop was the best learning algorithm in both scenarios (with and without augmentation).

Figure 5 presents the results of the statistical tests regarding the best version of HMC-LMLP (Rprop-NoComp) and the four baseline methods (Clus-HMC, Clus-HSC, Clus-SC, and *hm*Ant-Miner). Clus-HMC provides the lowest average rank (2.0), whereas Rprop-NoComp provides the second lowest (2.15) followed by Clus-HSC (2.25). Note that the difference in average rank among these three methods is very small, and indeed it is deemed as insignificant by the statistical tests. The performance achieved by *hm*Ant-Miner (average rank of 3.6) and Clus-SC (average rank of 5.0) is considerably lower than the best ranked methods. Even though *hm*Ant-Miner is within the limit of the critical difference, Clus-SC is out of

TABLE II: $AU(\overline{PRC})$ values obtained

Dataset	Bp-Comp	Bp-NoComp	Rprop-Comp	Rprop-NoComp	Clus-HMC	Clus-HSC	Clus-SC	*hm*Ant-Miner
Cellcycle	0.352 ± 0.0025	0.359 ± 0.0007	0.357 ± 0.001	0.365 ± 0.001	0.357	**0.371**	0.252	0.325 ± 0.0079
Church	0.336 ± 0.0015	0.340 ± 0.0011	0.341 ± 0.003	0.347 ± 0.001	0.348	**0.397**	0.289	0.334 ± 0.0010
Derisi	0.336 ± 0.0013	0.345 ± 0.0006	0.336 ± 0.002	0.349 ± 0.001	**0.355**	0.349	0.218	0.321 ± 0.0068
Eisen	0.393 ± 0.0014	0.395 ± 0.0012	0.396 ± 0.003	**0.403 ± 0.001**	0.380	0.365	0.270	0.373 ± 0.0110
Gaschl	0.373 ± 0.0029	0.378 ± 0.0012	0.372 ± 0.004	**0.384 ± 0.001**	0.371	0.351	0.239	0.352 ± 0.0082
Gasch2	0.359 ± 0.0019	0.362 ± 0.0012	0.359 ± 0.003	0.369 ± 0.001	0.365	**0.378**	0.267	0.334 ± 0.0165
Pheno	0.315 ± 0.0025	0.322 ± 0.0011	0.316 ± 0.002	0.325 ± 0.002	0.337	**0.416**	0.316	0.336 ± 0.0017
Spo	0.334 ± 0.0021	0.340 ± 0.0007	0.332 ± 0.003	0.345 ± 0.001	0.352	**0.371**	0.213	0.329 ± 0.0078
Expr	0.369 ± 0.0031	0.371 ± 0.0014	0.373 ± 0.003	**0.384 ± 0.002**	0.368	0.351	0.249	0.343 ± 0.0066
Seq	0.368 ± 0.0034	0.368 ± 0.0018	0.375 ± 0.003	0.384 ± 0.002	**0.386**	0.282	0.197	0.371 ± 0.0069
Average	0.355	0.358	0.356	**0.365**	0.362	0.363	0.251	0.342

TABLE III: Best $AU(\overline{PRC})$ obtained in specific classes for the Seq dataset.

Level	Classes	Bp-Comp	Bp-NoComp	Rprop-Comp	Rprop-NoComp	Clus-HMC	Clus-HSC	Clus-SC	*hm*Ant-Miner
1	GO:0044464	0.963 ± 0.006	**0.966 ± 0.002**	0.965 ± 0.002	0.964 ± 0.006	0.960	0.951	0.951	0.953 ± 0.0054
1	GO:0009987	0.869 ± 0.005	0.868 ± 0.004	0.870 ± 0.004	**0.873 ± 0.006**	0.872	0.844	0.844	0.870 ± 0.0072
1	GO:0008152	0.791 ± 0.010	0.790 ± 0.008	**0.797 ± 0.007**	0.796 ± 0.006	0.774	0.700	0.700	0.730 ± 0.0098
2	GO:0044424	0.934 ± 0.002	**0.937 ± 0.003**	0.936 ± 0.003	0.935 ± 0.003	0.922	0.897	0.894	0.916 ± 0.0051
2	GO:0044237	0.723 ± 0.011	0.734 ± 0.009	0.734 ± 0.007	**0.742 ± 0.009**	0.714	0.694	0.686	0.685 ± 0.0123
2	GO:0044238	0.690 ± 0.012	0.691 ± 0.010	**0.702 ± 0.009**	0.701 ± 0.006	0.664	0.632	0.634	0.651 ± 0.0142
3	GO:0044446	**0.691 ± 0.007**	0.687 ± 0.007	0.677 ± 0.011	0.674 ± 0.005	0.649	0.610	0.564	0.615 ± 0.0162
3	GO:0044444	0.648 ± 0.014	**0.652 ± 0.009**	0.630 ± 0.014	0.636 ± 0.008	0.629	0.546	0.530	0.570 ± 0.0083
3	GO:0043229	0.591 ± 0.008	0.597 ± 0.007	**0.601 ± 0.007**	0.595 ± 0.006	0.584	0.554	0.547	0.561 ± 0.0109
4	GO:0043231	0.558 ± 0.009	0.562 ± 0.008	**0.567 ± 0.008**	0.561 ± 0.004	0.535	0.500	0.485	0.514 ± 0.0081
4	GO:0044428	0.491 ± 0.019	0.495 ± 0.015	0.496 ± 0.013	**0.497 ± 0.019**	0.446	0.353	0.346	0.409 ± 0.0256
4	GO:0044267	0.428 ± 0.015	**0.460 ± 0.010**	0.450 ± 0.011	0.458 ± 0.012	0.383	0.337	0.329	0.376 ± 0.0240
5	GO:0006412	0.684 ± 0.014	**0.684 ± 0.012**	0.664 ± 0.018	0.678 ± 0.012	0.491	0.440	0.501	0.435 ± 0.0279
5	GO:0005634	0.321 ± 0.013	0.323 ± 0.017	0.317 ± 0.009	0.320 ± 0.011	**0.327**	0.241	0.283	0.322 ± 0.0192
5	GO:0005739	0.395 ± 0.016	**0.403 ± 0.007**	0.390 ± 0.012	0.386 ± 0.014	0.308	0.284	0.310	0.244 ± 0.0096
6	GO:0045449	**0.239 ± 0.020**	0.226 ± 0.016	0.191 ± 0.013	0.218 ± 0.023	0.167	0.106	0.127	0.188 ± 0.0267
6	GO:0017111	0.115 ± 0.024	0.122 ± 0.028	0.301 ± 0.033	**0.304 ± 0.030**	0.134	0.190	0.200	0.071 ± 0.0053
6	GO:0043687	0.227 ± 0.027	0.218 ± 0.026	**0.236 ± 0.026**	0.235 ± 0.026	0.105	0.096	0.120	0.115 ± 0.0136
7	GO:0006355	**0.233 ± 0.017**	0.225 ± 0.016	0.183 ± 0.010	0.209 ± 0.021	0.173	0.100	0.117	0.174 ± 0.0217
7	GO:0016568	0.099 ± 0.014	0.092 ± 0.012	0.093 ± 0.006	**0.121 ± 0.011**	0.094	0.051	0.058	0.079 ± 0.0128
7	GO:0000723	0.081 ± 0.008	**0.086 ± 0.011**	0.082 ± 0.012	0.081 ± 0.009	0.085	0.064	0.056	0.082 ± 0.0104
	Average	0.513	0.515	0.518	**0.523**	0.477	0.438	0.442	0.455

the significance margin, which means it is outperformed by the first three methods with statistical significance.

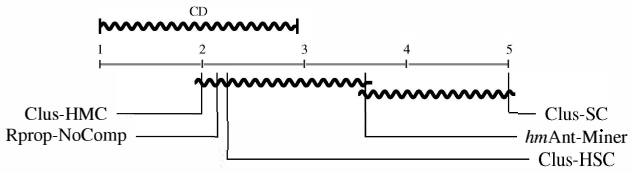


Fig. 5: Critical diagram presenting results of the best HMC-LMLP network and the baseline algorithms.

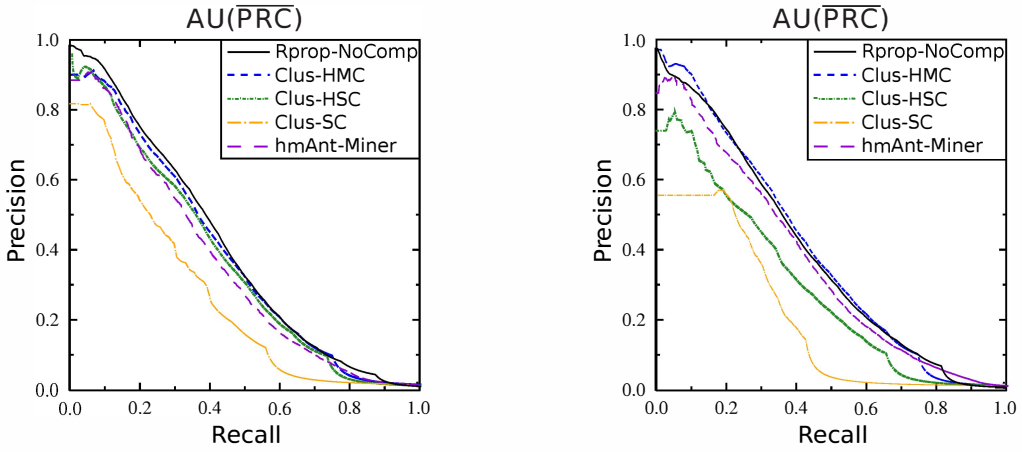
Considering the results in specific classes (Table III), HMC-LMLP provided the best results for most of the classes. These results were unexpected, given that Clus-HMC achieved a better overall performance than Bp-Comp, Bp-NoComp and Rprop-Comp (Table II). Notwithstanding, we verified that from the 2849 classes that belong to the test dataset, Clus-HMC provided the best results for 2192, whereas the HMC-LMLP's versions achieved the best results in a much smaller number of classes. Hence the results of Clus-HMC in Table II.

The PR-curves obtained for datasets Eisen and Seq are depicted in Figure 6. These datasets are the ones where Clus-HMC obtained its best results. We compared the PR-curves of the literature methods with the PR-curve obtained by Rprop-NoComp, since it was the best among the HMC-LMLP's versions. The PR-curves shown for HMC-LMLP and *hm*Ant-Miner are those from the executions in which they obtained the best results in the validation data. We can see that Clus-HMC, Clus-HSC and HMC-LMLP are quite even performance-wise.

VI. Conclusion

In this paper, we presented **H**ierarchical **M**ulti-Label **C**lassification with **L**ocal **M**ulti-**L**ayer **P**erceptron (HMC-LMLP), which associates an MLP to each level of a DAG-structure class hierarchy. HMC-LMLP employs the Local Classifier per Level (LCL) [3] strategy, complementing the feature vectors of the instances with their true classes, in order to make use of local information within each level of the hierarchy. With this, we try to avoid problems such as the loss of label dependency during training.

We tested HMC-LMLP on ten datasets related to protein function prediction, in which the protein functions were organized following the Gene Ontology. We compared its performance against state-of-the-art methods from the literature of HMC, and the empirical analysis indicated that HMC-LMLP matches the predictive performance of the state-of-the-art methods, often presenting better results in specific



(a) Eisen dataset (b) Seq dataset

Fig. 6: PR-curves of Rprop-NoComp, Clus-HMC, CLus-HSC, Clus-SC, and *hm*Ant-Miner.

classes of the DAG hierarchy. According to the results, true labels to complement the feature vectors did not improve the classification performance. We showed this may be due to the adaptation performed in the DAG taxonomies, which resulted in no use of the complete information regarding parent-child class relationship.

As future work, we plan to investigate other neural networks training algorithms. We will also investigate alternatives to adapt the DAG taxonomies to be used with HMC-LMLP, trying to overcome the disadvantages of the currently used adaptation. Other strategies for correct inconsistencies in the prediction will also be tried. Finally, we plan to incorporate other knowledge source in the training process, such as protein-protein interactions networks, and also use HMC-LMLP in other application domains, such as text classification.

ACKNOWLEDGMENT

The authors would like to thank the Brazilian research agencies FAPESP and CNPq.

REFERENCES

[1] E. P. Costa, A. C. Lorena, A. C. P. L. F. Carvalho, and A. A. Freitas, "Top-down hierarchical ensembles of classifiers for predicting g-protein-coupled-receptor functions," in *Brazilian Symposium on Bioinformatics*, ser. LNBI, vol. 5167. Springer-Verlag, 2008, pp. 35–46.

[2] M. Ashburner *et al.*, "Gene ontology: tool for the unification of biology. The Gene Ontology Consortium." *Nature Genetics*, vol. 25, pp. 25–29, 2000.

[3] C. Silla and A. Freitas, "A survey of hierarchical classification across different application domains," *Data Mining and Knowledge Discovery*, vol. 22, pp. 31–72, 2010.

[4] R. Cerri, R. Barros, and A. C. P. L. F. Carvalho, "Hierarchical multi-label classification using local neural networks," *Journal of Computer and System Sciences*, vol. 80, no. 1, pp. 39–56, 2013.

[5] C. Vens, J. Struyf, L. Schietgat, S. Džeroski, and H. Blockeel, "Decision trees for hierarchical multi-label classification," *Machine Learning*, vol. 73, pp. 185–214, 2008.

[6] R. Alves, M. Delgado, and A. Freitas, "Knowledge discovery with artificial immune systems for hierarchical multi-label classification of protein functions," in *International Conference on Fuzzy Systems*, 2010, pp. 2097–2104.

[7] F. Otero, A. Freitas, and C. Johnson, "A hierarchical multi-label classification ant colony algorithm for protein function prediction," *Memetic Computing*, vol. 2, pp. 165–181, 2010.

[8] N. Cesa-Bianchi, M. Re, and G. Valentini, "Synergy of multi-label hierarchical ensembles, data fusion, and cost-sensitive methods for gene functional inference," *Machine Learning*, pp. 1–33, 2011.

[9] N. Cesa-Bianchi and G. Valentini, "Hierarchical cost-sensitive algorithms for genome-wide gene function prediction," *Journal of Machine Learning Research*, vol. 8, pp. 14–29, 2010.

[10] G. Valentini, "True path rule hierarchical ensembles for genome-wide gene function prediction," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 8, no. 3, pp. 832–847, May 2011.

[11] M. Kordmahalleh, A. Homaifar, and B. Dukka, "Hierarchical multi-label gene function prediction using adaptive mutation in crowding niching," in *Bioinformatics and Bioengineering (BIBE), 2013 IEEE 13th International Conference on*, Nov 2013, pp. 1–6.

[12] D. Stojanova, M. Ceci, D. Malerba, and S. Dzeroski, "Using ppi network autocorrelation in hierarchical multi-label classification trees for gene function prediction," *BMC Bioinformatics*, vol. 14, no. 1, p. 285, 2013.

[13] R. Cerri, R. C. Barros, A. A. Freitas, and A. C. de Carvalho, "Evolving relational hierarchical classification rules for predicting gene ontology-based protein functions," in *Proceedings of the 2014 Conference Companion on Genetic and Evolutionary Computation Companion*, ser. GECCO Comp '14. New York, NY, USA: ACM, 2014, pp. 1279–1286.

[14] W. Bi and J. Kwok, "Mandatory leaf node prediction in hierarchical multilabel classification," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 12, pp. 2275–2287, Dec 2014.

[15] R. Baraniuk, V. Cevher, M. Duarte, and C. Hegde, "Model-based compressive sensing," *IEEE Transactions on Information Theory*, vol. 56, no. 4, pp. 1982–2001, April 2010.

[16] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1999.

[17] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

[18] D. E. Rumelhart and J. L. McClelland, *Parallel distributed processing: explorations in the microstructure of cognition*, D. E. Rumelhart and J. L. McClelland, Eds. Cambridge, MA: MIT Press, 1986, vol. 1.

[19] M. Riedmiller and H. Braun, "A Direct adaptive method for faster backpropagation learning: The RPROP algorithm," in *International Conference on Neural Networks*, 1993, pp. 586–591.

[20] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: an update," *SIGKDD Explor. Newsl.*, vol. 11, no. 1, pp. 10–18, 2009.