Journal of Computer and System Sciences 80 (2014) 39–56

Contents lists available at SciVerse ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

Hierarchical multi-label classification using local neural networks

Ricardo Cerri *, Rodrigo C. Barros, André C.P.L.F. de Carvalho

Departamento de Ciências de Computação, Instituto de Ciências Matemáticas e de Computação — ICMC, Universidade de São Paulo — Campus de São Carlos, Caixa Postal 668, 13560-970 São Carlos, SP, Brazil

ARTICLE INFO

Article history:
Received 16 July 2012
Received in revised form 25 November 2012
Accepted 14 March 2013
Available online 22 March 2013

Keywords:
Hierarchical multi-label classification
Neural networks
Local classification method

ABSTRACT

Hierarchical multi-label classification is a complex classification task where the classes involved in the problem are hierarchically structured and each example may simultaneously belong to more than one class in each hierarchical level. In this paper, we extend our previous works, where we investigated a new local-based classification method that incrementally trains a multi-layer perceptron for each level of the classification hierarchy. Predictions made by a neural network in a given level are used as inputs to the neural network responsible for the prediction in the next level. We compare the proposed method with one state-of-the-art decision-tree induction method and two decision-tree induction methods, using several hierarchical multi-label classification datasets. We perform a thorough experimental analysis, showing that our method obtains competitive results to a robust global method regarding both precision and recall evaluation measures.

1. Introduction

Classification is a machine learning (ML) task that aims at building class distribution models taking into account a set of predictive attributes (also known as features). The outcome of such a model is used for assigning class labels to new examples whose only known information are the values of the predictive attributes. The set of examples whose class distribution is known is named the training set — $\{\mathbf{x}^i, c^i\}_{i=1}^N$ — where $\mathbf{x}^i$ is the i-th vector of predictive attributes $\mathbf{x}^i = (x_1^i, x_2^i, x_3^i, \ldots, x_n^i)$ and c^i is the corresponding class label.

A classification algorithm usually operates in two steps. In the first step, the training set $\{\mathbf{x}^i, c^i\}_{i=1}^N$ is used to induce a model able to represent the relationship between each training example, defined by a set of predictive attribute values, and its class label. In the second step, the classification model is used to classify new examples whose class labels are unknown.

Most classification problems addressed in the literature consider each example to be associated with only one class label. In addition, class labels are treated as unrelated, in the sense that classification algorithms do no assume the existence of any kind of relationship between different class labels. This classification scheme is usually referred to as *flat* (or *non-hierarchical*) *single-label* problems. Nevertheless, in many real-world classification problems (*e.g.*, classification of biological data), one or more class labels can be divided in subclasses or grouped in superclasses. In these cases, the classes form a hierarchical structure, usually in the form of either a tree or a Directed Acyclic Graph (DAG). These problems are known in the ML literature as hierarchical classification problems, in which new examples are assigned to classes associated to

* Corresponding author.
E-mail addresses: cerri@icmc.usp.br (R. Cerri), rcbarros@icmc.usp.br (R.C. Barros).

0022-0000/$ – see front matter
http://dx.doi.org/10.1016/j.jcss.2013.03.007

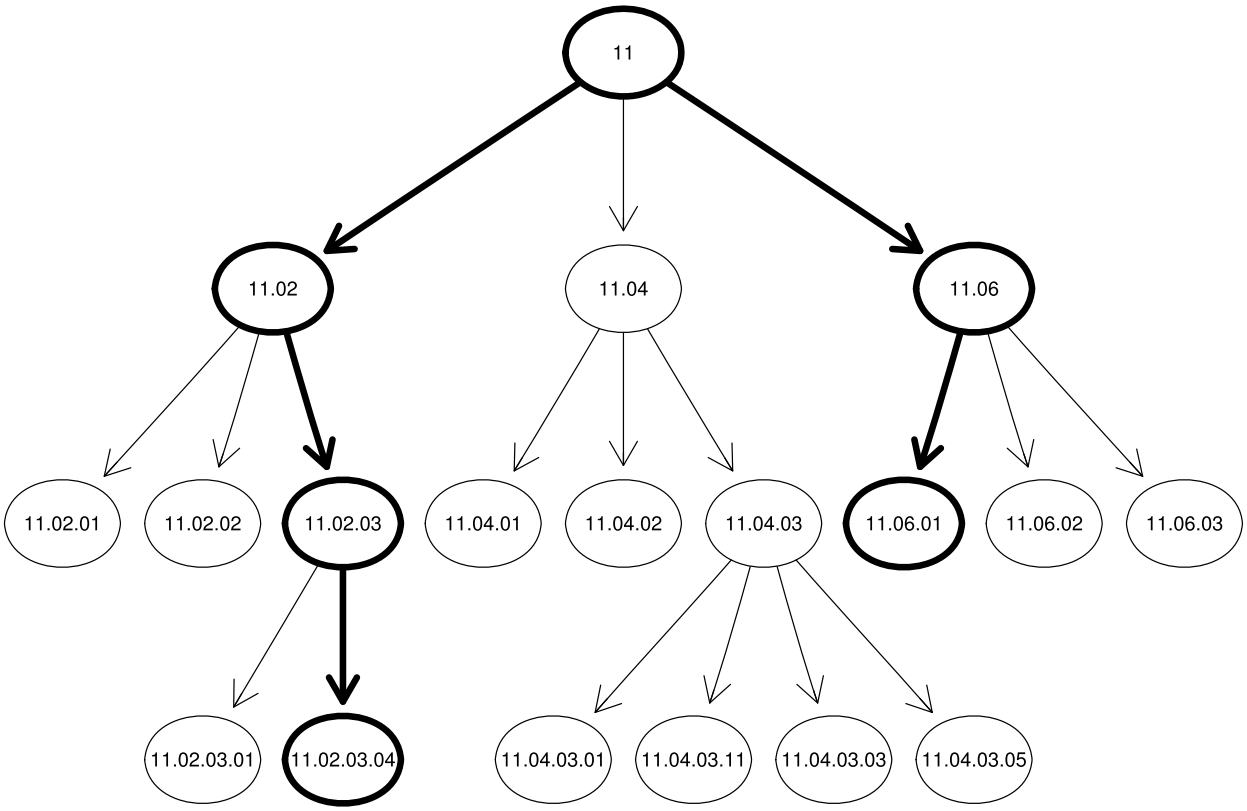


Fig. 1. Example of a tree hierarchical structure. Adapted from [11].

nodes belonging to a hierarchy [1]. For instance, an example that belongs to a given class automatically belongs to all its superclasses (the so-called *hierarchy constraint*).

Hierarchical problems can be further structured in an even more complex manner. For instance, there are cases in which examples can be assigned to different hierarchical paths — the example may belong simultaneously to different classes in a same hierarchical level. These problems are known as Hierarchical Multi-label Classification (HMC) problems, and are very common in protein and gene function prediction tasks [2–8].

Due to its own nature, HMC is more challenging than flat single-label classification. Indeed, it is generally much more difficult to discriminate between classes that lie in the bottom of the hierarchy than classes at the top of the hierarchy, specially considering that the number of examples per class is usually much smaller at the deeper levels of the hierarchy. Respecting the *hierarchy constraint* and also discriminating among a large number of classes is yet another challenge in HMC. Formally, an HMC problem can be posed as follows:

Definition. Given a space of examples **X**, the objective of the training process is to find a function able to map each example in **X** into a set of classes, respecting the constraints of the hierarchical structure and optimizing a quality criterion.

Although the given HMC definition says that an induced model has to classify an example into proper hierarchical paths, there are some works that allow inconsistent predictions. This is the case in the studies made by Obozinski [9], and Valentini [10], where predictions inconsistent with the hierarchy are made, followed by an additional step of making the class assignments consistent with the hierarchy.

An example of an HMC problem structured as a tree is illustrated in Fig. 1, in which an example is assigned to two paths of the hierarchy, formed by the classes 11.02.03.04 and 11.06.01.

Two main approaches have been used to solve HMC problems, known as local and global approaches. In the local approach, conventional classification algorithms, such as decision-trees [12] and support vector machines [13], are trained to produce an hierarchy of classifiers, which are in turn executed following a top-down strategy to classify unlabelled examples [14]. In this approach, local information about the class hierarchy is used during the training process of each base classifier. According to Silla and Freitas [15], the local information can be used in different ways, depending on how the local classifiers are induced. Roughly, three main strategies for using local information have been identified: one Local Classifier per Node (LCN), one Local Classifier per Parent Node (LCPN), and one Local Classifier per Level (LCL). The LCN strategy trains one binary classifier for each class of the hierarchy [16]. The LCPN strategy trains, for each internal class,

a multi-class classifier to distinguish between its subclasses [17], and the LCL strategy trains one multi-class classifier for each hierarchical level, where each classifier is responsible for the prediction in its associated level [11].

Different from the local approach, the global approach induces only one classifier using all classes of the hierarchy at once. After training the classifier, the classification of a new example occurs in just one step [5]. Since global-based methods induce just one classifier to consider the specificities of the classification problem, they do not use conventional classification algorithms, unless these are adapted to consider the hierarchy of classes.

In our previous works [18,11], we proposed a novel method, named Hierarchical Multi-label Classification with Local Multi-Layer Perceptron (HMC-LMLP). It is a local HMC method where an MLP network is associated with each hierarchical level and responsible for the predictions in that level. The predictions for a level are later used as inputs for the MLP network associated with the next level. We investigated two alternatives for the training of the MLP networks: the Back-propagation algorithm [19] and the Resilient back-propagation algorithm (Rprop) [20]. Rprop performs a local adaptation of the weight-updates according to the behaviour of the error function. In Rprop, a step-size — the update amount of a weight — is adapted for each weight individually, and, different from other techniques, its step-sizes are independent of the absolute value of the partial derivative. The benefits of this update scheme are fully described in [20].

In this paper, we use different parameter values regarding the previous studies, which, as a result, improved HMC-LMLP predictive performance. We also perform a more detailed analysis of the experimental results obtained, using two variations of precision–recall curves, analyzing the overall performance of the methods considering all classes and also the performance of the methods regarding specific classes. In addition, we add two new datasets to the experiments. The remaining of the paper is organized as follows. In Section 2, we review other studies related to our approach. Our local method for HMC employing artificial neural networks and its motivation are described in Section 3. In Section 4 we explain the methodology followed in the experimental analysis. The experimental results are presented in Section 5, in which our method is compared with three decision-tree induction methods for HMC problems, using for such protein function prediction datasets. Finally, we summarize our conclusions and point out to future research directions in Section 6.

2. Related work

Several studies have investigated new alternatives to solve HMC problems. Some of these studies propose new methods capable of dealing with hierarchies either structured as trees or as DAGs, whereas others propose more general methods able to handle both structures.

Regardless of the nature of the hierarchical problem, the methods proposed in the literature can be categorized as following either a local (top-down) or a global (one-shot, big-bang) approaches. In the local approach, conventional classification algorithms are trained to induce a tree of classifiers, which are in turn used in a top-down fashion for the classification of new examples. Initially, the most generic class — located at the first hierarchical level — is predicted. Afterwards, it is used to reduce the set of possible classes for the next level. In the global approach, as the name indicates, the objective is to learn a single global model for all classes in the hierarchy. This section describes some of the recent local and global methods investigated in the literature.

2.1. Local methods

A Bayesian classifier, named HBayes, was proposed by Cesa-Bianchi et al. [21] and applied to the hierarchical classification of documents. The method trains a classifier for each hierarchical node and calculates class probabilities $\hat{p}_j(\mathbf{x}^i)$ for all examples. For the classification process, the method is initialized with all hierarchical classes in a set. These classes are then removed one by one, and every time an example is assigned to a given class c^j, it is also assigned to all classes belonging to the path between this class and the root node. A node can be removed from the set if it is a leaf node, or if all nodes of its subtree have been removed. The work of Cesa-Bianchi and Valentini [22] extended the method described by Cesa-Bianchi et al. [21], resulting in a variant named HBayes-CS (HBayes Cost Sensitive). HBayes-CS, according to the authors, is more suitable for dealing with skewed datasets. In HBayes-CS, a parameter was introduced to balance the false positive (FP) and false negative (FN) mistake costs.

An ensemble of local classifiers, named TPR (True Path Rule Ensemble), was proposed and applied to gene function prediction by Valentini [16]. In this method, each trained classifier estimates the local probability $\hat{p}_j(\mathbf{x}^i)$ that a given example $\mathbf{x}^i$ belongs to a given class c^j. A combination phase estimates the global consensual probability $p_j(\mathbf{x}^i)$. In [23] and [10], the authors modified this method in order to modulate the relation between the prediction of a class and the prediction of its descendants. The new method, named TPR-W, employs a weight w_p within the parent classes, $0 \leqslant w_p \leqslant 1$, such that if $w_p = 1$, the classification decision in the corresponding node depends only on the local classifier in the node. For $w_p \neq 1$, the prediction decision in the node is proportionally divided between the local classifier in the node and the set of all its child classifiers. Experiments using biological datasets showed the superiority of TPR-W over TPR.

Cesa-Bianchi et al. [24] investigated the synergy between different strategies related to the gene function prediction task. They integrated kernel-based data fusion tools and ensemble algorithms [25] with cost sensitive HMC methods [22,10]. The authors defined synergy as the improvement, considering any evaluation measure, in the classification performance due to the use of concurrent learning strategies. The synergy is detected when the combined action of two strategies achieves better classification performance than the average of the performances of the two strategies used separately [24].

Mayne and Perri [26] proposed a local-based method and employed it in the document classification task. A hierarchy of Naive Bayes classifiers was used, where each classifier outputs the larger class probability estimate. This probability is then transmitted to the classifiers associated to the child classes of the current class, and added to the attribute vector of the example being classified. The class probabilities are concatenated and added to the attribute vectors of the examples in each hierarchical level.

In a study from Cerri and Carvalho [27], two new methods following the local approach were proposed, based on multi-label non-hierarchical classification methods, and applied to gene function prediction problems. The first method, named HMC-LP, is based on the Label-Powerset method [28]. The second method, named HMC-CT, is based on the Cross-Training method [29]. In each hierarchical level, HMC-LP combines the classes assigned to an example, generating a new and unique class, and transforming the original HMC problem into a hierarchical single-label problem. The HMC-CT method uses a class decomposition strategy, transforming the original HMC problem into a set of hierarchical single-label problems. The SVM algorithm was used as base classifier. A study from Cerri et al. [30] extended Cerri and Carvalho [27] using different algorithms as base classifiers.

2.2. Global methods

One of the first global methods was proposed by Clare [2]. This method, named HMC4.5, is based on decision-tree induction algorithms, and was applied to gene function prediction. It is a variation of the C4.5 algorithm [12], with modifications in the calculation of class entropy. In the original C4.5 algorithm, the entropy is used to decide the best data split in the decision-tree, *i.e.*, the best attribute to be placed in a tree node. The proposed modification uses the sum of the entropies of all classes, and also the information related to the size of the tree rooted by a given class.

Vens et al. [5] compared three methods based on the concept of PCT (Predictive Clustering Trees). The authors used the Clus-HMC method [31], based on the global approach, which trains only one decision-tree to cope with the entire classification problem, and compared its performance with two other local methods. The first method, named Clus-SC, trains an independent decision-tree for each class of the hierarchy, ignoring the relationships between classes. The second method, named Clus-HSC, explores the hierarchical relationships between the classes to induce a decision-tree for each class. The authors applied the methods to hierarchies structured as trees and DAGs, and discussed about the modifications needed so the algorithms could cope with both types of hierarchical structures. The experiments performed using biological datasets showed that the global method was superior both in the predictive performance and size of the induced decision-tree. Still based on PCT, a study from Schietgat [32] used an ensemble technique to combine the decision-trees induced by the Clus-HMC method. The proposed method, named Clus-HMC-Ens, uses the Bagging technique [33] to train different classifiers using replications (bootstrap) of the training dataset. Experiments on biological datasets showed that the ensemble method obtained better results.

Alves et al. [34] proposed a global method using Artificial Immune Systems (AIS) [35] for the generation of HMC rules. The method, named Multi-label Hierarchical Classification with an Artificial Immune System (MHC-AIS), was employed in the protein function prediction task. The training algorithm is divided into two basic procedures: Sequential Covering (SC) and Rule Evolution (RE). These procedures produce candidate classification rules comprised of two parts: an antecedent (IF part), represented by a vector of attribute-value conditions, and a consequent (THEN part), represented by a set of predicted classes. The SC procedure iteratively calls the RE procedure until all (or almost all) training examples (antigens) are covered by the discovered rules. The RE procedure evolves classification rules (antibodies) that are used to classify the examples. The best antibody is added to the set of discovered rules. Alves et al. [6] added some procedures to the algorithm previously proposed by them [34] in order to improve its performance. The first procedure consists in recalculating the fitness of all rules considering all training examples, and removing from the rules all classes that present a fitness value lower than a given threshold. Another modification was the use of pruning techniques and local search in the best rule discovered by the RE procedure, in order to obtain higher simplicity and precision.

A work from Sangsuriyun [36] proposed a global method based on rule sets, named Hierarchical Multi-label Associative Classification (HMAC), and applied it in the classification of protein and Gene Ontology data. The method uses the so-called negative rules, which consider important the absence of a given attribute to the classification of an example. The method also takes into account the rules that predict a negative set of classes, *i.e.*, rules that indicate that an example does not belong to a given set of classes. The method is divided into three phases. The first one generates rules that are then pruned in the second phase. The pruning process is performed for simplification and elimination of redundancies in the generated rules. A third phase then builds a final set of rules, comprised of the best rules in the initial set.

Otero et al. [7] extend an Ant Colony Optimization (ACO) [37,38] method for hierarchical single-label classification (*h*Ant-Miner [39]) to allow multi-label classification. The *h*Ant-Miner discovers hierarchical classification rules in the format IF antecedent THEN consequent, and was applied to the task of protein and gene function prediction. The construction of a rule is divided into two ant colonies which work in a cooperative manner, one for the rule antecedents and the other for the rule consequents. Basically, a sequential example covering procedure is applied to create classification rules that cover all (or almost all) training examples. The method is initialized with an empty set of rules, and a new rule is added to the set while the number of examples not covered by any rule is higher than a given threshold. At each iteration, a rule is built by the pairing of an antecedent ant with a consequent ant. In addition to allowing multi-label predictions, the new proposed method, called *hm*Ant-Miner, differs from the original method in the following aspects: (i) the consequent

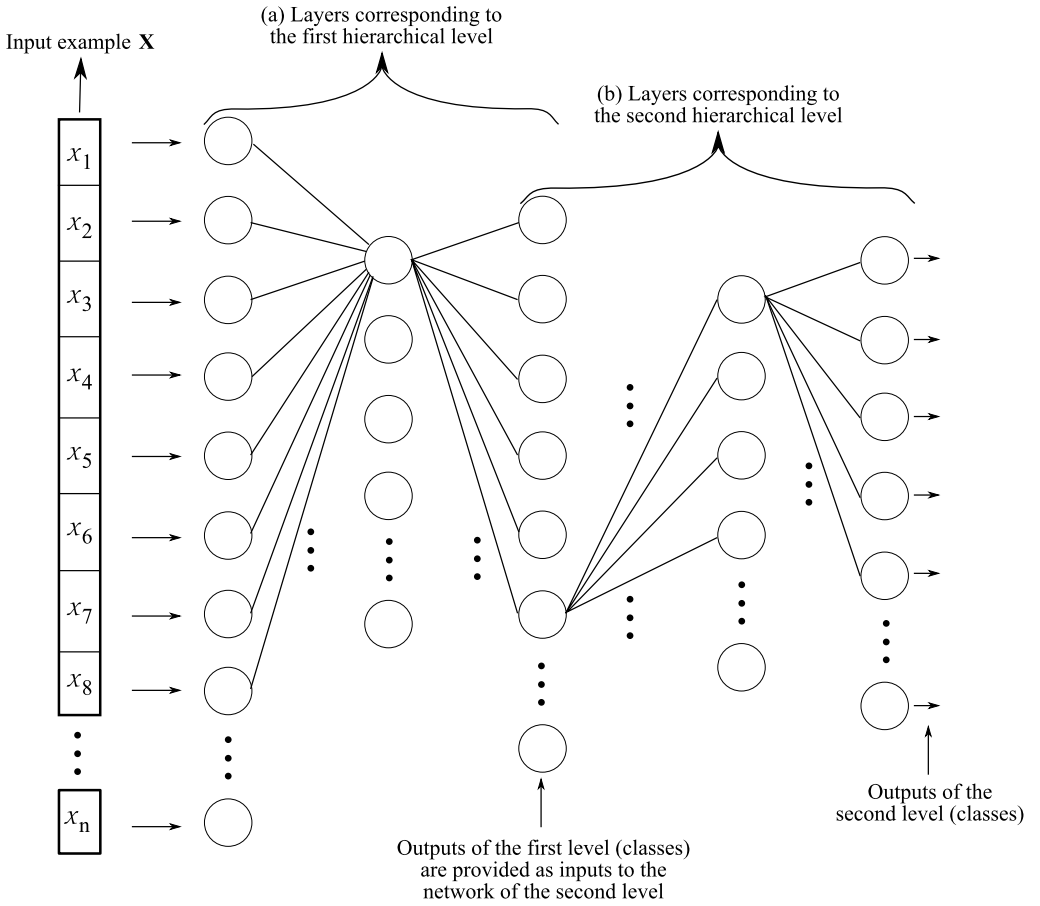


Fig. 2. Architecture of the HMC-LMLP for a two-level hierarchy. Adapted from [11].

of a rule is obtained using a deterministic procedure based on the examples covered by the rule, and not using a specific colony; (ii) a heuristic is used based on the Euclidean distance, where each example is represented by a vector of values representing the class probabilities; (iii) the quality of a rule is evaluated using a distance-based measure; and (iv) the pruning procedure for a rule is not applied in its consequent, but the consequent is recalculated when its antecedent is modified during the pruning.

3. Hierarchical multi-label classification with local multi-layer perceptron

Hierarchical Multi-label Classification with Local Multi-Layer Perceptron (HMC-LMLP), is a local-based HMC method that associates one Multi-Layer Perceptron (MLP) to each classification hierarchical level. It is a method designed to be used in tree structured hierarchies. The method trains the MLPs incrementally, level by level, and after the training process of one neural network for a specific level, the predictions of this network for the training dataset are used as inputs for the training of the next neural network associated with the next hierarchical level. This process continuous until the last level of the hierarchy is reached. Fig. 2 shows an example of the HMC-LMLP architecture for a two-level hierarchy.

The motivation for the proposed architecture is our belief that a neural network can be naturally used as a multi-label classifier, since the use of many output neurons facilitates the multi-label prediction by associating each output neuron to one of the possible classes. Moreover, the use of a neural network per level makes the final architecture similar to a deep neural network with many layers. Thus, a neural network at level l will make use of information that was learned by the neural network in level $l-1$. By using the Local Classifier per Level (LCL) strategy, we try to avoid deficiencies from both local and global methods when dealing with local information. Different from the global approach, here we are able to use local information in the first level (level with more information), and we try to make use of the learned information to classify examples in different hierarchical levels. In addition, in contrast to the Local Classifier per Node (LCN) and Local Classifier per Parent Node (LCPN) strategies, we do not decompose the original classification problem into a large number of sub-problems. The use of many classifiers per level can result in the use of too-specific local information and loss of important global information during the training process [15].

3.1. Training process

The detailed training process of HMC-LMLP is performed as follows. Initially, an MLP is associated with the first hierarchical level (hierarchical level closer to the root node). This MLP can be trained by any training algorithm. The network architecture has one hidden layer and one output layer (classes of the first level). In this study, we employed the Back-propagation [19] and Resilient back-propagation [20] algorithms to train the MLP networks because they are very popular, easy to implement and fast. But of course, other training algorithms can be used. Each MLP network is able to predict more than one class simultaneously, because HMC-LMLP associates each class of the hierarchical level with an output neuron of the network. These outputs are then compared to the desired outputs of the examples in that level in order to compute the network classification errors.

When the training of the MLP network associated with the first hierarchical level is finished (Fig. 2(a) layers corresponding to the first hierarchical level), then a second neural network is associated with the next level of the hierarchy. This MLP network has also one hidden layer and its training follows the same training process used in the previous neural network. The only difference is that the inputs of this network are now the outputs provided by the previous trained MLP (Fig. 2(b) layers corresponding to the second hierarchical level). This process of incrementally training an MLP for each hierarchical level is done until the last level of the hierarchy is reached.

When training an MLP network for a specific hierarchical level, the MLP networks associated with the previous levels are not re-trained, considering that their training already took place in the previous steps of the method. When training an MLP network for the level l, the previous $l-1$ neural networks are used only to provide the inputs to train the MLP associated with the level l. Hence, when training an MLP network for the level l, its inputs are obtained by feeding the MLP network associated with the first level with the training examples. The outputs of the first MLP network are then used as inputs to the second MLP network (associated with the second level), which then provides its outputs to be used as inputs for the next MLP network, and so on, until the MLP associated with the level l is reached.

3.2. Final predictions

To obtain the final predictions (predictions in a test dataset), the test examples are fed to the first MLP network (first level), and then the output of the first level is used as input to the neural network associated with the second level, which in turn gives its output to be used as input to the next neural network. This procedure is repeated until the last level is reached. When this procedure is finalized, thresholds are applied to the output layers of each one of the MLP networks to output predictions for all levels. If the output of a given neuron j is equal or larger than a given threshold, the example being classified is assigned to the 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 equal or larger than a given threshold, the position $\mathbf{v}_j$ is assigned the value 1. Otherwise, it receives the value 0. It is expected that different threshold values result in different classes being predicted. As the activation function used in the neurons is the logistic sigmoid function, the outputs of the neurons range from 0 to 1, and thus we can make use of threshold values also ranging from 0 to 1. The larger the threshold value used, the lower the number of predicted classes. Conversely, the lower the threshold value used, the larger the number of predicted classes.

3.3. Correcting inconsistencies

After the final predictions for new examples by HMC-LMLP, a post-processing phase is used to correct inconsistencies which may have occurred during the classification, *i.e.*, when a subclass is predicted but its superclass is not. These inconsistencies may occur because each MLP network makes its predictions individually, which means that a prediction in a level does not depend (to a certain extent) on the predictions for the other levels. The post-processing phase guarantees that only consistent predictions are made by removing those predicted classes that do not have predicted superclasses. Fig. 3 gives an example of a vector of predicted classes provided by HMC-LMLP before (Fig. 3(a)) and after (Fig. 3(b)) the application of a threshold value of 0.5. Fig. 3(c) shows the final classification after the application of the post-processing step to correct inconsistencies (dotted circles highlight the changed bits). In the example of Fig. 3, the final classification assigns two paths of the hierarchy to the example: 1.1.1.1 and 2.1.1, being "." the level separator.

3.4. Computational analysis

Considering the computational cost, each MLP used in HMC-LMLP 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 of the examples, H_l the number of hidden neurons of the MLP associated with level l, and O_l the number of output neurons of the MLP associated with level l. We can then 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 $(O_{l-1}+1)\times H_l + (H_l+1)\times O_l$. The overall training cost of each MLP associated with level l is then $\mathcal{O}(W_l \times m \times n)$, with m the number of training examples and n the number of training epochs.

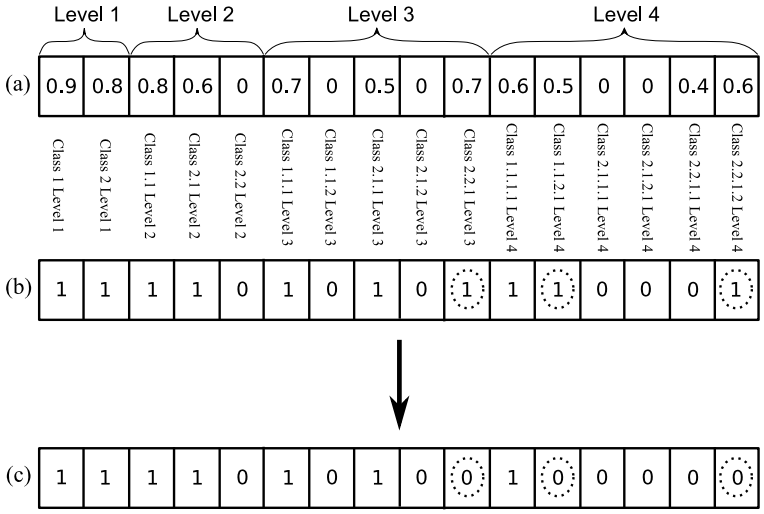


Fig. 3. Example of the class-predicted vector provided by HMC-LMLP. (a) outputs of the neurons; (b) prediction after applying a threshold value of 0.5; (c) final classification after correcting inconsistencies.

4. Materials and methods

This section presents the experimental methodology and datasets used in the experimental analysis. These datasets, the HMC-LMLP source code, and additional scripts to construct graphs and calculate the evaluation measures are available at http://sites.google.com/site/cerrirc/downloads.

4.1. Datasets

Twelve freely-available[1] datasets associated with the task of protein function prediction are used in the experiments. The datasets are related to issues like phenotype data and gene expression levels. They are organized in a tree structure according to the FunCat scheme of classification [40]. Tables 1 and 2 present the main characteristics of the training, validation and testing datasets employed in the experiments.

The datasets Hom and Struc have a very large number of attributes (19 628 attributes for Struc and 47 034 attributes for Hom). Hence, an attribute selection procedure is required in order to allow an adequate use of these datasets by neural networks. For such, we applied an attribute selection technique on these two datasets, selecting 500 attributes for each one. It is not of our knowledge the existence of attribute selection techniques for hierarchical classification. With that in mind, we considered just the classes of the first hierarchical level to apply an existing technique for multi-label classification. We employed the attribute selection technique available in Mulan [28], a Java library for multi-label learning. First, we removed from the datasets the classes from the second level onwards. The attribute selection technique then uses the Label-Powerset [28] transformation method in the classes of the first hierarchical level, and then evaluates the worth of an attribute by measuring the gain ratio with respect to each class [41]. After selecting the attributes, we restore the original multi-label classes to the dataset, reinserting the classes from the second level onwards. The statistics for the datasets Hom and Struc shown in Tables 1 and 2 were calculated after the attribute selection process. Since we just changed their number of attributes, all the hierarchical and multi-label characteristics remain the same.

These datasets are very skewed, having very few positive examples for each class. A description of each dataset can be found in Vens et al. [5]. All numeric attributes of the datasets were standardized (a mean of zero and a variance of 1) and the nominal attributes were transformed into numeric using the one-attribute-per-value approach, where an attribute with k values is transformed into k binary attributes. Instead of 0 and 1, the nominal attributes were assigned -1 (absence) and 1 (presence) values, which are more suitable for the training of neural networks.

4.2. Evaluation measures

As presented in Section 3, the outputs from HMC-LMLP, for each class, are real numbers between 0 and 1. The same is true for the decision-tree methods used in the comparisons. Thus, in order to obtain the final predictions for the methods, a threshold value was employed. When classifying an example, if the corresponding output value for a given class is equal or larger than the threshold, the class is assigned to the example. Otherwise, it is not assigned to the example.

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

Table 1
Summary of datasets: number of attributes ($|A|$), number of classes ($|C|$), number of classes per level (Classes per level), total number of examples (Total) and number of multi-label examples (Multi).

Dataset	$\|A\|$	$\|C\|$	Classes per level	Training		Valid		Test	
				Total	Multi	Total	Multi	Total	Multi
Cellcycle	77	499	18/80/178/142/77/4	1628	1323	848	673	1281	1059
Church	27	499	18/80/178/142/77/4	1630	1322	844	670	1281	1057
Derisi	63	499	18/80/178/142/77/4	1608	1309	842	671	1275	1055
Eisen	79	461	18/76/165/131/67/4	1058	900	529	441	837	719
Expr	551	499	18/80/178/142/77/4	1639	1328	849	674	1291	1064
Gasch1	173	499	18/80/178/142/77/4	1634	1325	846	672	1284	1059
Gasch2	52	499	18/80/178/142/77/4	1639	1328	849	674	1291	1064
Pheno	69	455	18/74/165/129/65/4	656	537	353	283	582	480
Seq	478	499	18/80/178/142/77/4	1701	1344	879	679	1339	1079
Spo	80	499	18/80/178/142/77/4	1600	1301	837	666	1266	1047
Hom	500	499	18/80/178/142/77/4	1669	1323	870	672	1315	1061
Struc	500	499	18/80/178/142/77/4	1665	1340	860	677	1313	1074

Table 2
Summary of datasets: number of examples per level, average number of examples per class for each level, and average number of classes per example for each level.

Dataset		Number of examples per level	Average number of examples per class for each level	Average number of classes per example for each level
Cellcycle	Train	1628/1610/1472/975/303/11	90.44/20.12/8.27/6.87/3.93/2.75	2.40/2.87/2.41/1.77/1.20/1.0
	Valid	848/836/756/492/164/10	47.11/10.45/4.25/3.46/2.13/2.5	2.39/2.84/2.40/1.76/1.18/1.0
	Test	1281/1272/1163/766/245/8	71.17/15.90/6.53/5.39/3.18/2.0	2.47/2.94/2.46/1.76/1.19/1.0
Church	Train	1630/1612/1474/976/302/11	90.55/20.15/8.28/6.87/3.92/2.75	2.40/2.87/2.41/1.77/1.20/1.0
	Valid	844/832/752/490/164/10	46.89/10.40/4.22/3.45/2.13/2.5	2.39/2.84/2.40/1.76/1.80/1.0
	Test	1281/1272/1164/764/243/8	71.17/15.90/6.54/5.38/3.15/2.0	2.46/2.94/2.45/1.76/1.19/1.0
Derisi	Train	1608/1590/1456/969/300/11	89.33/19.87/8.18/6.82/3.90/2.75	2.41/2.88/2.43/1.77/1.20/1.0
	Valid	842/831/751/489/164/10	46.78/10.39/4.22/3.44/2.13/2.5	2.40/2.84/2.40/1.75/1.18/1.0
	Test	1275/1266/1153/761/243/8	70.83/15.82/6.48/5.36/3.15/2.0	2.48/2.95/2.47/1.76/1.19/1.0
Eisen	Train	1058/1054/997/667/210/8	58.78/13.87/6.04/5.09/3.13/2.0	2.48/2.99/2.50/1.80/1.23/1.0
	Valid	529/525/493/323/104/7	29.39/6.91/2.99/2.46/1.55/1.75	2.48/2.96/2.45/1.81/1.21/1.0
	Test	837/834/784/517/161/4	46.50/10.97/4.75/3.95/2.40/1.0	2.54/3.04/2.52/1.80/1.21/1.0
Expr	Train	1639/1621/1481/979/303/11	91.06/20.26/8.32/6.89/3.93/2.75	2.40/2.87/2.41/1.77/1.20/1.0
	Valid	849/837/757/493/164/10	47.17/10.46/4.25/3.47/2.13/2.5	2.39/2.84/2.40/1.76/1.18/1.0
	Test	1291/1282/1173/767/245/8	71.72/16.02/6.59/5.40/3.18/2.0	2.46/2.93/2.45/1.76/1.19/1.0
Gasch1	Train	1634/1616/1477/977/303/11	90.78/20.20/8.30/6.88/3.93/2.75	2.40/2.87/2.41/1.77/1.20/1.0
	Valid	846/834/754/491/164/10	47.00/10.42/4.24/3.46/2.13/2.5	2.39/2.84/2.40/1.76/1.18/1.0
	Test	1284/1275/1167/764/243/8	71.33/15.94/6.56/5.38/3.16/2.0	2.46/2.94/2.45/1.76/1.19/1.0
Gasch2	Train	1639/1621/1481/979/303/11	91.05/20.26/8.32/6.89/3.93/2.75	2.40/2.87/2.41/1.77/1.20/1.0
	Valid	849/837/757/493/164/10	47.17/10.46/4.25/3.47/2.13/2.5	2.39/2.84/2.40/1.76/1.18/1.0
	Test	1291/1282/1173/767/245/8	71.72/16.02/6.59/5.40/3.18/2.0	2.46/2.93/2.45/1.76/1.19/1.0
Pheno	Train	656/649/593/403/133/6	36.44/8.77/3.59/3.12/2.05/1.5	2.54/3.05/2.54/1.76/1.16/1.0
	Valid	353/349/316/204/69/6	19.61/4.72/1.91/1.58/1.06/1.5	2.50/2.90/2.38/1.77/1.17/1.0
	Test	582/578/524/348/108/3	32.33/7.81/3.18/2.70/1.66/0.75	2.59/3.05/2.50/1.78/1.19/1.0
Seq	Train	1701/1639/1499/990/305/11	94.50/20.49/8.42/6.97/3.96/2.75	2.37/2.89/2.42/1.77/1.20/1.0
	Valid	879/842/762/497/164/10	48.83/10.52/4.28/3.50/2.13/2.5	2.35/2.84/2.40/1.75/1.18/1.0
	Test	1339/1298/1188/774/248/8	74.39/16.22/6.67/5.45/3.22/2.0	2.42/2.93/2.45/1.77/1.18/1.0
Spo	Train	1600/1582/1448/963/299/11	88.89/19.77/8.13/6.78/3.88/2.75	2.41/2.88/2.42/1.77/1.20/1.0
	Valid	837/826/747/486/163/10	46.50/10.32/4.20/3.42/2.12/2.5	2.40/2.84/2.40/1.76/1.18/1.0
	Test	1266/1257/1146/758/243/8	70.33/15.71/6.44/5.34/3.16/2.0	2.47/2.95/2.47/1.77/1.19/1.0
Hom	Train	1669/1607/1470/979/302/11	92.72/20.09/8.26/6.89/3.92/2.75	2.38/2.89/2.43/1.77/1.20/1.0
	Valid	870/833/753/492/164/10	48.33/10.41/4.23/3.46/2.13/2.5	2.36/2.84/2.40/1.76/1.18/1.0
	Test	1315/1275/1167/766/245/8	73.06/15.94/6.56/5.39/3.19/2.0	2.43/2.93/2.46/1.77/1.19/1.0
Struc	Train	1665/1634/1495/988/304/11	92.50/20.42/8.40/6.96/3.95/2.75	2.40/2.89/2.42/1.77/1.20/1.0
	Valid	860/840/760/495/162/10	47.78/10.50/4.27/3.49/2.10/2.5	2.37/2.82/2.39/1.75/1.18/1.0
	Test	1313/1292/1182/770/245/8	72.94/16.15/6.64/5.42/3.19/2.0	2.44/2.92/2.44/1.77/1.18/1.0

The choice of the "optimal" threshold value is a difficult task, since low threshold values lead to many classes being assigned to the examples, resulting in high recall and low precision. Conversely, larger threshold values lead to very few examples being classified, resulting in high precision and low recall. To deal with this problem, we used Precision–Recall curves (PR-curves) as the evaluation measure for the methods. To obtain a PR-curve for a given classification method, different thresholds ranging within [0..1] are applied to the outputs of the methods, and thus different values of precision and recall are obtained for each threshold. Each threshold represents a point within the PR-space. The union of these points form a PR-curve, and then the area below the curve (*AUPRC*) is calculated. Different methods can be compared based on their areas below the PR-curves.

In order to calculate the area below the PR-curve, the PR-points must me interpolated [42]. This interpolation guarantees that the area below the curve is not artificially increased, which would happen if the curves were constructed just connecting the points without interpolation.

In this work, we used two PR-curve variations to compare the investigated methods: the area under the average PR-curve ($AU(\overline{PRC})$) and the weighted average of the areas under the individual (per class) PR-curves ($\overline{AUPRC}_w$).

In order to verify the significance of the results, we employed the Friedman and Nemenyi tests, recommended for comparisons involving many datasets and several classifiers [43]. We used a confidence level of 95% in the statistical tests.

4.2.1. Area under the average PR-curve

Given a threshold value, a precision–recall point ($\overline{Prec}, \overline{Rec}$) in the PR-space can be obtained using Eqs. (1) and (2):

$$\overline{Prec} = \frac{\sum_i TP_i}{\sum_i TP_i + \sum_i FP_i} \tag{1}$$

$$\overline{Rec} = \frac{\sum_i TP_i}{\sum_i TP_i + \sum_i FN_i} \tag{2}$$

In Eqs. (1) and (2), i ranges over all available classes, corresponding to the micro-average of precision and recall.

4.2.2. Weighted average of the areas under the individual PR-curves

In order to calculate the weighted average of the areas under the individual PR-curves, we first calculate the $AUPRC_i$ for each class separately, with i ranging from 1 to $|C|$. Afterwards, we obtain the $\overline{AUPRC}_w$ using Eq. (3):

$$\overline{AUPRC}_w = \sum_i w_i \cdot AUPRC_i \tag{3}$$

In Eq. (3), we use w_i to weight the contribution of a class according to its frequency, *i.e.*, $w_i = v_i / \sum_j v_j$, with v_i the c_i's frequency in the dataset [5].

4.3. Parameters

The parameters used in the HMC-LMLP were obtained after preliminary non-exhaustive experiments using the validation datasets, and we leave parameter optimization as a subject for future research.

The parameters employed are the following: (i) number of neurons in each hidden layer (beginning with the hidden layer of the MLP network associated with the first level, and ending 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, (iii) the range of values used to initialize the neural network weights, and (iv) the parameter values of the Resilient back-propagation algorithm. To reduce the influence of parameter selection in the MLP performance, the number of hidden neurons was chosen to represent a fraction of the number of inputs. For instance, 0.5 means that the number of hidden neurons in this MLP network is equal to 50% the number of inputs for the corresponding level. For the Resilient back-propagation algorithm, the parameter values employed were suggested in [20], and no attempt was made to tune these values.

By *preliminary non-exhaustive experiments using the validation datasets* we mean that we run our method for the validation datasets using different sets of parameter values. We used different initial weight values, number of hidden neurons, learning rate values and momentum constant values. We did not use all possible sets of values because the space of possibilities is very large.

For the initial weights, we noticed that the higher their initial values, the more susceptible to over-fitting were the neural networks, having a better predictive performance on more frequent classes but a worse overall predictive performance. We varied the initial weights following the range from $[-0.1, 0.1]$ to $[-1, 1]$, gradually increasing their values. For the number of neurons, we tested a limited number of neurons for each hidden layer, beginning with 1.0/0.9/0.8/0.7/0.6/0.5 neurons in each layer and gradually decreased these values until 0.1/0.08/0.06/0.04/0.03/0.02. These hidden neuron numbers represent the fraction of the total number of network inputs. Thus, if a neural network has 100 inputs, the value 0.6 means that it has actually 60 hidden neurons. Considering the learning rate and momentum constant values, we begun our experiments with the same values used in Weka [44], which are 0.3 for the learning rate and 0.2 for the momentum constant. We gradually decreased theses values and noticed that as the values decreased, the less susceptible to over-fitting the networks became. The final parameters obtained for the two variations of HMC-LMLP after the preliminary experiments are listed next.

Table 3
$AU(\overline{PRC})$ comparison between the HMC-LMLP variations.

Dataset	Bp-Old	Rprop-Old	Bp	Rprop
Cellcycle	0.142 ± 0.013	0.128 ± 0.005	0.185 ± 0.001	0.183 ± 0.001
Church	0.144 ± 0.003	0.132 ± 0.010	0.164 ± 0.001	0.163 ± 0.001
Derisi	0.144 ± 0.006	0.139 ± 0.005	0.170 ± 0.001	0.169 ± 0.001
Eisen	0.180 ± 0.008	0.148 ± 0.015	0.208 ± 0.001	0.206 ± 0.002
Gasch1	0.134 ± 0.021	0.134 ± 0.010	0.196 ± 0.001	0.199 ± 0.001
Gasch2	0.135 ± 0.013	0.116 ± 0.017	0.184 ± 0.001	0.183 ± 0.003
Pheno	0.100 ± 0.009	0.110 ± 0.009	0.159 ± 0.001	0.155 ± 0.002
Spo	0.146 ± 0.006	0.135 ± 0.007	0.172 ± 0.001	0.168 ± 0.001
Expr	0.097 ± 0.005	0.112 ± 0.012	0.196 ± 0.003	0.200 ± 0.002
Seq	0.123 ± 0.007	0.137 ± 0.004	0.195 ± 0.003	0.203 ± 0.002
Average	0.134	0.129	0.183	0.183

- Number of hidden neurons per level: 0.6/0.5/0.4/0.3/0.2/0.1 for all datasets, except Pheno, Hom and Struc, where the values 0.1/0.08/0.06/0.04/0.03/0.02 were used;
- Learning rate and momentum constant used in Back-propagation for hidden and output layers: $\{0.05, 0.03\}$ and $\{0.02, 0.01\}$, respectively;
- Initial weights of the neural networks: within $[-0.1, 0.1]$;
- Parameter values for Resilient back-propagation: initial Delta $(\Delta_0) = 0.1$, max Delta $(\Delta_{max}) = 50.0$, min Delta $(\Delta_{min}) = 1e^{-6}$, increase factor $(\eta^+) = 1.2$ and decrease factor $(\eta^-) = 0.5$.

Note that we decreased the number of hidden neurons of the neural networks as the hierarchical levels became deeper. Our intention is to avoid over-fitting, since the number of examples is smaller for the networks associated with the deeper hierarchical levels. We believe that this is necessary because, when feeding the networks with an example, we are passing prediction information from a level $l-1$ to a level l even if the example is not assigned to classes of the level l, and we observed that a large number of hidden neurons in the last layers led to over-fitting of the networks. Besides, we used a small number of hidden neurons in the MLPs for the Pheno, Hom and Struc datasets, because we have observed that a higher number of neurons resulted in worst results in their validation dataset, probably due to over-fitting. This may be due to the own nature of the datasets. For instance, the Pheno dataset had only nominal attribute values that were converted into numeric values: -1 (absence) and 1 (presence). The Hom and Struc datasets also have only -1 and 1 as attribute values.

5. Experimental analysis and discussion

Table 3 presents the new $AU(\overline{PRC})$ values for the two variations of HMC-LMLP: conventional Back-propagation (Bp) and Resilient back-propagation (Rprop). The Results reported for HMC-LMLP are the average and standard deviation values for 15 executions, where each execution is performed after randomly initializing the network weights. We also show in Table 3 the results obtained in our previous work [11], where we used the number of 0.5 hidden neurons in all MLPs, and we used the values 0.2/0.1 and 0.1/0.05 for hidden and output layers, respectively, for the learning rate and momentum constant parameters of Back-propagation. Also, in our previous work, we initialized the weights of the neural networks with values within the range $[-1, 1]$. For the Rprop algorithm, we used the same parameter values. We named the HMC-LMLP variations used with the older parameters as Bp-Old and Rprop-Old. Since we are comparing the results obtained in this work with the results obtained in our previous work, Table 3 shows the results for the ten datasets used in our two previous experimental studies [18,11]. As it can be seen, the new parameter values used in this paper improved the predictive performance of HMC-LMLP, for both Bp and Rprop variations.

In Table 4, we show a comparison of our new results with the results obtained by the PCT (decision-tree) methods. The results presented for the decision-tree methods are those provided by Vens et al. [5], since we are using exactly the same training, validation and test datasets. We show the results considering two variations of precision–recall curves, $AU(\overline{PRC})$ and $\overline{AUPRC}_w$. We also show the results obtained for the two new datasets used in the experiments.

5.1. Analysis of the results obtained considering the $AU(\overline{PRC})$ values

As shown in Table 4, the best results overall were obtained by the Clus-HMC method, followed by the two variations of HMC-LMLP, Clus-HSC and Clus-SC. Despite the differences in the results between Clus-HMC and HMC-LMLP, the performance of HMC-LMLP is competitive with the performance of Clus-HMC. When comparing the HMC-LMLP method with the local versions of the PCT methods (Clus-HSC and Clus-SC), we can see that HMC-LMLP performs significantly better than these two methods. Comparing the two variations of HMC-LMLP, we can see that the Rprop version performed better than the Back-propagation version in the Gasch1, Expr and Seq datasets. These datasets have a large number of attributes, which

Table 4
Comparisons between HMC-LMLP and Clus-variations.

Dataset	Bp	Rprop	Clus-HMC	Clus-HSC	Clus-SC
		$AU(\overline{PRC})$ values			
Cellcycle	0.185 ± 0.001	0.183 ± 0.001	0.172	0.111	0.106
Church	0.164 ± 0.001	0.163 ± 0.001	0.170	0.131	0.128
Derisi	0.170 ± 0.001	0.169 ± 0.001	0.175	0.094	0.089
Eisen	0.208 ± 0.001	0.206 ± 0.002	0.204	0.127	0.132
Gasch1	0.196 ± 0.001	0.199 ± 0.001	0.205	0.106	0.104
Gasch2	0.184 ± 0.001	0.183 ± 0.003	0.195	0.121	0.119
Pheno	0.159 ± 0.001	0.155 ± 0.002	0.160	0.152	0.149
Spo	0.172 ± 0.001	0.168 ± 0.001	0.186	0.103	0.098
Expr	0.196 ± 0.003	0.200 ± 0.002	0.210	0.127	0.123
Seq	0.195 ± 0.003	0.203 ± 0.002	0.211	0.091	0.095
Hom	0.195 ± 0.002	0.182 ± 0.004	0.254	0.155	0.153
Struc	0.154 ± 0.0008	0.152 ± 0.0009	0.181	0.118	0.114
Average	0.181	0.180	0.194	0.120	0.118
		$\overline{AUPRC}_w$ values			
Cellcycle	0.145 ± 0.001	0.144 ± 0.002	0.142	0.146	0.146
Church	0.118 ± 0.0007	0.118 ± 0.002	0.129	0.127	0.128
Derisi	0.127 ± 0.0009	0.127 ± 0.001	0.137	0.125	0.122
Eisen	0.163 ± 0.001	0.163 ± 0.003	0.183	0.169	0.173
Gasch1	0.157 ± 0.002	0.158 ± 0.002	0.176	0.154	0.153
Gasch2	0.142 ± 0.001	0.144 ± 0.006	0.156	0.148	0.147
Pheno	0.114 ± 0.0009	0.111 ± 0.002	0.124	0.125	0.127
Spo	0.129 ± 0.001	0.125 ± 0.002	0.153	0.139	0.139
Expr	0.167 ± 0.003	0.165 ± 0.002	0.179	0.167	0.167
Seq	0.166 ± 0.002	0.168 ± 0.002	0.183	0.151	0.154
Hom	0.159 ± 0.002	0.146 ± 0.006	0.240	0.205	0.205
Struc	0.112 ± 0.0008	0.108 ± 0.001	0.161	0.152	0.152
Average	0.142	0.140	0.164	0.151	0.151

suggests that Rprop was more robust to over-fitting. Although the Hom and Struc datasets also have a large number of attributes, the Rprop algorithm obtained results slightly worse than Back-propagation. In this case, we believe that the attribute values (-1 absence and 1 presence) may have contributed to harm the results obtained by Back-propagation and Rprop. This also happened with the Pheno dataset.

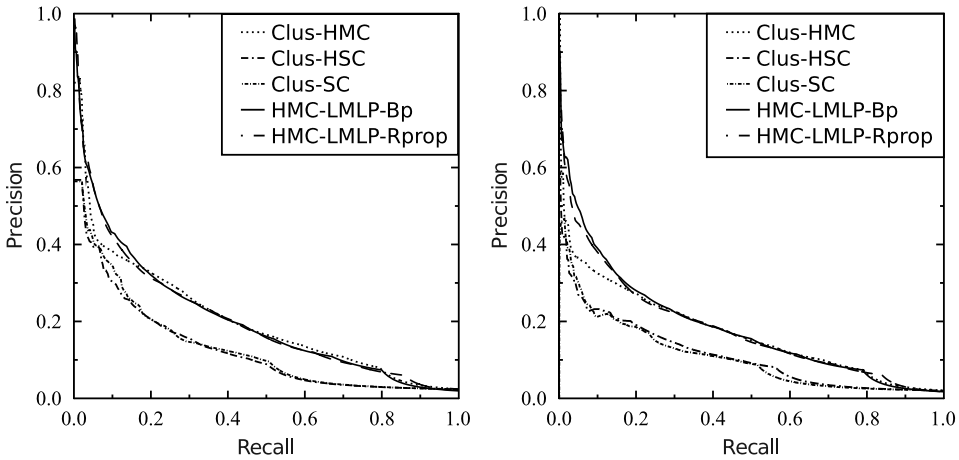
Table 4 also shows that HMC-LMLP performed better than any method for the Cellcycle and Eisen datasets. We chose these two datasets to show the precision and recall results obtained with the application of different thresholds. These results are presented in Table 5. Due to space limitations, we do not show the values obtained with the application of all threshold values that were tested. To compute the PR-curve, we used threshold values ranging within $[0, 1]$, increased by a factor of 0.02 (0.0, 0.02, 0.04, …, 0.98, 1.0). We show part of these thresholds in Table 5, comparing the precision and recall results of the HMC-LMLP variations and Clus-HMC. The threshold values are multiplied by 100. These results are from the run where HMC-LMLP obtained the best $AU(\overline{PRC})$ values in the validation dataset. In Fig. 4, we show the corresponding PR-curves obtained by interpolating the points presented in Table 5.

Comparing the precision and recall values presented in Table 5, we can see that HMC-LMLP obtained higher precision values in the majority of cases, while Clus-HMC obtained higher recall values in the majority of cases. Indeed, we can observe that HMC-LMLP made less prediction in the deeper hierarchical levels (less examples were classified in classes belonging to the deeper levels). We believe that this happened because HMC-LMLP does not use the training examples when making the predictions from the second level onwards, but only the predictions provided by the previous neural networks, which may reduce the chances of predicting the correct class(es) in a deeper level, and also may reduce the number of classes predicted in these levels. A possible improvement to HMC-LMLP would be the employment of the examples for the classification process in all levels, not just for the first level. Clus-HMC, on the other hand, make their predictions considering all classes at once. We believe that this facilitates the correct prediction of deeper classes.

Observing the summary of the datasets in Table 2, it is possible to see how difficult it is for local-based methods to use local information related to each class. The training datasets have very few examples in the last level (from 6 to 11), and each class has an average of two examples, which makes the learning process very difficult in these levels. Fig. 5 illustrates the deeper classes predicted by the HMC-LMLP variations and Clus-HMC considering a threshold value of 12.0 in the Eisen dataset. Note that Clus-HMC makes a larger number of class predictions for this dataset, which increases the recall value (but eventually decreases the precision value). We are aware that the small number of predictions may be seen as a limitation of HMC-LMLP. This may also be due to the blocking problem [45], a common problem in the local-based approaches. As a

Table 5
Precision and recall for the Cellcycle and Eisen datasets, varying the threshold values.

Thres.	Eisen dataset						Cellcycle dataset					
	Bp		Rprop		Clus-HMC		Bp		Rprop		Clus-HMC	
	Prec	Rec	Prec	Rec	Prec	Rec	Prec	Rec	Prec	Rec	Prec	Rec
0.0	0.02	1.0	0.02	1.0	0.02	1.0	0.02	1.0	0.02	1.0	0.02	1.0
4.0	0.11	0.67	0.09	0.72	0.11	0.70	0.11	0.62	0.09	0.71	0.10	0.68
8.0	0.19	0.44	0.17	0.48	0.16	0.51	0.18	0.41	0.17	0.44	0.16	0.47
12.0	0.25	0.32	0.23	0.35	0.22	0.38	0.23	0.29	0.22	0.31	0.20	0.35
16.0	0.31	0.21	0.30	0.23	0.26	0.30	0.30	0.18	0.29	0.18	0.24	0.27
20.0	0.36	0.16	0.35	0.16	0.30	0.25	0.32	0.15	0.32	0.15	0.27	0.21
24.0	0.39	0.14	0.38	0.13	0.32	0.22	0.35	0.13	0.35	0.12	0.30	0.14
28.0	0.41	0.12	0.42	0.10	0.35	0.15	0.37	0.12	0.39	0.10	0.33	0.09
32.0	0.43	0.10	0.46	0.08	0.37	0.12	0.39	0.10	0.42	0.07	0.35	0.07
36.0	0.46	0.08	0.52	0.06	0.41	0.06	0.42	0.09	0.45	0.05	0.36	0.05
40.0	0.49	0.07	0.58	0.05	0.44	0.05	0.44	0.07	0.49	0.04	0.39	0.03
44.0	0.52	0.06	0.63	0.04	0.47	0.04	0.47	0.06	0.55	0.03	0.45	0.02
48.0	0.55	0.05	0.66	0.03	0.52	0.04	0.50	0.05	0.60	0.02	0.46	0.02
52.0	0.58	0.04	0.73	0.02	0.55	0.04	0.52	0.04	0.66	0.01	0.48	0.01
56.0	0.60	0.04	0.83	0.01	0.57	0.03	0.56	0.03	0.72	0.009	0.53	0.01
60.0	0.62	0.03	0.93	0.01	0.59	0.03	0.60	0.03	0.78	0.005	0.57	0.01
64.0	0.68	0.03	0.96	0.007	0.65	0.03	0.63	0.02	0.88	0.003	0.59	0.01
68.0	0.73	0.02	0.97	0.005	0.65	0.03	0.63	0.01	0.91	0.003	0.58	0.007
72.0	0.76	0.02	1.0	0.004	0.72	0.02	0.67	0.01	0.95	0.001	0.57	0.006
76.0	0.82	0.01	1.0	0.001	0.77	0.02	0.67	0.008	0.92	0.001	0.61	0.005
80.0	0.83	0.01	1.0	0.0001	0.80	0.02	0.79	0.006	1.0	0.0004	0.77	0.003
84.0	0.90	0.008	0.0	0.0	0.83	0.01	0.88	0.004	0.0	0.0	0.78	0.003
88.0	0.90	0.006	0.0	0.0	0.83	0.01	0.86	0.002	0.0	0.0	0.78	0.003
92.0	0.97	0.004	0.0	0.0	0.83	0.01	1.0	0.0008	0.0	0.0	0.81	0.003
96.0	0.90	0.001	0.0	0.0	0.83	0.01	1.0	0.0008	0.0	0.0	0.97	0.003
100.0	0.0	0.0	0.0	0.0	0.82	0.01	0.0	0.0	0.0	0.0	0.0	0.0



(a) Eisen dataset (b) Cellcycle dataset

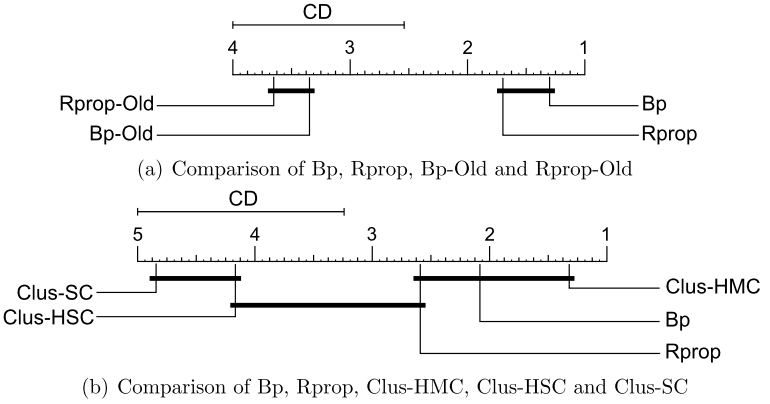
Fig. 4. PR-curves obtained according to the $AU(\overline{PRC})$ measure.

result of the blocking problem, an example may not be classified in any sub-class of the hierarchy in the moment of the top-down classification, or may not be classified in the deeper classes. In the LCN and LCPN strategies, this may occur if examples are rejected by classifiers at higher hierarchical levels and are not passed to classifiers in the following levels [45]. In our case, the blocking problem could be caused by the use of only the output from the previous neural networks by MLPs from the second level onwards. Without using the examples, we believe that the chances of predicting classes from the second level onwards decrease. Different strategies can be used to alleviate the effects of the blocking problem. One of them is to reduce, by some factor, the threshold values used for classifying examples as the level of the hierarchy becomes deeper. The analysis of blocking reduction strategies for hierarchical classification problems is out of the scope of this work. The reader is referred to Sun et al. [45] for different blocking reduction strategies.

This HMC-LMLP characteristic of predicting fewer classes in the deeper levels can also be observed in Table 5 when analyzing the results provided with the application of larger threshold values. For these threshold values (last rows of

01.05	01.05	01.01.03.02.01	11.02.01	20.01.01.01
02	02	01.03	11.02.02	20.01.03.01
10.03.01	10.01	01.04	11.02.03.01	20.01.10
11.02.03.04	10.03.01	01.05.02.07	11.02.03.04	20.01.15
12	11.02.03.04	01.05.03.03.04	11.04.01	20.03.22
14.07	12	01.06	11.04.03.01.10	20.09.01
16	14.07	01.20.19.01	11.06.01	20.09.04
18	16	02.01	12.01.01	20.09.07.03
20.01	18	02.04	12.04.01	20.09.16.09.03
20.09	20.01	02.10	12.04.02	20.09.18
32	20.09	02.11	12.07	30.05.02.24
34	30	02.13.01	12.10	32.01.07
40	32.01	02.13.03	14.01	32.01.09
42	34	02.16	14.04	32.05.01.03.03
43	40	02.19	14.07.11	32.07
(a)	41	02.45.15	14.10	34.01.01.03
	42	10.01.03	14.13.01.01	34.11.03.07
	43	10.01.05.01	16.01	40.01
	(b)	10.01.05.03.01	16.03.01	41.01.01
		10.01.05.03.03	16.03.03	42.01
		10.01.09.05	16.07	42.04
		10.03.01.01.11	16.17.09	42.10.03
		10.03.02	16.19.03	42.16
		10.03.04.05	16.21.08	43.01.03.05
		10.03.04.09	18.02.01.01	43.01.03.09
			18.02.09	
			(c)	

Fig. 5. Deepest predicted classes in the Eisen dataset when applying a threshold value of 12.0. (a) Back-propagation; (b) Rprop; (c) Clus-HMC.



(a) Comparison of Bp, Rprop, Bp-Old and Rprop-Old

(b) Comparison of Bp, Rprop, Clus-HMC, Clus-HSC and Clus-SC

Fig. 6. Results of the application of the Nemenyi statistical test considering $AU(\overline{PRC})$.

Table 5), the precision values provided by HMC-LMLP are larger than those provided by Clus-HMC, and the recall values are much lower. For a threshold value of 100.0, the precision and recall values are 0.0, which means that no predictions were made. Larger precision values are an indicative that the classifier is more suitable in predicting classes located in the levels closer to the root, whereas larger recall values are an indicative that the classifier is more suitable in predicting classes located in the deeper levels [24].

The diagrams from Figs. 6(a) and 6(b) show the results of pairwise comparison for all classifiers according to the Nemenyi test in terms of $AU(\overline{PRC})$. We performed two sets of comparisons, first comparing all HMC-LMLP variations (new and old) and second comparing the new HMC-LMLP results with Clus-HMC, Clus-HSC and Clus-SC. The p-values obtained with the application of the Friedman test were, for the first and second set of comparisons, $2.35 \times e^{-10}$ and $1.20 \times e^{-17}$ respectively.

In the diagrams, we connected the groups of classifiers that were not significantly different. There were no statistically significant differences between HMC-LMLP and Clus-HMC. When comparing HMC-LMLP with Clus-HSC and Clus-SC, the statistical tests indicate that the differences are statistically significant. The differences between the results when using Rprop and Back-propagation were also not statistically significant. When we compared Bp and Rprop with their older versions Bp-Old and Rprop-Old, we can see that, with the new parameter values, the new versions obtained statistically better results. The new parameter values, mainly smaller learning rates and smaller initial weights, helped in reducing the effect of data over-fitting.

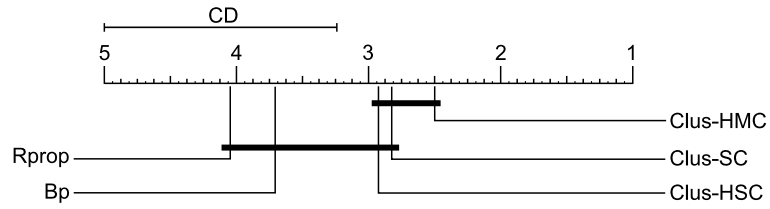


Fig. 7. Results of the application of the Nemenyi statistical test considering $\overline{AUPRC}_w$.

Table 6
$AU(\overline{PRC})$ values obtained in specific hierarchical classes for the Eisen dataset.

Classes	Bp	Rprop	Clus-HMC	Clus-HSC	Clus-SC
01	0.522 ± 0.007	0.501 ± 0.014	0.392	0.444	0.444
12	0.675 ± 0.010	0.680 ± 0.010	0.508	0.523	0.523
12.01	0.204 ± 0.026	0.130 ± 0.028	0.582	0.655	0.651
12.01.01	0.178 ± 0.030	0.125 ± 0.030	0.609	0.724	0.706
14	0.435 ± 0.004	0.449 ± 0.010	0.396	0.352	0.352
Average	0.403	0.377	0.497	0.540	0.535

Table 7
$AU(\overline{PRC})$ values obtained in specific hierarchical classes for the Expr dataset.

Classes	Bp	Rprop	Clus-HMC	Clus-HSC	Clus-SC
01	0.556 ± 0.019	0.533 ± 0.012	0.444	0.457	0.457
10	0.515 ± 0.011	0.511 ± 0.011	0.405	0.380	0.380
12	0.644 ± 0.041	0.617 ± 0.040	0.510	0.489	0.489
12.01	0.239 ± 0.089	0.143 ± 0.057	0.583	0.610	0.581
12.01.01	0.178 ± 0.081	0.117 ± 0.044	0.607	0.677	0.609
Average	0.426	0.384	0.510	0.523	0.503

5.2. Analysis of the results obtained considering the $\overline{AUPRC}_w$ values

As it can be seen in Table 4, the best predictive results, considering the $\overline{AUPRC}_w$ measure, were obtained by the PCT-based methods. As previously mentioned, HMC-LMLP makes fewer predictions as the hierarchical levels become deeper. This can explain its worse predictive performance when considering $\overline{AUPRC}_w$.

As an example, consider the Hom and Struct datasets. It is possible to notice that, with the simple label-transformation strategy used to reduce their dimensionality, HMC-LMLP obtained competitive $AU(\overline{PRC})$ results when compared with Clus-HSC and Clus-SC. Regarding $\overline{AUPRC}_w$, however, the predictive performance of HMC-LMLP was considerably decreased.

The $\overline{AUPRC}_w$ evaluation measure ignores the ability of the classifier to learn class frequencies, averaging the performance of individual classes [5]. This has an impact in the evaluation performance of HMC-LMLP, because the method concentrates its predictions in the more frequent classes (classes in the levels closest to the root).

Despite the decrease in its predictive performance, HMC-LMLP still obtained predictive performances similar to Clus-HSC and Clus-SC, like the performances obtained for the Derisi, Gasch1, Expr, and Seq datasets. HMC-LMLP also obtained a predictive performance similar to Clus-HMC for the Cellcycle dataset.

It is worth mentioning that the PCT-based methods were executed to optimize each of the evaluation measures, whereas the results shown for HMC-LMLP were obtained considering only the $AU(\overline{PRC})$ for the validation phase. The execution of two sets of experiments, first considering the $AU(\overline{PRC})$ during the validation and then considering the $\overline{AUPRC}_w$, could improve the results obtained using $\overline{AUPRC}_w$ in the evaluation.

The diagram of Fig. 7 shows the Nemenyi test results for the pairwise comparisons with all pairs of classifiers regarding $\overline{AUPRC}_w$. The p-value obtained with the application of the Friedman test was $2.0 \times e^{-4}$. According to this test, Clus-HMC obtained statistically better results than HMC-LMLP. However, no statistically significant differences were found when comparing HMC-LMLP, Clus-HSC and Clus-SC.

5.3. Analysis of the results obtained for specific classes

In this section, we analyze the results obtained for specific hierarchical classes. We chose the five classes where Clus-HMC obtained the best $AU(\overline{PRC})$ values in the training dataset. Besides, considering the absence of some training classes in the test dataset, we selected the classes with the best training performances that also have at least a 5% frequency in the test dataset. These results are shown in Tables 6 to 8. Due to space restrictions, we only show the three datasets where Clus-HMC obtained the largest average $AU(\overline{PRC})$ values.

Table 8
$AU(\overline{PRC})$ values obtained in specific hierarchical classes for the Hom dataset.

Classes	Bp	Rprop	Clus-HMC	Clus-HSC	Clus-SC
01	0.647 ± 0.008	0.583 ± 0.028	0.684	0.602	0.602
01.04	0.085 ± 0.008	0.095 ± 0.019	0.552	0.493	0.439
12	0.235 ± 0.028	0.179 ± 0.030	0.479	0.387	0.387
12.01.01	0.073 ± 0.008	0.079 ± 0.004	0.553	0.524	0.487
20	0.547 ± 0.008	0.491 ± 0.020	0.569	0.467	0.467
Average	0.317	0.285	0.567	0.495	0.476

Table 9
$AU(\overline{PRC})$ values obtained in specific hierarchical classes for the Eisen and Expr dataset.

Classes	Bp	Rprop	Clus-HMC	Clus-HSC	Clus-SC
		Eisen dataset			
10.03.01	0.134 ± 0.008	0.131 ± 0.008	0.181	0.169	0.154
11.02.03	0.238 ± 0.008	0.251 ± 0.015	0.236	0.243	0.215
Average	0.186	0.191	0.208	0.206	0.184
		Expr dataset			
10.03.01	0.168 ± 0.008	0.157 ± 0.009	0.181	0.162	0.153
11.02.03	0.221 ± 0.013	0.229 ± 0.014	0.200	0.209	0.225
Average	0.194	0.193	0.190	0.185	0.189

As shown in Tables 6 to 8, HMC-LMLP obtained competitive results with the PCT-based methods regarding the $AU(\overline{PRC})$ obtained for the classes from the first hierarchical level. According to the results obtained by HMC-LMLP for the classes from the second and third levels, the predictive performances in these classes presented a high decrease. These results can be a consequence of the smaller number or predictions made by HMC-LMLP from the second level onwards. They all indicate that a large number of examples and their use in the training process is needed to obtain better results in deeper levels.

Despite the worse predictive results obtained by HMC-LMLP for the classes from the levels 2 and 3, there is an indicative that the information (predictions) learned in a neural network associated with the level l were useful for the training process of the neural network associated with the level $l+1$. This can be seen in the results obtained by HMC-LMLP for the Eisen and Expr datasets, considering the third level classes 10.03.01 and 11.02.03. These results are shown in Table 9.

According to the results shown in Table 9, considering the classes 10.03.01 and 11.02.03, HMC-LMLP now obtained predictive results more similar to Clus-HMC, Clus-HSC, and Clus-SC. This suggests that information learned by the neural network associated with the first level was useful in the learning process of the neural network associated with the second level. Similarly, the information learned in the second level was useful to induce the classifier associated with the third level. These results can be improved by adding, in each level, the examples to the training process, and also use the learned information (predictions) to complement the feature vectors of the examples for training.

5.4. Comparing Bp and Rprop regarding the number of training epochs

Table 10 shows the average number of training epochs needed by HMC-LMLP to obtain its best predictive accuracy results. At each epoch, we calculate the $AU(\overline{PRC})$ resulting from the validation dataset, storing the network with the best weight values using the early stop strategy. A small number of training epochs were needed to obtain results that are competitive with the decision-tree induction methods compared.

Regarding the number of training epochs needed for the Expr and Seq datasets, we can see that Rprop took many more epochs than Back-propagation, but obtained better classification results (Table 4, $AU(\overline{PRC})$ values). As already discussed, this fact suggests that Rprop can better cope with over-fitting caused by the large number of attributes. We can see that the training process of the Back-propagation algorithm ends very rapidly, probably due to over-fitting. Techniques for attribute selection specific for multi-label problems [46,47] could be used or adapted to improve the predictive performance of the methods. Regarding the Hom and Struc datasets, we believe that the range of the attributes, -1 for absence and 1 for presence, led to over-fitting of the networks, as suggest the very poor results obtained in the classes from the second and third levels of the Hom dataset (Table 8). Even with more training epochs, the Rprop algorithm obtained worse results than the Back-propagation algorithm.

6. Conclusions and future work

This paper presented a thorough experimental extension of our previous works [18,11]. In our previous works, we presented a new local-based method for hierarchical multi-label classification named Hierarchical Multi-label Classification with Local Multi-Layer Perceptron (HMC-LMLP). The method incrementally trains a multi-layer perceptron network for each

Table 10
Average number of training epochs needed for HMC-LMLP to provide its results.

Dataset	Bp	Rprop
Cellcycle	14.47 ± 3.94	15.60 ± 1.88
Church	26.67 ± 6.99	27.73 ± 13.9
Derisi	18.20 ± 4.19	23.53 ± 3.76
Eisen	17.47 ± 4.37	15.73 ± 1.62
Gasch1	10.13 ± 2.33	17.67 ± 1.88
Gasch2	24.53 ± 7.56	25.47 ± 7.97
Pheno	54.47 ± 21.5	26.93 ± 10.3
Spo	15.27 ± 4.23	16.40 ± 2.38
Expr	04.27 ± 1.79	17.87 ± 1.55
Seq	03.93 ± 0.80	16.53 ± 1.12
Hom	42.07 ± 8.21	57.40 ± 9.13
Struc	26.20 ± 10.7	43.93 ± 13.3

level of the hierarchy. Each neural network is associated with a level and responsible for the predictions in its level. The predictions of a neural network in a given level are used as inputs to the next neural network associated with the next level.

We extended our previous works by performing experiments with a new set of parameter values that considerably improved the previously obtained classification results. We evaluated the results considering two variations of the area under the precision–recall curves, denoted $AU(\overline{PRC})$ and $\overline{AUPRC}_w$. Additionally, we added two new datasets to the experiments and extended the experimental analysis including results obtained in specific classes of the hierarchy.

We also compared our new results with the results obtained by one state-of-the-art decision tree learner and two other decision-tree based methods, all three based on Predictive Clustering Trees (PCT), and experimentally showed that HMC-LMLP can achieve competitive results compared with the global (state-of-the-art) version of the PCT-based methods, Clus-HMC, regarding $AU(\overline{PRC})$. The experimental comparison also showed that HMC-LMLP achieved superior or competitive predictive performances if compared with the local versions of the PCT-based methods, Clus-HSC and Clus-SC in terms of $AU(\overline{PRC})$ and $\overline{AUPRC}_w$.

We also pointed out some characteristics of HMC-LMLP that can be considered a disadvantage of the method if compared to global methods. We discussed about the smaller number of predictions HMC-LMLP makes in the deeper hierarchical levels, and argued that this characteristic can be a consequence of the use of the training examples only in the first hierarchical level. Also, we argued that HMC-LMLP may suffer from the blocking problem. Based on the application of different threshold values, we demonstrated how HMC-LMLP obtained high precision values in the majority of the cases, whereas Clus-HMC obtained high recall values in the majority of the cases. Higher precision values are an indicative that the classifier is better in predicting classes in the levels closest to the root, whereas higher recall values indicate that the classifier in better in prediction the classes located in the deepest hierarchical levels.

In order to overcome this disadvantage of HMC-LMLP, we pointed out some strategies that can be used as future work to improve its performance. Instead of using the training examples as inputs only for the neural network associated to the first hierarchical level, we plan on using the training examples in all hierarchical levels. In this case, the outputs of a neural network associated to a previous level would be used as an additional information (attribute values) for the training examples that would be used to train the neural network associated to the following level. In each level, however, not all examples would be used, but only the examples that are assigned to classes that belong to the level which is associated to the neural network being trained. Another strategy to try to increase the number of predicted classes in the deeper hierarchical levels would be to reduce, by some factor, the threshold values used for classification as the level of the hierarchy becomes deeper. As lower threshold values lead to a higher recall, it is expected that the number of predicted classes would increase in the deeper levels.

As HMC related datasets can have a very high dimensionality, with hundreds or even thousands of attributes, the use of specific techniques for feature selection can be a good strategy to improve the HMC-LMLP predictive performance and to reduce its execution time. In our experiments, we showed how a high number of attributes can reduce the performance of the Back-propagation algorithm, which had a premature end of its training process in the datasets with the larger number of attributes. We also used a simple label-transformation strategy for two datasets to reduce their dimensionality, and showed that HMC-LMLP obtained competitive results with the other two local-based methods regarding $AU(\overline{PRC})$.

As an additional future work, we plan to test our approach in other domains such as multi-label hierarchical text categorization, and also use hierarchies structured as directed acyclic graphs in future experiments. Such hierarchies contain a much larger number of classes, and differently from the tree structured hierarchies, each class can have more than one superclass at the same time, making the classification process even more complex. To cope with DAG structured hierarchies, modifications will be required in the HMC-LMLP training and testing processes. To use hierarchies structured as DAGs, an initial idea would be the use of a transformation strategy to map the DAG structure into a tree structure. In this way, HMC-LMLP could be applied without modifications. We are aware that, with a much larger number of classes, the induction time of the neural networks would be very high. However, we could implement HMC-LMLP in a parallel way, which would significantly reduce its execution time.

Different training algorithms can also be used in HMC-LMLP. One algorithm which would require a very small modification in our method in order for it to be used is the Improved Resilient Back-propagation [48], which also uses information about the network error in the moment of the weight update. Other neural network approaches, such as Radial Basis Function [49], can also be used as base classifiers to our method.

Acknowledgments

We would like to thank the Brazilian research agencies CNPq and FAPESP. We would also like to thank Dr. Celine Vens and Dr. Leander Schietgat for providing support with the PCT-based methods.

References

[1] A. Freitas, A.C. Carvalho, A tutorial on hierarchical classification with applications in bioinformatics, in: Research and Trends in Data Mining Technologies and Applications, Idea Group, 2007, pp. 175–208 (Ch. VII).
[2] A. Clare, R.D. King, Predicting gene function in Saccharomyces cerevisiae, Bioinformatics 19 (2003) 42–49.
[3] J. Struyf, H. Blockeel, A. Clare, Hierarchical multi-classification with predictive clustering trees in functional genomics, in: Workshop on Computational Methods in Bioinformatics, in: Lect. Notes Artif. Intell., vol. 3808, Springer, 2005, pp. 272–283.
[4] H. Blockeel, L. Schietgat, J. Struyf, S. Dzeroski, A. Clare, Decision trees for hierarchical multilabel classification: A case study in functional genomics, in: Knowledge Discovery in Databases, 2006, pp. 18–29.
[5] C. Vens, J. Struyf, L. Schietgat, S. Džeroski, H. Blockeel, Decision trees for hierarchical multi-label classification, Mach. Learn. 73 (2008) 185–214.
[6] R. Alves, M. Delgado, 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, C. Johnson, A hierarchical multi-label classification ant colony algorithm for protein function prediction, Memetic Comput. 2 (2010) 165–181.
[8] R. Cerri, A.C.P.L.F. Carvalho, Hierarchical multilabel classification using top-down label combination and artificial neural networks, in: Brazilian Symposium on Artificial Neural Networks, 2010, pp. 253–258.
[9] G. Obozinski, G. Lanckriet, C. Grant, W. Jordan, M.I. Noble, Consistent probabilistic outputs for protein function prediction, Genome Biology 9 (Suppl. 1).
[10] G. Valentini, True path rule hierarchical ensembles for genome-wide gene function prediction, IEEE/ACM Trans. Comput. Biol. Bioinform. 8 (3) (2011) 832–847.
[11] R. Cerri, R. Barros, A. de Carvalho, Hierarchical multi-label classification for protein function prediction: A local approach based on neural networks, in: Intelligent Systems Design and Applications (ISDA), 2011, pp. 337–343.
[12] J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.
[13] V.N. Vapnik, The Nature of Statistical Learning Theory, Springer-Verlag, New York, 1999.
[14] E. Costa, A. Lorena, A. Carvalho, A. Freitas, N. Holden, Comparing several approaches for hierarchical classification of proteins with decision trees, in: II Brazilian Symposium on Bioinformatics, in: Lect. Notes in Bioinform., vol. 4643, Springer-Verlag, Berlin, Heidelberg, 2007, pp. 126–137.
[15] C. Silla, A. Freitas, A survey of hierarchical classification across different application domains, Data Min. Knowl. Discov. 22 (2010) 31–72.
[16] G. Valentini, True path rule hierarchical ensembles, in: International Workshop on Multiple Classifier Systems, 2009, pp. 232–241.
[17] S. Kiritchenko, S. Matwin, A.F. Famili, Hierarchical text categorization as a tool of associating genes with gene ontology codes, in: European Workshop on Data Mining and Text Mining in Bioinformatics, 2004, pp. 30–34.
[18] R. Cerri, A.C.P.L.F. Carvalho, Hierarchical multilabel protein function prediction using local neural networks, in: Brazilian Symposium on Bioinformatics, 2011, pp. 10–17.
[19] D.E. Rumelhart, J.L. McClelland, Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol. 1, MIT Press, Cambridge, MA, 1986.
[20] M. Riedmiller, H. Braun, A Direct adaptive method for faster backpropagation learning: The RPROP algorithm, in: International Conference on Neural Networks, 1993, pp. 586–591.
[21] N. Cesa-Bianchi, C. Gentile, L. Zaniboni, Incremental algorithms for hierarchical classification, Mach. Learn. 7 (2006) 31–54.
[22] N. Cesa-Bianchi, G. Valentini, Hierarchical cost-sensitive algorithms for genome-wide gene function prediction, J. Mach. Learn. Res. 8 (2010) 14–29.
[23] G. Valentini, M. Re, Weighted true path rule: a multilabel hierarchical algorithm for gene function prediction, in: 1st Workshop on Learning from Multi-Label Data (MLD) held in conjunction with ECML/PKDD, 2009, pp. 132–145.
[24] N. Cesa-Bianchi, M. Re, G. Valentini, Synergy of multi-label hierarchical ensembles, data fusion, and cost-sensitive methods for gene functional inference, Mach. Learn. 88 (1–2) (2012) 209–241.
[25] M. Re, G. Valentini, Simple ensemble methods are competitive with state-of-the-art data integration methods for gene function prediction, J. Mach. Learn. Res. — Proc. Track 8 (2010) 98–111.
[26] A. Mayne, R. Perry, Hierarchically classifying documents with multiple labels, in: 2009 IEEE Symposium on Computational Intelligence and Data Mining, CIDM 2009 — Proceedings, 2009, pp. 133–139.
[27] R. Cerri, A.C.P.L.F. Carvalho, New top-down methods using SVMs for hierarchical multilabel classification problems, in: Proceedings of the International Joint Conference on Neural Networks (IJCNN), IEEE, Barcelona, 2010, pp. 3064–3071.
[28] G. Tsoumakas, I. Katakis, I.P. Vlahavas, Mining multi-label data, in: Data Mining and Knowledge Discovery Handbook, 2nd edition, Springer, 2010, pp. 667–685.
[29] X. Shen, M. Boutell, J. Luo, C. Brown, Multi-label machine learning and its application to semantic scene classification, Proc. SPIE 5307 (2004) 188–199.
[30] R. Cerri, A.C.P.L.F. Carvalho, A.A. Freitas, Adapting non-hierarchical multilabel classification methods for hierarchical multilabel classification, Intelligent Data Analysis 15 (6) (2011) 861–887, http://dx.doi.org/10.3233/IDA-2011-0500.
[31] H. Blockeel, M. Bruynooghe, S. Dzeroski, J. Ramon, J. Struyf, Hierarchical multi-classification, in: Workshop on Multi-Relational Data Mining, 2002, pp. 21–35.
[32] L. Schietgat, C. Vens, J. Struyf, H. Blockeel, D. Kocev, S. Dzeroski, Predicting gene function using hierarchical multi-label decision tree ensembles, BMC Bioinformatics 11 (2010) 2.
[33] L. Breiman, Bagging predictors, Mach. Learn. 24 (2) (1996) 123–140.
[34] R. Alves, M. Delgado, A. Freitas, Multi-label hierarchical classification of protein functions with artificial immune systems, in: III Brazilian Symposium on Bioinformatics, in: Lect. Notes in Bioinform., vol. 5167, Springer-Verlag, Berlin, Heidelberg, 2008, pp. 1–12.
[35] L.N. de Castro, J.I. Timmis, Artificial Immune Systems: A New Computational Intelligence Approach, Springer-Verlag, London, 2002.
[36] S. Sangsuriyun, S. Marukatat, K. Waiyamai, Hierarchical Multi-label Associative Classification (HMAC) using negative rules, in: International Conference on Cognitive Informatics, IEEE, 2010, pp. 919–924.
[37] M. Dorigo, V. Maniezzo, A. Colorni, Positive feedback as a search strategy, Tech. rep., Dipartimento di Elettronica, Politecnico di Milano, IT, 1991.

[38] M. Dorigo, Optimization, learning and natural algorithms, PhD thesis, Dipartimento di Elettronica, Politecnico di Milano, IT, 1992.
[39] F. Otero, A. Freitas, C. Johnson, A hierarchical classification ant colony algorithm for predicting gene ontology terms, in: European Conference on Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics, in: Lect. Notes Comput. Sci., Springer, 2009, pp. 68–79.
[40] A. Ruepp, A. Zollner, D. Maier, K. Albermann, J. Hani, M. Mokrejs, I. Tetko, U. Güldener, G. Mannhaupt, M. Münsterkötter, H.W. Mewes, The FunCat, a functional annotation scheme for systematic classification of proteins from whole genomes, Nucl. Acids Res. 32 (18) (2004) 5539–5545.
[41] I.H. Witten, E. Frank, Data Mining: Practical Machine Learning Tools and Techniques, 2nd edition, Morgan Kaufmann, 2005.
[42] J. Davis, M. Goadrich, The relationship between precision–recall and ROC curves, in: International Conference on Machine Learning, 2006, pp. 233–240.
[43] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[44] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I.H. Witten, The WEKA data mining software: an update, SIGKDD Explor. Newsl. 11 (1) (2009) 10–18.
[45] A. Sun, E. Peng Lim, W.K. Ng, J. Srivastava, Blocking reduction strategies in hierarchical text classification, IEEE Trans. Knowl. Data Eng. 16 (2004) 1305–1308.
[46] K. Trohidis, G. Tsoumakas, G. Kalliris, I. Vlahavas, Multilabel classification of music into emotions, in: International Conference on Music Information Retrieval, 2008.
[47] G. Doquire, M. Verleysen, Feature selection for multi-label classification problems, in: Proceedings of the 11th International Conference on Artificial Neural Networks Conference on Advances in Computational Intelligence, Springer-Verlag, Berlin, Heidelberg, 2011, pp. 9–16.
[48] C. Igel, M. Husken, Improving the Rprop learning algorithm, in: International Symposium on Neural Computation, 2000.
[49] M.J.D. Powell, Radial basis functions for multivariable interpolation: a review, in: Algorithms for Approximation, Clarendon Press, New York, NY, USA, 1987, pp. 143–167.