Multi-label Classification for Tree and Directed Acyclic Graphs Hierarchies

Mallinali Ramírez-Corona, L. Enrique Sucar, and Eduardo F. Morales

Instituto Nacional de Astrofísica, Óptica y Electrónica,
Luis Enrique Erro No. 1, Sta. Ma. Tonantzintla, Puebla, 72840, México
{mallinali.ramirez,esucar,emorales}@inaoep.mx

Abstract. Hierarchical Multi-label Classification (HMC) is the task of assigning a set of classes to a single instance with the peculiarity that the classes are ordered in a predefined structure. We propose a novel HMC method for tree and Directed Acyclic Graphs (DAG) hierarchies. Using the combined predictions of locals classifiers and a weighting scheme according to the level in the hierarchy, we select the "best" single path for tree hierarchies, and multiple paths for DAG hierarchies. We developed a method that returns paths from the root down to a leaf node (Mandatory Leaf Node Prediction or MLNP) and an extension for Non Mandatory Leaf Node Prediction (NMLNP). For NMLNP we compared several pruning approaches varying the pruning direction, pruning time and pruning condition. Additionally, we propose a new evaluation metric for hierarchical classifiers, that avoids the bias of current measures which favor conservative approaches when using NMLNP. The proposed approach was experimentally evaluated with 10 tree and 8 DAG hierarchical datasets in the domain of protein function prediction. We concluded that our method works better for deep, DAG hierarchies and in general NMLNP improves MLNP.

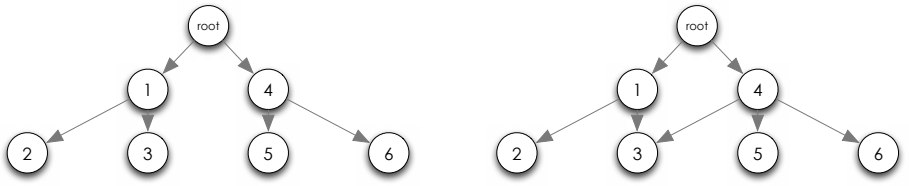
1 Introduction

The traditional classification task deals with problems where each example e is associated with a single label $y \in L$, where L is the set of classes. However, some classification problems are more complex and multiple labels are needed. This is called multi-label classification. A multi-label dataset D is composed of N instances $(x_1, J_1), (x_2, J_2), ..., (x_N, J_N)$, where $J \subset L$. The task is called Hierarchical Multi-label Classification (HMC) when the labels are ordered in a predefined structure, typically a tree or a DAG (Direct Acyclic Graph), the main difference between them is that in the DAG a node can have more than one parent node.

In hierarchical classification, an example that belongs to certain class automatically belongs to all its superclasses (hierarchy constraint), e.g., in Figure 1b an instance that belongs to class node 3 also belongs to nodes 1, 4 and root.

Some major applications of HMC can be found in the fields of text categorization [10], protein function prediction [13], music genre classification [12], phoneme classification [6], etc.

L.C. van der Gaag and A.J. Feelders (Eds.): PGM 2014, LNAI 8754, pp. 409–425, 2014.



(a) Tree hierarchy (b) DAG hierarchy

Fig. 1. An example of a tree and a DAG structure

Two general approaches can be distinguished in HMC [14]. The first is a global approach that builds a single classification model, taking into account the class hierarchy as a whole. These methods are incapable of handling large scale datasets because the models become too complex and thus time consuming. The second is a local approach that divides the problem in several subproblems according to a strategy (can be a local classifier per level, per node or per non leaf node). The main problem of this approach is that it does not incorporate the relations (underlying structure) in the local classification.

We propose a novel HMC approach, Chained Path Evaluation (CPE). CPE belongs to the local approaches so it can work effciently with large scale datasets; a local classifier is trained for each non-leaf node in the hierarchy. To include the relations between the classes, and diminish the limitation of local approaches, an extra attribute is added to the instances in each node which corresponds to the parent node class according to the hierarchy. We also incorporated a weighting scheme to value more the predictions of the more general classes than the more particular ones. CPE scores all the paths in the hierarchy to select the best one. CPE predicts single paths from the root down to a leaf node for tree hierarchies (e.g., in Figure 1a 2, 1, root) and multiple paths for DAG hierarchies (e.g., in Figure 1b 3, 1, 4, root).

We developed an extension of the base method for Non Mandatory Leaf Node Prediction (NMLNP); in which a pruning phase is performed to select the best path. We compared several pruning approaches, the best approach for the task was to prune and then choose the optimal path in a top-down fashion, using the most probable child as the condition to prune a node. Additionally, we proposed a new evaluation metric for hierarchical classifiers, that avoids the bias of current measures which favor conservative predictions (predictions of short paths that only predict the most general classes) when using NMLNP.

The proposed approach was experimentally evaluated with 10 tree and 8 DAG hierarchical datasets in the domain of protein function prediction. We concluded that our method in both versions, MLNP and NMLNP, is competitive with other methods in the state of the art, and performs better in deeper, DAG hierarchies.

The document is organized as follows. Section 2 reviews the relevant work in the area, Section 3 describes the method in detail, Section 4 outlines the framework for the experiments, Section 5 evaluates experimentally our approach, and Section 6 summarizes the paper and suggests possible future work.

2 Related Work

When the labels in a multi-label classification problems are ordered in a predefined structure, typically a tree or a Direct Acyclic Graph (DAG), the task is called Hierarchical Multi-label Classification (HMC). The class structure represent an "IS-A" relationship, these relations in the structure are asymmetric (e.g., all cats are animals, but not all animals are cats) and transitive (e.g., all Siameses are cats, and all cats are animals; therefore all Siameses are animals). In hierarchical classification, there are basically two types of classifiers: global classifiers and local classifiers.

Global classifiers construct a global model and train it to predict all the classes of an instance at once. Vens et al. [15] present a global method that applies a Predicting Clustering Tree (PCT) to hierarchical multi-label classification, transforms the problem in a hierarchy of clusters with reduced intra-cluster variance. One problem of global classifiers is that the computational complexity grows exponentially with the number of labels in the hierarchy.

Local classifiers can be trained in three different ways: a Local Classifier per hierarchy Level (LCL), that trains one multi-class classifier for each level of the class hierarchy; training a Local binary Classifier per Node (LCN), where each classifier decides if a node is predicted or not; the third way is training a Local Classifier per Parent Node (LCPN), where a multi-class classifier is trained to predict its child nodes.

Cerri et al. [5] propose a method that incrementally trains a multilayer perceptron for each level of the classification hierarchy (LCL). Predictions made by a neural network at a given level are used as inputs to the network of the next level. The labels are predicted using a threshold value. Finally, a post processing phase is used to correct inconsistencies (when a subclass is predicted but its superclass is not). Some difficulties of this approach are the selection of a correct threshold and the need of a post-processing phase.

Alaydie et al. [1] developed HiBLADE (Hierarchical multi-label Boosting with LAbel DEpendency), an LCN algorithm that takes advantage of not only the predefined hierarchical structure of the labels, but also exploits the hidden correlation among the classes that is not shown through the hierarchy. This algorithm attaches the predictions of the parent nodes as well as the related classes. However, appending multiple attributes can create models that over-fit the data.

Silla et al. [12] propose an LCPN algorithm combined with two selective methods for training. The first method selects the best features to train the classifiers, the second selects both the best classifier and the best subset of features simultaneously, showing that selecting a classifier and features improves the classification performance. A drawback of this approach is that the selection of the best features and the best classifier for each node can be a time-consuming process.

Bi et al. [3,4] propose HIROM, a method that uses the local predictions to search for the optimal consistent multi-label classification using a greedy strategy. Using Bayesian decision theory, they derive the optimal prediction rule by

minimizing the conditional risk. The limitations of this approach is that it optimizes a function that does not necessarily maximizes the performance in other measures.

The approach of Hernandez et al. [7], used for tree structured taxonomies, learns an LCPN. In the classification phase, it classifies a new instance with the local classifier at each node, and combines the results of all of them to obtain a score for each path from the root to a leaf-node. Two fusion rules were used to achieve this: product rule and sum rule. Finally it returns the path with the highest score. One limitation of this method is that it favors shorter (product rule) or longer paths (sum rule) depending on which combination rule is used. Another limitation is that it does not take into account the relations between nodes when classifying an instance.

Extending the work of Hernandez et al., our method (Chained Path Evaluation or CPE), changes the way the classifiers are trained to include the relations between the labels, specifically of the parent nodes of the labels, to boost the prediction. The score for each path is computed using a fusion rule that takes into account the level in the hierarchy, thus minimizing the effect that the length of the path has in the score. We also extended the method to work with DAG structured hierarchies.

To include the relations of the parent nodes we used the idea of chain classifiers proposed by Read et al. [9] and further extended by Zaragoza et al. [16]. The chain classifiers proposed by Read link the classifiers along a chain where each classifier deals with the binary classification problem associated with a label. The feature space of each classifier in the chain is extended with the $0/1$ label of all the previous classifiers in the chain. Zaragoza et al. propose a Bayesian Chain Classifier where they obtain a dependency structure out of the data. This structure determines the order of the chain, so that the order of the class variables in the chain is consistent with the structure found in the first stage. We adapt this idea to a hierarchical classifier, such that the chain structure is determined by the hierarchy.

3 Chained Path Evaluation

Let D be a training set with N examples, $e_k = (x_k, J_k)$, where x_k is a d-dimensional feature vector and $J \subset L$, $L = \{l_1, 1_2, ..., l_M\}$ a finite set of M possible labels. These labels are represented as $Y \in \{0,1\}^M$, where $y_i = 1$ iff $y_i \in J_k$ else $y_i = 0$. The parent of label y_i in the hierarchy is represented as $pa(y_i)$, the children nodes as $child(y_i)$ and the siblings as $sib(y_i)$, the siblings include all the children nodes of $pa(y_i)$ except y_i. Our method exploits the correlation of the labels with its ancestors in the hierarchy and evaluates each possible path from the root to a leaf node, taking into account the level of the predicted labels to give a score to each path and finally return the one with the best score. The method is composed of two phases: training and classification. There is an additional optional phase, pruning, which can be applied for non-mandatory leaf node prediction.

3.1 Training

The method trains local classifiers per parent node (LCPN). A multi-class classifier C_i is trained for each non leaf node y_i. The classes in C_i are the labels in the set of $child(y_i)$ plus an "*unknown*" label that corresponds to the instances that do not belong to any $child(y_i)$.

The training set for C_i is composed of two sets. The positive training set ($Tr^+(C_i)$) consists of the instances where $child(y_i) = 1$. Each instance in this set will be labeled with the corresponding $child(y_i)$ label. The negative training set ($Tr^-(C_i)$) consists of instances in $sib(y_i)$, in case y_i has no siblings this set will include the uncle nodes, these instances are labeled as "*unknown*". The number of instances on $Tr^-(C_i)$ is proportional to the average of the training examples for each $child(y_i)$ to create a balanced training set. The idea behind $Tr^-(C_i)$ is to include instances where the associated label of the parent has the value zero. The intuition is that the instances that has the parent label set as zero, will have less probability to be predicted as true that the ones that have the parent predicted as one.

As in multidimensional classification, the class of each node in the hierarchy is not independent from the other nodes. To incorporate these relations, inspired by chain classifiers, we include the class predicted by the parent node(s) as an additional attribute in the LCPN classifier. That is, the feature space of each node in the hierarchy is extended with the 0/1 label association of the parent (tree structure) or parents (DAG structure) of the node, as in a Bayesian Chain Classifier [16].

3.2 Classification

The classification phase consists in calculating for each new instance with feature vector x_e, the probability of a node i to occur given the feature vector and the prediction of the parents at each label $P(y_i = 1|x_e, pa(y_i))$. When the structure of the dataset is a DAG it is possible to obtain more than one prediction for one class, then the associated prediction is the average of the prediction of all the parents for that class. After computing a probability for each node, the predictions are merged using a rule to obtain a score for each path.

Merging Rule. The rule that merges the predictions of each local classifier into one score considers the level in the hierarchy of the node to determine the weight that this node will have in the overall score. Misclassifications at the upper hierarchy levels (which correspond to more generic concepts) are more expensive than those at the lower levels (which correspond to more specific concepts).

To achieve this task, the weight of a node ($w(y_i)$) is defined in Equation (2) and depicted on Figure 2, where $level(y_i)$ is the level at which the node y_i is placed in the hierarchy (Equation (1)). For a tree structure it is simply the weight of its parent plus one, and for DAG structures it is computed as the mean of the levels of the m parents ($pa(y_i)$) of the node (y_i) plus one. Finally, $maxLevel$ is the length of the longest path in the hierarchy. This way of computing the weight

of each node assures that the weights are well distributed along the hierarchy; so that the weights of the lower levels do not tend rapidly to zero, as in other approaches [3,15].

$$level(y_i) = 1 + \frac{\sum_{j=1}^{m} level(pa(y_i)_j)}{|pa(y_i)|} \tag{1}$$

$$w_i = 1 - \frac{level(y_i)}{maxLevel + 1} \tag{2}$$

Equation (3) describes the merging rule which is the sum of the logarithms of the probabilities on the nodes along the path or paths (when it is a DAG), where n is the number of nodes in the path, h_i is the ith node in the path and $P(h_i = 1|x_e, pa(h_i))$ is the probability of the node h_i to be predicted as true by the local classifier. Taking the sum of logarithms is used to ensure numerical stability when computing the probability for long paths. Figure 2 depicts the classification procedure.

$$score = \sum_{i=1}^{n} w_{h_i} * log(P(h_i|x_e, pa(h_i))) \tag{3}$$

This scheme assumes independence between the labels, although in an indirect way the dependencies with the parent nodes are considered by incorporating them as additional attributes. As in chain classifiers, this scheme looks for a balance between classification accuracy and computational complexity.

For DAG structures there might be numerous paths from the root to one leaf node. In that case, all the paths that end in that leaf node are returned.

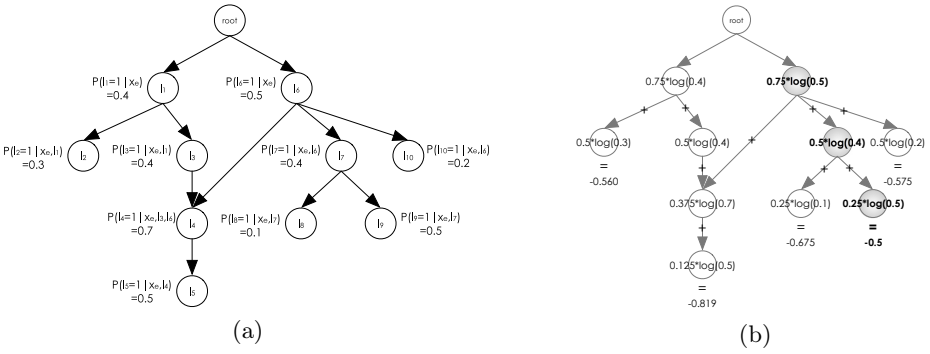


Fig. 2. Example of the application of the merging rule. (a) Each node has an associated probability. (b) The local probabilities are combined to obtain a score for each path. The path with highest score is highlighted.

3.3 Pruning

Sometimes the information available is not sufficient to estimate the class of an instance at the lower levels in the hierarchy, so it could be better to truncate

the predicted path at some level, this is known as non-mandatory leaf node prediction (NMLNP). We introduce a pruning phase to obtain NMLNPs. We consider three decisions that need to be taken into account for pruning: pruning direction, pruning time and pruning condition.

Pruning direction. Determines the way the hierarchy is traversed to prune:

1. Top-Down. The hierarchy is traversed starting from the root node, when the pruning condition is met in one node, the traversing is stopped and the descendants of the node are pruned.
2. Bottom-Up. The hierarchy is traversed starting from the leaf nodes, when the pruning condition is met in one node, the traversing is stopped and the node and its descendants are pruned.

Pruning time. Determines when to perform the pruning stage:

1. Prune & Choose. Prune the hierarchy before the classification phase.
2. Choose & Prune. Prune the path that the classification phase selected.

Pruning condition. Establishes the condition to fulfill to prune a node:

1. Sum children probabilities (SUM). Prunes if the sum of the probabilities of the children is less than the probability of the 'unknown' label.
2. Most probable child (BEST). Prunes if the probability of the most probable child is less than the probability of the 'unknown' label.
3. Information Gain (IG). Prunes if there is not information gain when including the child in the prediction.

In the experiments we compared the different pruning strategies.

4 Experimental Setup

The proposed method, Chained Path Evaluation (CPE), was evaluated experimentally with several tree and DAG structured hierarchies, using five different evaluation metrics and compared with several state of the art hierarchical classification techniques.

4.1 Databases

Eighteen datasets were used in the tests, these datasets are from the field of functional genomics[1]. Ten of them (tree structured) are labeled using the FunCat annotation scheme [11]. The remaining eight datasets (DAG structured) are labeled using the Gene Ontology vocabulary [2]. From the set of paths that each instance owned we selected only the first to get instances with just one path. We prunned the hierarchy to obtain nodes with enough instances (more than 50) to train. The two tables in Table 1 represent two datasets: the first used in

[1] `http://dtai.cs.kuleuven.be/clus/hmcdatasets/`

MLNP experiments (all the class paths end on a leaf node) and the second used in NMLNP experiments (the class paths does not always end on a leaf node); they have the same names and number of attributes but not the same number of labels and instances, because the prunning was more exhaustive in MLNP case.

Table 1. Description of the datasets for the experiments. M=Number of Labels, A=Number of Attributes, N=Number of Instances and D=Maximum Depth.

(a) MLNP experiments

Dataset	M	A	N	D
Tree Hierarchies				
cellcylcle_FUN	36	77	2339	4
church_FUN	36	29	2340	4
derisi_FUN	37	65	2381	4
eisen_FUN	25	81	1681	3
expr_FUN	36	553	2346	4
gasch1_FUN	36	175	2356	4
gasch2_FUN	36	54	2356	4
pheno_FUN	17	71	1162	3
seq_FUN	39	480	2466	4
spo_FUN	36	82	2302	4
DAG Hierarchies				
cellcycle_GO	53	77	1708	11
church_GO	53	29	1711	11
derisi_GO	54	65	1746	11
expr_GO	53	553	1720	11
gasch1_GO	53	175	1716	11
gasch2_GO	53	54	1720	11
seq_GO	52	480	1711	11
spo_GO	53	82	1685	11

(b) NMLNP experiments

Dataset	M	A	N	D
Tree Hierarchies				
cellcylcle_FUN	49	77	3602	4
church_FUN	49	29	3603	4
derisi_FUN	49	65	3675	4
eisen_FUN	35	81	2335	4
expr_FUN	49	553	3624	4
gasch1_FUN	49	175	3611	4
gasch2_FUN	49	54	3624	4
pheno_FUN	22	71	1462	3
seq_FUN	51	480	3765	4
spo_FUN	49	82	3553	4
DAG Hierarchies				
cellcycle_GO	56	77	3516	11
church_GO	56	29	3515	11
derisi_GO	57	65	3485	11
expr_GO	56	553	3537	11
gasch1_GO	56	175	3524	11
gasch2_GO	56	54	3537	11
seq_GO	59	480	3659	11
spo_GO	56	82	3466	11

4.2 Evaluation Metrics

Measures for conventional classification are not adequate for hierarchical multi-label classification, for that reason specific measures for HMC have been proposed. In our work we use four of the most common evaluation metrics and propose a new metric.

Let M be the number of labels in L, N the number of instances in the training set, y_i the real set of labels and $\hat{y}_i$ the predicted set of labels. The labels of an instance are represented in a $0/1$ vector of size M where the predicted/real labels are set to 1 and the rest with 0.

Accuracy. Is the ratio of the size of the union and intersection of the predicted and actual label sets, taken for each example, and averaged over the number of examples.

$$Accuracy = \frac{1}{N}\sum_{i=1}^{N}\frac{|y_i \wedge \hat{y}_i|}{|y_i \vee \hat{y}_i|} \tag{4}$$

Exact Match. The exact match represents the proportion of the real label sets that were predicted.

$$ExactMatch = \frac{1}{N}\sum_{i=1}^{N} 1_{y_i=\hat{y}_i} \tag{5}$$

F1-measure. F1-measure (F1) is calculated as in Equation (6) but redefining precision and recall as:

precision as the fraction of predicted labels which are actually true $\frac{|z_i \wedge \hat{z}_i|}{|\hat{z}_i|}$.

recall as the fraction of true labels which are also predicted $\frac{|z_i \wedge \hat{z}_i|}{|z_i|}$.

$$F1 = \frac{2 \times precision \times recall}{precision + recall} \tag{6}$$

We have specified a vector z instead of the y_i vector, because in the multi-label context there are two ways to average this measure.

F1-macro D (Equation (7)) is averaged by instances; we obtain N vectors of $z_i \equiv y_i$.

$$F1macro^{\times D}(D) = \frac{1}{N}\sum_{i=0}^{N} F1(z_i, \hat{z}_i) \tag{7}$$

F1-macro L (Equation (8)) is averaged by labels; we obtain M vectors of $z_i \equiv [y_i^1, ..., y_i^N]$.

$$F1macro^{\times L}(D) = \frac{1}{M}\sum_{i=0}^{M} F1(z_i, \hat{z}_i) \tag{8}$$

Gain-Loose Balance. In this paper we propose a new evaluation measure for hierarchical classifieres that avoids conservative predictions when using NMLNP. Gain-Loose Balance (GLB) is a measure that rewards the nodes that are correct and penalizes the ones that are incorrect. The rewards and penalties are determined using the number of siblings of the node and the depth of the node in the hierarchy.

Based on the notion that discriminating few categories is much easier than discriminating many of them, a correctly classified node with few siblings has a minor impact on the rewards than one with many. On the contrary, a misclassified node with few sibling has a mayor impact on the penalty than one with many.

A correctly classified node that belongs to a deep level in the hierarchy has more impact on the rewards than one in shallow levels, because reaching the most specific node is the goal of the prediction. In contrast, a deeper misclassified node in a deep level of the hierarchy has less impact in the penalty than one in shallow levels, while the predicted classification is near to the real it become less expensive.

Equation (9) describes the GLB measure, where n_p is the number of correct classified labels, n_{fp} is the number of false positive errors, n_{fn} is the number of

false negative errors, n_t is the number of true labels; N represents the number of siblings plus one (the node that is being evaluated).

$$\frac{\sum_{i=0}^{n_p}(1-\frac{1}{N})(1-w_i)}{\sum_{i=0}^{n_t}(1-\frac{1}{N})(1-w_i)} - \left(\sum_{i=0}^{n_{fp}}\frac{1}{N}w_i + \sum_{i=0}^{n_{fn}}\frac{1}{N}w_i\right) \tag{9}$$

Gain-Loose Balance ranges from 1 (when the predicted path is equal to the real path) to $-\frac{maxL}{2}$ (see Equation (10)), where $maxL$ is the maximum number of levels in the hierarchy (see Equation (1)). In the worst case scenario the node has just two sibling and $N = 2$. w_i is defined in Equation (2).

As we know the maximum and minimum values of the GLB measure we transformed it into a (0, 1) range maintaining the ratio.

$$minValue = -2\sum_{i=1}^{maxL}\frac{1}{N}w_i = -2\sum_{i=1}^{maxL}\frac{1}{2}\left(1-\frac{i}{maxL+1}\right) \tag{10}$$

$$= -2\left(\sum_{i=1}^{maxL}\frac{1}{2} - \frac{1}{2(maxL+1)}\sum_{i=1}^{maxL} i\right) \tag{11}$$

$$= -maxL + \frac{maxL}{2} \tag{12}$$

5 Experiments

The proposed method, Chained Path Evaluation (CPE), was evaluated experimentally with a number of tree and DAG structured hierarchies and compared with various hierarchical classification techniques. We performed three sets of experiments to: (i) compare our method using MLNP with other state of the art techniques, (ii) evaluate the different pruning strategies, (iii) analyze the NMLNP alternative, comparing it with MLNP and other method.

For MLNP related experiments we used the datasets in Table 1a and considered the first four evaluation metrics, for NMLNP related we used Table 1b and considered the five metrics, including GLB. The results were obtained by a stratified 10-fold cross-validation. The best results are marked in bold. Random Forest was used as base classifier for CPE in all the experiments because is the base classifier that best suits the data in our method.

5.1 Comparison of CPE-MLNP against other Methods

Results are summarized in Tables 2 and 3, the complete set of tables is presented in Appendix A.1.

Tree Structured Datasets. For tree structured hierarchies, we compared CPE against three HMC methods:

1. Top-Down LCPN (TD). Proposed by Koller et al. [8], this method trains a LCPN and selects at each level the most probable node. Only the children of this node are explored to preserve the consistency of the prediction. Random Forest was used as base classifier. This is the most typical approach for MHC.
2. Multidimensional Hierarchical Classifier (MHC). Proposed by Hernandez et al. [7]. Random Forest was used as base classifier as reported by the authors. This is the method that we are extending.
3. HIROM. Proposed by Bi et al. [4]. The used base classifier was Support Vector Machines as reported by the authors. This is a recent work in local based HMC.

Table 2. Comparing CPE against other methods in tree structured datasets

Metric	CPE	TD	MHC	HIROM
Accuracy	**23.63**	20.67	19.93	3.10
Exact Match	**18.33**	16.63	9.79	3.05
F1-macro D	**26.30**	22.76	25.07	3.12
F1-macro L	13.79	**14.38**	2.44	0.86

DAG Structured Datasets. For DAG structured datasets, we compared CPE against tree HMC methods:

1. Top-Down LCPN (TD).
2. Top-Down LCPN Corrected (TD-C). The only difference between this method and TD is that when a leaf node is reached, all the paths to that node are appended to the final prediction. TD returns a single path.
3. HIROM. The variant for DAG structures.

Table 3. Comparing CPE against other methods in DAG structured datasets

Metric	CPE	TD	TD-C	HIROM
Accuracy	**38.74**	36.48	36.42	19.02
Exact Match	**21.87**	18.59	19.22	0.0
F1-macro D	**48.86**	46.76	46.57	29.28
F1-macro L	13.68	16.18	**16.94**	2.93

Discussion. For tree structured datasets we observe that the proposed method is superior in terms of accuracy and exact match to other methods, and in most cases the difference is significant. In the case of F1-macro D and F1-macro L, it is superior to HIROM and competitive with MHC and TD. For DAG hierarchies, our method is clearly superior for the first three measures and for the fourth measure it is beaten by TD-C. One possible explanation is that it is difficult to design a method that is better in all measures, however the proposed approach is overall competitive in all measures and superior in some.

F1-macro L metric is the exception where CPE never obtains the best results. In this metric the results are averaged by labels, that means that there is probably a label(s) which does not have many instances classified, where our method could fail.

5.2 Selection of the Best NMLNP Approach

The pruning approaches described in Subsection 3.3 for the NMLNP version of CPE were tested to select the best one. In the case of the DAG structured datasets the root of the hierarchy has only one child and this child (l_0) is parent of the rest of the nodes of the hierarchy. The problem in this kind of hierarchies is that most measures score the conservative classifications as the better ones. In this case, the method *"Top-Down, Select & Prune, IG"* predict just the l_0 for every new instance, this classification is useless due to the fact that every instance belongs to l_0 and nevertheless is the one that is better scored.

Gain-Loose Balance deals with this problem and gives better scores to other methods that return relevant predictions. For that reason, and the fact that the other measures were inconsistent along the databases and the different structures, the methods were compared using Gain-Loose Balance. The results for the NMLNP approaches are depicted on Table 4, the results are averaged along the datasets.

Table 4. Comparison in terms of GLB (%) of the different approaches for NMLNP in tree and DAG structures

Dataset	Top-Down						Bottom-Up					
	Prune & Select			Select & Prune			Prune & Select			Select & Prune		
	SUM	BEST	IG	SUM	BEST	IG	SUM	BEST	IG	SUM	BEST	IG
Tree/DAG	71.02	**71.53**	68.71	69.79	70.12	69.31	70.34	70.61	68.68	69.66	69.83	68.58

Discussion. We observe that *"Top-Down, Prune & Select, BEST"* method obtains in most of the cases the better score in both, tree and DAG structures. The datasets have approximately 16% percent of instances which real label set is just the label l_0 . Since the average number of labels per instance is three, when a method predict only l_0 it already has 1/3 of the correct answer. This can be one of the reasons why the rest of the metrics give high scores with the *"Top-Down, Select & Prune, IG"* method.

5.3 Comparison of CPE NMLNP-version against MLNP-version

We compared the best NMLNP method (Top-Down, Prune & Select, BEST) against the MLNP to determine if it is worth to prune the hierarchies. Table 5 depicts the results. The datasets in Table 1b were split by hierarchy structure and the results averaged along the datasets.

Table 5. Gain-Loose Balance Metric (%) comparing NMLNP against MLNP

Dataset structure	NMLNP	MLNP
Tree	**61.688**	59.419
DAG	**83.826**	80.262

Discussion. In every dataset the results obtained by NMLNP version were significantly superior compared to the MLNP version for NMLNP datasets. Thus, prunning obtains better scores than returning complete paths to a leaf node according to the GLB metric.

5.4 Comparison of CPE-NMLNP against Other Methods

We compared CPE-NMLNP against HIROM, a method proposed by Bi et al. [4] which has a variant for DAG structures. The base classifier used for HIROM was SVM as reported by the authors. The results are depicted on Table 6.

Table 6. Comparison of MLNP methods

(a) Tree structured datasets

Metric	CPE	HIROM
Accuracy	**17.10**	3.98
Exact Match	**11.16**	3.89
F1-macro D	**19.87**	4.02
F1-macro L	**11.51**	0.76
GLB	**59.41**	41.35

(b) DAG structured datasets

Metric	CPE	HIROM
Accuracy	**32.29**	14.92
Exact Match	**9.31**	0.02
F1-macro D	**44.68**	24.81
F1-macro L	**8.56**	3.33
GLB	**80.18**	64.03

Discussion. CPE obtains better results than HIROM in most of the datasets along all the metrics, most of them with statistical relevance. This can be due to the fact that HIROM optimizes a loss function that does not necessarily improves its score in other metrics. Other fact is that HIROM is designed to return multiple paths instead of just one.

6 Conclusions and Future Work

We presented a novel approach for hierarchical multi-label classification for tree and DAG structures. The method estimates the probability of each path by combining LCPNs, including a pruning phase for NMLNP. A new metric was introduced that avoids conservative classifications. Experiments with 18 tree and DAG hierarchies show that: (i) the proposed method is competitive compared against other state-of-the-art methods for tree hierarchies and superior for DAGs, (ii) the best pruning strategy is top-down, prune first and based on the most probable child; (iii) NMLNP improves mandatory leaf-node prediction.

As future work we plan to extend the proposed method for multiple path prediction.

References

1. Alaydie, N., Reddy, C.K., Fotouhi, F.: Exploiting label dependency for hierarchical multi-label classification. In: Tan, P.-N., Chawla, S., Ho, C.K., Bailey, J. (eds.) PAKDD 2012, Part I. LNCS, vol. 7301, pp. 294–305. Springer, Heidelberg (2012)
2. Ashburner, M., Ball, C.A., Blake, J.A.: Gene Ontology: Tool for the unification of biology. Nature Genetics 25, 25–29 (2000)
3. Bi, W., Kwok, J.T.: Multilabel classification on tree- and dag-structured hierarchies. In: Proc. of the 28th Inter. Conf. on ML (ICML), pp. 17–24. Omnipress (2011)
4. Bi, W., Kwok, J.T.: Hierarchical multilabel classification with minimum bayes risk. In: IEEE Intl. Conf, on Data Mining (ICDM), pp. 101–110. IEEE Computer Society (2012)
5. Cerri, R., Barros, R.C., de Carvalho, A.C.P.L.F.: Hierarchical multi-label classification using local neural networks. J. Comput. System Sci. 1, 1–18 (2013)
6. Dekel, O., Keshet, J., Singer, Y.: An online algorithm for hierarchical phoneme classification. In: Bengio, S., Bourlard, H. (eds.) MLMI 2004. LNCS, vol. 3361, pp. 146–158. Springer, Heidelberg (2005)
7. Hernandez, J.N., Sucar, L.E., Morales, E.F.: A hybrid global-local approach for hierarchical classification. In: Boonthum-Denecke, C., Youngblood, G.M. (eds.) Proceedings FLAIRS Conference, pp. 432–437. AAAI Press (2013)
8. Koller, D., Sahami, M.: Hierarchically classifying documents using very few words. In: Proceedings of the 14th International Conference on Machine Learning, ICML 1997, pp. 170–178. Morgan Kaufmann Publishers Inc., San Francisco (1997)
9. Read, J., Pfahringer, B., Holmes, G., Frank, E.: Classifier chains for multi-label classification. Machine Learning, 254–269 (2011)
10. Rousu, J., Saunders, C., Szedmak, S., Shawe-Taylor, J.: Kernel-based learning of hierarchical multilabel classification models. J Mach. Learn. Res. 7, 1601–1626 (2006)
11. Ruepp, A., Zollner, A., Maier, D., Albermann, K., Hani, J., Mokrejs, M., Tetko, I., Güldener, U., Mannhaupt, G., Münsterkötter, M., Mewes, H.W.: The FunCat, a functional annotation scheme for systematic classification of proteins from whole genomes. Nucleic Acids Research 32(18), 5539–5545 (2004)
12. Silla Jr., C.N., Freitas, A.A.: Novel top-down approaches for hierarchical classification and their application to automatic music genre classification. In: IEEE Inter. Conf. on Systems, Man, and Cybernetics, pp. 3499–3504 (2009)
13. Silla Jr., C.N., Freitas, A.A.: A global-model naive bayes approach to the hierarchical prediction of protein functions. In: Proceedings of the 2009 9th IEEE International Conference on Data Mining, ICDM 2009, pp. 992–997. IEEE Computer Society, Washington, DC (2009)
14. Tsoumakas, G., Katakis, I.: Multi-Label Classification: An Overview. Int. J. Data Warehouse. Min. 3, 1–13 (2007)
15. Vens, C., Struyf, J., Schietgat, L., Džeroski, S., Blockeel, H.: Decision trees for hierarchical multi-label classification. Machine Learning 73(2), 185–214 (2008)
16. Zaragoza, J.H., Sucar, L.E., Morales, E.F., Bielza, C., Larrañaga, P.: Bayesian Chain Classifiers for Multidimensional Classification. Comp. Intelligence, 2192–2197 (2011)

A Appendix

For the comparison of more than two methods (Appendix A.1) we performed a Friedman test and for the comparison of two methods (Appendix A.2) we performed a one tailed t-test; both with a confidence degree of 95%. Statistically inferior results against CPE are marked with $\downarrow$ and statistically superior results are marked with $\uparrow$.

A.1 Comparison of CPE-MLNP against Other Methods

Table 7. Tree structured datasets

(a) Accuracy (%)

Dataset	CPE	TD	MHC	HIROM
cellcycle_FUN	**22.76**	20.14	19.74	4.20↓
church_FUN	14.24	12.79	**19.71**	6.30↓
derisi_FUN	18.58	13.20	**19.43**	4.41↓
eisen_FUN	**31.07**	27.25	21.77	3.27↓
expr_FUN	**29.46**	23.36	19.63↓	1.96↓
gasch1_FUN	**28.39**	19.83	19.66↓	2.83↓
gasch2_FUN	**22.32**	19.57	19.63	3.94↓
pheno_FUN	18.39	**23.0**	22.49	6.63
seq_FUN	30.52	**32.36**	17.30	2.48↓
spo_FUN	**20.57**	15.22↓	19.92	3.82↓

(b) Exact Match (%)

Dataset	CPE	TD	MHC	HIROM
cellcycle_FUN	**17.53**	17.02	9.92	4.10↓
church_FUN	**11.20**	10.90	9.87	6.24↓
derisi_FUN	**13.27**	10.63	9.70	4.28↓
eisen_FUN	**23.32**	**23.32**	9.99↓	3.27↓
expr_FUN	**23.47**	20.20	9.85↓	1.78↓
gasch1_FUN	**22.12**	16.88	9.85↓	2.77↓
gasch2_FUN	**17.49**	15.58	9.85	3.78↓
pheno_FUN	**14.90**	11.62	9.47↓	6.63↓
seq_FUN	**25.26**	27.90	9.41	2.31↓
spo_FUN	**14.73**	12.29	9.95↓	3.69↓

(c) F1-macro D (%)

Dataset	CPE	TD	MHC	HIROM
cellcycle_FUN	**25.40**	21.71	24.74	4.24↓
church_FUN	15.77	13.87	**24.71**	6.32↓
derisi_FUN	21.27	14.73	**24.38**	4.45↓
eisen_FUN	**34.98**	29.33	27.66	3.27↓
expr_FUN	**32.47**	24.98	24.60↓	2.01↓
gasch1_FUN	**31.53**	21.33↓	24.64	2.85↓
gasch2_FUN	**24.76**	21.62	24.60	3.99↓
pheno_FUN	20.16	28.71	**29.00**↑	6.63
seq_FUN	**33.17**	34.64	21.40	2.53↓
spo_FUN	**23.49**	16.64	24.98	3.87↓

(d) F1-macro L (%)

Dataset	CPE	TD	MHC	HIROM
cellcycle_FUN	14.34	**18.05**	2.09	0.69↓
church_FUN	**7.72**	6.80	2.09	0.98↓
derisi_FUN	**11.54**	10.53	2.01↓	0.70↓
eisen_FUN	**19.59**	17.73	3.21↓	0.76↓
expr_FUN	18.45	**20.30**	2.08	0.57↓
gasch1_FUN	**17.24**	15.73	2.08↓	0.47↓
gasch2_FUN	8.74	**11.22**	2.08	0.61↓
pheno_FUN	**11.14**	9.83	4.86↓	1.46↓
seq_FUN	17.21	**21.05**	1.75	0.71↓
spo_FUN	11.97	**12.54**	2.11↓	0.63↓

Table 8. DAG structured datasets

(a) Accuracy (%)

Dataset	CPE	TD	TC-C	HIROM
cellcycle_GO	**36.60**	34.70	34.64	19.01↓
church_GO	**32.09**	31.11	30.86	19.11↓
derisi_GO	**33.42**	32.61	32.45	18.44↓
expr_GO	**42.80**	38.93	38.91↓	19.18↓
gasch1_GO	**42.03**	39.39	39.23	19.17↓
gasch2_GO	**39.53**	36.44	36.29↓	19.18↓
seq_GO	**48.99**	46.03	46.55	19.26↓
spo_GO	**34.45**	32.63	32.44↓	18.79↓

(b) Exact Match (%)

Dataset	CPE	TD	TD-C	HIROM
cellcycle_GO	**19.26**	16.74	17.27	0.00↓
church_GO	**13.79**	12.45	12.74	0.00↓
derisi_GO	**15.41**	14.09	14.43	0.00↓
expr_GO	**27.33**	21.86↓	22.50	0.00↓
gasch1_GO	**26.28**	22.09	22.73	0.00
gasch2_GO	**22.62**	19.01↓	19.30	0.00↓
seq_GO	**33.31**	28.46↓	30.51	0.00↓
spo_GO	**16.97**	14.01↓	14.24	0.00↓

(c) F1-macro D (%)

Dataset	CPE	TD	TD-C	HIROM
cellcycle_GO	**47.03**	45.19	44.98	29.28↓
church_GO	**43.24**	42.05	41.68	29.40↓
derisi_GO	**44.25**	43.40	43.14	28.59↓
expr_GO	**52.14**	48.77	48.64↓	29.47↓
gasch1_GO	**51.52**	49.27	48.97	29.47↓
gasch2_GO	**49.60**	46.69	46.41↓	29.47↓
seq_GO	**57.94**	55.28	55.55	29.54↓
spo_GO	**45.12**	43.45	43.15	29.03↓

(d) F1-macro L (%)

Dataset	CPE	TD	TD-C	HIROM
cellcycle_GO	10.45	14.32	**15.17**↑	2.92
church_GO	8.72	10.41	**10.51**	2.93↓
derisi_GO	9.50	13.17	**13.35**↑	2.81
expr_GO	17.45	17.82	**18.80**	2.95↓
gasch1_GO	16.70	18.24	**18.29**	2.94↓
gasch2_GO	12.52	15.02	**15.42**	2.95
seq_GO	24.28	28.21	**31.60**↑	3.02
spo_GO	9.79	12.21	**12.38**	2.89

A.2 Comparison of CPE-NMLNP against Other Methods

Table 9. Tree structured datasets

(a) Accuracy (%)

Dataset	CPE	HIROM
cellcycle_FUN	**16.26**	4.2↓
church_FUN	**9.12**	6.3↓
derisi_FUN	**13.71**	4.41↓
eisen_FUN	**21.46**	3.27↓
expr_FUN	**21.44**	1.96↓
gasch1_FUN	**20.92**	2.83↓
gasch2_FUN	**16.44**	3.94↓
pheno_FUN	**13.36**	6.63↓
seq_FUN	**24.05**	2.48↓
spo_FUN	**14.26**	3.82↓

(b) Exact Match (%)

Dataset	CPE	HIROM
cellcycle_FUN	**10.74**	4.2↓
church_FUN	5.86	**6.3**
derisi_FUN	**8.62**	4.41↓
eisen_FUN	**14.91**	3.27↓
expr_FUN	**13.99**	1.96↓
gasch1_FUN	**13.87**	2.83↓
gasch2_FUN	**10.76**	3.94↓
pheno_FUN	**8.35**	6.63↓
seq_FUN	**15.64**	2.48↓
spo_FUN	**8.81**	3.82↓

(c) F1-macro D

Dataset	CPE	HIROM
cellcycle_FUN	**18.87**	4.24↓
church_FUN	**10.7**	6.32↓
derisi_FUN	**16.22**	4.45↓
eisen_FUN	**24.45**	3.27↓
expr_FUN	**24.81**	2.01↓
gasch1_FUN	**24.19**	2.85↓
gasch2_FUN	**19.11**	3.99↓
pheno_FUN	**15.63**	6.63↓
seq_FUN	**27.85**	2.53↓
spo_FUN	**16.84**	3.87↓

(d) F1-macro L (%)

Dataset	CPE	HIROM
cellcycle_FUN	**11.94**	0.69↓
church_FUN	**4.47**	0.98↓
derisi_FUN	**8.58**	0.7↓
eisen_FUN	**13.28**	0.76↓
expr_FUN	**16.21**	0.57↓
gasch1_FUN	**15.82**	0.47↓
gasch2_FUN	**8.28**	0.61↓
pheno_FUN	**10.64**	1.46↓
seq_FUN	**16.36**	0.71↓
spo_FUN	**9.52**	0.63↓

(e) GLB (%)

Dataset	CPE	HIROM
cellcycle_FUN	**58.96**	39.99↓
church_FUN	**56.40**	41.18↓
derisi_FUN	**57.66**	40.03↓
eisen_FUN	**60.96**	44.5↓
expr_FUN	**60.48**	38.78↓
gasch1_FUN	**60.69**	39.2↓
gasch2_FUN	**59.26**	39.89↓
pheno_FUN	**59.6**	50.62↓
seq_FUN	**62.16**	39.55↓
spo_FUN	**57.9**	39.71↓

Table 10. DAG structured datasets

(a) Accuracy (%)

Dataset	CPE	HIROM
cellcycle_GO	**41.08**	16.73↓
church_GO	**40.41**	18.49↓
derisi_GO	**40.98**	17.08↓
expr_GO	**39.15**	14.7↓
gasch1_GO	**40.82**	15.4↓
gasch2_GO	**42.14**	17.19↓
seq_GO	**40.9**	14.08↓
spo_GO	**41.19**	17.18↓

(b) Exact Match (%)

Dataset	CPE	HIROM
cellcycle_GO	**3.61**	0.0↓
church_GO	**3.10**	0.0↓
derisi_GO	**4.10**	0.09↓
expr_GO	**8.76**	0.0↓
gasch1_GO	**7.01**	0.0↓
gasch2_GO	**5.00**	0.0↓
seq_GO	**9.65**	0.0↓
spo_GO	**4.53**	0.0↓

(c) F1-macro D (%)

Dataset	CPE	HIROM
cellcycle_GO	**55.81**	27.43↓
church_GO	**55.27**	29.90↓
derisi_GO	**55.61**	27.85↓
expr_GO	**52.53**	24.59↓
gasch1_GO	**54.80**	25.60↓
gasch2_GO	**56.69**	28.14↓
seq_GO	**54.28**	23.70↓
spo_GO	**55.76**	28.10↓

(d) F1-macro L (%)

Dataset	CPE	HIROM
cellcycle_GO	**5.09**	3.96↓
church_GO	3.77	4.07
derisi_GO	**5.16**	3.93↓
expr_GO	**10.34**	3.34↓
gasch1_GO	**8.87**	3.55↓
gasch2_GO	**4.64**	3.87
seq_GO	**11.44**	2.96↓
spo_GO	**5.37**	4.12↓

(e) GLB (%)

Dataset	CPE	HIROM
cellcycle_GO	**83.83**	65.51↓
church_GO	**83.57**	67.32↓
derisi_GO	**83.73**	65.65↓
expr_GO	**83.55**	63.76↓
gasch1_GO	**83.96**	64.35↓
gasch2_GO	**84.12**	66.27↓
seq_GO	**84.01**	63.24↓
spo_GO	**83.84**	65.87↓