International Journal of Approximate Reasoning ••• (••••) •••–•••

Contents lists available at ScienceDirect

International Journal of Approximate Reasoning

www.elsevier.com/locate/ijar

Hierarchical multilabel classification based on path evaluation ☆

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

Instituto Nacional de Astrofísica, Óptica y Electrónica, Luis Enrique Erro No. 1, Sta. Ma. Tonantzintla, Puebla, 72840, Mexico

ARTICLE INFO

Article history:
Received 16 December 2014
Received in revised form 19 July 2015
Accepted 21 July 2015
Available online xxxx

Keywords:
Multi-label classification
Hierarchical classification
Chain classifiers

ABSTRACT

Multi-label classification assigns more than one label for each instance; when the labels are ordered in a predefined structure, the task is called Hierarchical Multi-label Classification (HMC). In HMC there are global and local approaches. Global approaches treat the problem as a whole but tend to explode with large datasets. Local approaches divide the problem into local subproblems, but usually do not exploit the information of the hierarchy. This paper addresses the problem of HMC for both tree and Direct Acyclic Graph (DAG) structures whose labels do not necessarily reach a leaf node. A local classifier per parent node is trained incorporating the prediction of the parent(s) node(s) as an additional attribute to include the relations between classes. In the classification phase, the branches with low probability to occur are pruned, performing non-mandatory leaf node prediction. Our method evaluates each possible path from the root of the hierarchy, taking into account the prediction value and the level of the nodes; selecting the path (or paths in the case of DAGs) with the highest score. We tested our method with 20 datasets with tree and DAG structured hierarchies against a number of state-of-the-art methods. Our method proved to obtain superior results when dealing with deep and populated hierarchies.

1. Introduction

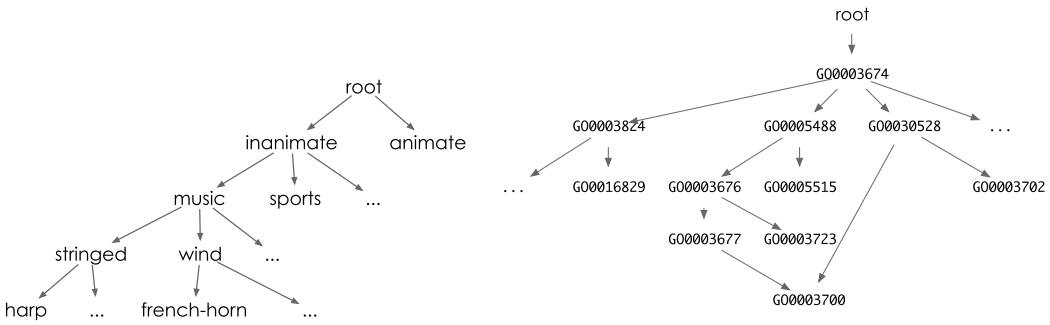
The traditional classification task deals with problems where each example e is associated with a single label $y \in L$, where L is the set of classes, also known as multi-class classification problem. However, some classification problems are more complex and an instance can have multiple labels. When the classes are binary it is called multi-label classification [22,29], when the classes can have multiple class values it is known as multidimensional classification. A multi-label dataset D is composed of N instances, each example has associated a set Y of labels, where $Y \subseteq L$. The task is called Hierarchical Multi-label Classification (HMC) [20] when the labels are ordered in a predefined structure, typically a tree or a Direct Acyclic Graph (DAG); the main difference between them is that in a DAG a node can have more than one parent node (see Fig. 1).

We propose a novel HMC approach, Chained Path Evaluation (CPE), that follows a local classifier approach, by training a local classifier for each non-leaf node in the hierarchy. This decomposition of the problem makes it possible to handle

☆ This is an extended version of the article presented at PGM 2014 [16] that incorporates several additional experiments and a more detailed comparison and analysis. In particular, we include an analysis of the depth effect in the performance of the method, a comparison against the flat approach, and additional experiments with other more complex hierarchies in a different application field.

* Corresponding author.
E-mail addresses: mallinali.ramirez@inaoep.mx (M. Ramírez-Corona), esucar@inaoep.mx (L.E. Sucar), emorales@inaoep.mx (E.F. Morales).

http://dx.doi.org/10.1016/j.ijar.2015.07.008
0888-613X/



(a) Tree structure - Caltech-256 (b) DAG structure - Cellcycle GO

Fig. 1. An example of tree and DAG structured datasets (see Subsection 4.1). The complete hierarchy for the Cellcycle GO data set is shown in Fig. 5 in Appendix A.

large datasets. In contrast to previous local hierarchical approaches, the proposed method has several improvements, that constitute the main contributions of this work. These are listed below and explained in more detail in the paper.

- It adds an extra attribute to the instances in each node with the prediction of its parent node to include the relations between the classes.
- It incorporates a weighting scheme to value more the predictions of the more general classes over more specific ones.
- It scores all the paths, giving a complete overview of the possible predictions, based on the combined probabilities and positions of the nodes in the path for selecting the best one. CPE predicts single paths from the root down to another node (leaf or non-leaf) for tree hierarchies and multiple paths to a single node for DAG hierarchies.
- It obtains predictions where the most specific label is not necessarily a leaf node, what is known as a non-mandatory leaf node prediction (NMLNP), with a novel pruning phase performed before selecting the best path. The pruning discards the branches with less probability to appear in the real label set, thus avoiding potential errors.

We noticed that many evaluation measures score the short paths that only predict the most general classes better than longer and more specific paths, that is why we also propose a new evaluation measure that avoids the bias toward conservative predictions in the case of NMLNP.

The proposed approach was experimentally evaluated with 12 tree and 8 DAG hierarchical datasets in the domain of protein function prediction and image categorization. We concluded that the proposed method, CPE, performs better, than other state-of-the-art methods in deep and populated 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 experimental setup, in Section 5 the proposed approach is evaluated experimentally and contrasted to other methods, and Section 6 summarizes the paper and suggests possible future work.

2. Related work

When the labels in a multi-label classification problem are ordered in a predefined structure, typically a tree or a DAG, the task is called HMC. The class structure is represented using "IS-A" relations; these relations in the structure are asymmetric (e.g., all *harps* are *stringed* instruments, but not all *stringed* instruments are *harps*) and transitive (e.g., all *harps* are *stringed* instruments, and all *stringed* instruments are *music* instruments; therefore all *harps* are *music* instruments).

In HMC, an example that belongs to a certain class automatically belongs to all its superclasses (hierarchy constraint). When a prediction fulfills the hierarchy constraint it is called a consistent prediction. An example (using the hierarchy in Fig. 1a) of a consistent prediction would be *inanimate, music, stringed, harp*; and an example of an inconsistent prediction would be *inanimate, music, stringed, french-horn.*

There are two kinds of predictions [20]: Mandatory Leaf Node Prediction (MLNP), that returns paths that reach a leaf node in the hierarchy, and Non-Mandatory Leaf Node Prediction (NMLNP) that returns paths that can end in an intermediate node of the hierarchy.

HMC methods are grouped according to the exploration policy they use to solve the classification problem. The most common policies or approaches are: flat, global and top-down.

The flat classification approach predicts only the leaf nodes ignoring completely the information of the hierarchy. Traditional multi-label classification algorithms conform to this approach. However, this very simple approach has the serious disadvantage of having to build a classifier to discriminate among a possibly large number of classes (all the leaf nodes), and does not take advantage of the information provided by the hierarchy.

The global approach learns a single global model for all classes, i.e., it is able to predict each class (node) of the hierarchy. For instance, in a binary tree of depth 3 there are 14 internal and leave nodes, so the global approach needs to build a classifier with 14 classes. This generated model takes into account the class hierarchy as a whole during a single run of the

Please cite this article in press as: M. Ramírez-Corona et al., Hierarchical multilabel classification based on path evaluation, International Journal of Approximate Reasoning (2015), http://dx.doi.org/10.1016/j.ijar.2015.07.008

classification algorithm. The problem with global methods is that they become too complex and thus time consuming when the size of the dataset increases. Vens et al. [27], Blockeel et al. [5], Blockeel and Bruynooghe [4] presented a global HMC method. It is a decision tree induction algorithm that is based on Predictive Clustering Trees (PCT). PCT were also used by Dimitrovski et al. [10] to construct ensembles.

In the local classifier approach the hierarchy is taken into account by using a local information perspective. This approach can be further divided based on how the local information is grouped and the classifiers are built. More precisely, there are three main ways of using the local information: a local classifier per node (LCN), a local classifier per parent node (LCPN), and a local classifier per level (LCL). The three types of local hierarchical classification algorithms differ significantly in their training phase but most of them share a very similar top-down approach in their testing phase. In this top-down approach, for each new example in the test set, the system first predicts its root-level (most generic) class, then it uses the predicted class to narrow the choices of classes at the next level, and so on, until the most specific prediction is made. A limitation of the top-down schema, is that an error in the upper levels will propagate down in the hierarchy.

The LCN approach trains one binary classifier for each node of the class hierarchy (except the root node). Bi and Kwok [2,3] proposed HIROM, a method that uses the local predictions (independently of the way they are trained) to search for the optimal consistent multi-label classification using a greedy strategy. Using Bayesian decision theory, they derive an optimal prediction rule by minimizing the conditional risk. The limitation of this approach is that it optimizes a function that does not necessarily maximize the performance in other evaluation measures.

The LCPN approach trains a multi-class classifier for each parent node in the class hierarchy, to distinguish between its child nodes. During the testing phase, this approach can also be coupled with a top-down prediction approach. Valentini and Cesa-Bianchi [24], Valentini [23], Valentini and Re [25] propose another approach for class-membership inconsistency correction based on the output of all classifiers, where the positive decisions for a node influence the decisions of the parents (bottom-up) turning them positive. Negative predictions in a node turn all its descendants negative. This approach is called *True Path Rule* (TPR), latter upgraded to *Weighted True Path Rule* (wTPR), where the nodes acquire a weight in the hierarchy depending on the positive and negative predicted children of the node. This method can propagate the errors in a bottom-up fashion.

The LCL consists on training one multi-class classifier for each level of the class hierarchy. This approach was used as a baseline comparison method in Clare and King [7] and Costa et al. [9].

Unlike most of the local methods, CPE exploits the information given by the hierarchy structure by taking into account the predictions of the parent nodes. CPE reviews all the possible paths to avoid the unrecoverable propagation of local errors, it is also able to deal with DAG hierarchies, and returns NMLNPs.

3. Chained path evaluation

We developed a hierarchical classification method, named Chained Path Evaluation (CPE), that exploits the relation of the labels with its ancestors in the hierarchy. CPE evaluates each possible path from the root to a leaf or intermediate node using a merging rule that takes 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 performs a pruning step before emitting the final prediction to keep just the labels where there is more certainty, thus returning Non-Mandatory Leaf Node Predictions.

3.1. Training

Let D be a training set with N examples, $e_e = (X_e, Y_e)$, where X_e is a d-dimensional feature vector and $Y_e \subseteq L$, $L = \{y_1, y_2, \ldots, y_{|L|}\}$ is a finite set of $|L|$ possible labels or classes. It bears mentioning that regularly Y_e is a small set in comparison with L. Y_e is represented as a 0/1 vector $Y_e \in \{0,1\}^{|L|}$, where $y_i = 1$ if and only if $y_i \in Y_e$ else $y_i = 0$.

In our framework, with the exception of the root nodes, all the nodes in the DAGs represent labels or classes and consequently are denoted by y_is. A multi-class classifier C_i is trained for each non-leaf node y_i (LCPN), henceforth called *base classifier*. Using an LCPN approach has the advantage of scaling well with large hierarchies. Any classifier can be used as *base classifier* as long as it returns a probability value associated with the predicted class or its output can be converted into a probability value. The classes in C_i are the labels of the children of y_i ($child(y_i)$) plus an "*unknown*" label that corresponds to instances that do not belong to $child(y_i)$.

As in multidimensional classification, the class of each node in the hierarchy is not independent of the other nodes. To incorporate these relations, inspired by chain classifiers [17], 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 associated to the parent (tree structure) or parents (DAG structure) of the node, as in Bayesian Chain Classifiers [21,28]. This incorporates information from the ancestors of a node, aiming to improve the accuracy of the classifier, while retaining a low computational complexity by considering only information from the parents.

The set of positive training examples ($Tr^+(C_i)$) consists of the instances where $child(y_i) = 1$, that is, all the instances that include a child of y_i in their set of classes ($child(y_i) \in Y_e$). Each instance in this set will be labeled with the corresponding $child(y_i)$ label. In this set, the added features of the parents will have the value true since all the instances are their children.

The negative training set ($Tr^-(C_i)$) consists of different instances in the siblings of y_i ($sib(y_i)$) or, in the case that y_i has no siblings, the uncles $sib(pa(y_i))$. The siblings include all the child nodes of the parents of y_i ($pa(y_i)$) except y_i. This set

Please cite this article in press as: M. Ramírez-Corona et al., Hierarchical multilabel classification based on path evaluation, International Journal of Approximate Reasoning (2015), http://dx.doi.org/10.1016/j.ijar.2015.07.008

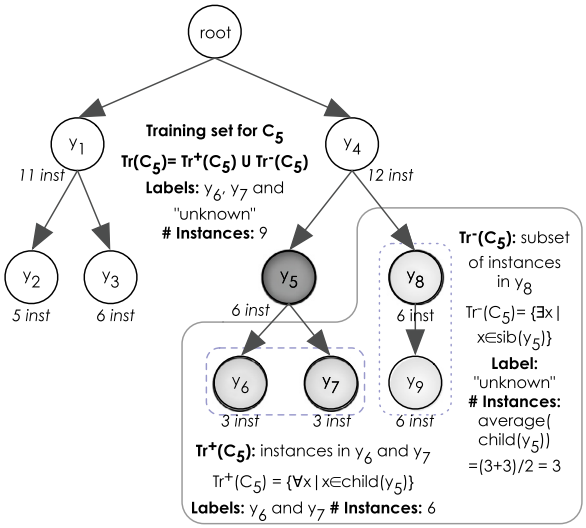


Fig. 2. Example of the selection of the training set for the classifier (C_5) in node y_5. The positive set ($Tr^+(C_5)$) consists of the instances where $child(y_5) = \{y_6, y_7\} = 1$, labeled with the corresponding $child(y_5)$ label. The negative set ($Tr^-(C_5)$) consists of the instances in $sib(y_5) = \{y_8\}$, this set will be labeled as "*unknown*". This set is randomly under-sampled to make the number of instances proportional to the average number of the training examples for $child(y_5)$.

will be labeled as "*unknown*". This set is under-sampled to create a balanced training set. The number of under-sampled instances is proportional to the mean of the training examples of each $e \in child(y_i)$. In this set the labels of the associated parent attributes have a zero value, as the idea is to reduce the probability of an instance whose parent is predicted as false. In Fig. 2 is depicted the selection of the instances that conform the training set for the local classifier C_5 in label node y_5.

The complexity of the training process (Equation (1)) depends on the complexity of the base classifier (in this case represented by *baseClassifier*), multiplied by the number of non-leaf nodes (*nonLeafNodes*).

$$O(nonLeafNodes \times baseClassifier) \tag{1}$$

3.2. Pruning

Sometimes the available information is not sufficient to accurately estimate the class of an instance at the lower levels in the hierarchy, so it is better to truncate the predicted path at some previous level, this is known as non-mandatory leaf node prediction (NMLNP). The set of labels of the instances contained in this kind of datasets does not necessarily contain a leaf node.

Pruning can be done in a top-down or bottom-up way, depending on how the hierarchy is traversed for pruning, it can be performed before the classification phase or by pruning first the path that the classification phase selected, and pruning can take place according to different conditions. In [16], we performed several experiments to evaluate the best strategy to follow for pruning. Based on these results, in this paper we pruned in a *top-down* fashion starting from the root and testing the branches in a descendant way. To classify a new instance, the hierarchy is pruned before computing the score to choose the best path for that instance. The condition to prune a path is if the probability of the most probable child of the node is smaller than the probability of the "*unknown*" label. The rationale for this strategy is that when a classifier cannot predict a class with higher confidence than examples from another class which is not one of its children, then it is safer to prune.

The pruning phase, described in Algorithm 1, is performed after obtaining the probability of each node for a given new instance, x_i. It starts from the root node and successively compares the probability of the most probable child, *maxConfidence* (lines 6–12) of the node y_i, with the probability that the node is not known (*unknown*). If $maxConfidence > unknown$ the process continues with each child of y_i in a recursive call (line 16), else all the descendants of y_i are pruned. If a node has no siblings, the algorithm looks up in the hierarchy for siblings of its ancestors. If there are none, it is equivalent to having a single line of descendants with only one parent, then there are no unknown examples and consequently there is no pruning.

After pruning the hierarchy, the paths from the root to a leaf node (in the pruned tree or DAG) are evaluated by scoring them according to a merging rule.

3.3. Merging rule

The rule that merges the predictions of each local classifier of the path 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

Please cite this article in press as: M. Ramírez-Corona et al., Hierarchical multilabel classification based on path evaluation, International Journal of Approximate Reasoning (2015), http://dx.doi.org/10.1016/j.ijar.2015.07.008

Algorithm 1 Prune. Pseudocode that describes the pruning phase of CPE, where $LC(y_i)$ (line 15) is for running a local classifier on node y_i.

Require: *confidences* (an array containing $P(y_j|x_e, pa(y_j))$ for label y_j in position j), y_i (root node of the hierarchical structure), C (set of classifiers, one for each non-leaf node)
Ensure: a pruned hierarchy

```
1: if y_i is not a leaf and has not been visited before then
2:     if all pa(y_i) have been visited then
3:         mark y_i visited
4:         maxConfidence ← 0
5:         totalConfidence ← 0
6:         for all y_j ∈ child(y_i) do
7:             totalConfidence ← totalConfidence+confidences[j]
8:             if confidences[j] > maxConfidence then
9:                 maxConfidence ← confidences[j]
10:            end if
11:        end for
12:        unknown ← 1 − totalConfidences
13:        if maxConfidence > unknown then
14:            for all y_j ∈ child(y_i) do
15:                LC(y_j)
16:            end for
17:        else
18:            prune descendants of y_i
19:        end if
20:    else
21:        continue with another node
22:    end if
23: end if
```

levels (which correspond to more generic concepts) are more expensive than those at lower levels (which correspond to more specific concepts). The intuition behind the selection of the weights is that at the upper levels there are more examples and the classes have more differences, so there is more information to discriminate the classes. Also, an error at upper levels will cause a wrong path from the beginning, whether if it is at lower levels just part of the path will be wrong.

To achieve this task, the weight of a node ($w(y_i)$) is defined by Equation (3), where $level(y_i)$ is the level at which the node y_i is placed in the hierarchy (Equation (2)).

$$level(y_i) = 1 + \frac{\sum_{j=1}^{m} level(pa(y_i)_j)}{|pa(y_i)|} \tag{2}$$

For a tree structure, the level is simply the level of its parent plus one (to avoid the deepest level to have weight equal to zero), 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. Using the mean of the levels of the parents is a compromise between taking the minimum level of the parents, which overweights some long paths, and taking the maximum level of the parents, which underweights some short paths. Finally, *maxLevel* is the length of the longest path in the hierarchy.

$$w(y_i) = 1 - \frac{level(y_i)}{maxLevel + 1} \tag{3}$$

The previous equation provides a linear decay of the weights as we go down 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 [2], or decrease too slowly [27]. We leave as future work experiments with different weighting strategies.

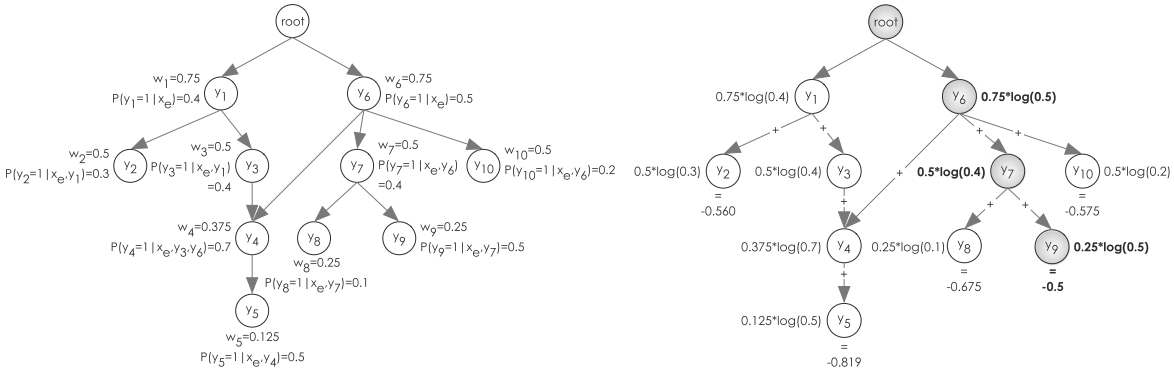
Equation (4) describes the merging rule which is the weighted sum of the logarithms of the predicted probabilities on the nodes along the path; where p is the number of nodes in the path, y_i is the ith node in the path and $P(y_i = 1|x_e, pa(y_i))$ is the probability of the node y_i to be predicted as true by the local classifier.

$$score = \sum_{i=1}^{p} w(y_i) * log(P(y_i|x_e, pa(y_i))) \tag{4}$$

To simplify the computation, 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 Bayesian Chain Classifiers, this scheme aims for a balance between classification accuracy and computational complexity.

The independence assumption implies a product of probabilities; and this is transformed to a sum of logarithms of the probabilities; which guarantees a monotonic transformation (the order is preserved). The sum of logarithms is used to ensure numerical stability; in other words, to minimize approximation errors, when computing the probability for long paths.

Please cite this article in press as: M. Ramírez-Corona et al., Hierarchical multilabel classification based on path evaluation, International Journal of Approximate Reasoning (2015), http://dx.doi.org/10.1016/j.ijar.2015.07.008



(a) Each node has an associated weight, w_i, and a probability, $P(y_i|x_e, pa(y_i))$, given by the base classifier.

(b) The probability is used to compute the overall score of the path, the path with the best score is returned as final prediction.

Fig. 3. An example of the application of the merging rule. The weight and probability for each node are depicted in (a). The best path is marked by the bold, gray nodes in (b).

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. The merging rule is applied by combining all the local classifiers along the paths that converge to a leaf node, as if it was a single path. Although theoretically there could be an exponential number of paths in a DAG, this is very rare in real hierarchies, and it has not been a problem in all the datasets that we used in the experiments.

Fig. 3 depicts the application of the merging rule, first the probabilities and weights are computed and then combined in a score; the path with the best score is returned as the final prediction.

The complexity of the classification process (Equation (5)) depends on the length of the paths (*lengthPath*), the number of leaf nodes (*numLeafs*) and the number of paths (*numPaths*).

$$O(|L| \times lenghtPath \times numLeafs + numPaths) \tag{5}$$

In the worst case, the number of paths in a DAG is exponential, so this will make the complexity exponential. However, in real hierarchies, this is very unlikely.

4. Experimental setup

This section outlines the framework used for the experiments.

4.1. Datasets

Twenty datasets were used in the tests. Eighteen of these datasets are from the field of functional genomics.[1] The last two datasets are from the field of image classification.[2]

From the genomics datasets, there are 10 with a tree-structured hierarchy named FunCat (FUN) [19] and there are 8 with a DAG structured hierarchy named Gene Ontology (GO) [1]. Since the labels of the instances in the genomic datasets include paths to different nodes, the labels have been trimmed to get instances with paths to a single node. Only the nodes with more than 50 instances for DAG and 70 instances for tree hierarchies were considered. The datasets, after pruning, are described in Table 1. The complete hierarchy for one of the Gene Ontology datasets, the Cellcycle GO hierarchy, is depicted in Fig. 5 in Appendix A.

There are two image datasets, Caltech-101 [11] and Caltech-256 [13], both contain instances where the most specific labels in the instances are part of the leaf nodes. Caltech-101 hierarchy has 101 leaf nodes and Caltech-256 has 256 leaf nodes. The hierarchy of Caltech-256 is supposed to be a superset of the Caltech-101 hierarchy, but 101 has labels that 256 does not contain; for this reason we added to the 256 hierarchy the labels which fitted clearly in one branch and the rest were deleted.[3]

To obtain the attributes from the images in both Caltech datasets we used the vl-feat library from Matlab [26], using the basic recognition sample application: *phow_caltech101.m*[4] that obtains phow features, which are dense SIFT applied at

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

[2] http://www.vision.caltech.edu/archive.html.

[3] Deleted labels: face, anchor, barrel, cellphone, chair, cougar_face, crocodile_head, dollar_bill, flamingo_head, gramophone, lamp, metronome, stapler, stop_sign, wheelchair, windsor_chair and ying_yang.

[4] http://www.vlfeat.org/applications/apps.html.

Please cite this article in press as: M. Ramírez-Corona et al., Hierarchical multilabel classification based on path evaluation, International Journal of Approximate Reasoning (2015), http://dx.doi.org/10.1016/j.ijar.2015.07.008

Table 1
Description of the datasets for the experiments. $|L|$ = Number of Labels, LC = Label Cardinality (average number of labels relevant to each instance), A = Number of attributes, N = Number of instances and D = Maximum depth.

Dataset	$\|L\|$	LC	A	N	D
(a) MLNP datasets					
caltech-101	126	3.80	12 000	2520	5
caltech-256	319	3.82	12 000	7680	6
cellcycle_FUN	36	2.47	78	2339	4
church_FUN	36	2.45	28	2340	4
derisi_FUN	37	2.48	64	2381	4
eisen_FUN	25	2.26	80	1681	3
expr_FUN	36	2.46	552	2346	4
gasch1_FUN	36	2.47	174	2356	4
gasch2_FUN	36	2.46	53	2356	4
pheno_FUN	17	1.94	70	1162	3
seq_FUN	39	2.43	479	2466	4
spo_FUN	36	2.47	81	2302	4
cellcycle_GO	53	4.28	78	1708	11
church_GO	53	4.28	28	1711	11
derisi_GO	54	4.46	64	1746	11
expr_GO	53	4.34	552	1720	11
gasch1_GO	53	4.33	174	1716	11
gasch2_GO	53	4.37	53	1720	11
seq_GO	52	4.26	479	1711	11
spo_GO	53	4.31	81	1685	11
(b) NMLNP datasets					
cellcycle_FUN	49	2.45	78	3602	4
church_FUN	49	2.45	28	3603	4
derisi_FUN	49	2.44	64	3675	4
eisen_FUN	35	2.26	80	2335	4
expr_FUN	49	2.44	552	3624	4
gasch1_FUN	49	2.45	174	3611	4
gasch2_FUN	49	2.48	53	3624	4
pheno_FUN	22	1.96	70	1462	3
seq_FUN	51	2.40	479	3765	4
spo_FUN	49	2.45	81	3553	4
cellcycle_GO	56	3.38	78	3516	11
church_GO	56	3.37	28	3515	11
derisi_GO	57	3.41	64	3485	11
expr_GO	56	3.39	552	3537	11
gasch1_GO	56	3.38	174	3524	11
gasch2_GO	56	3.44	53	3537	11
seq_GO	59	3.45	479	3659	11
spo_GO	56	3.39	81	3466	11

several resolutions. Then it creates a word dictionary and finally generates spatial histograms. *phow_caltech101.m* selects 30 images per class to include in each dataset for training.

There are two versions of the genomic datasets, one in which all the instances reach a leaf node in the hierarchy and other in which they do not. To obtain those datasets, the instances that did not reached a leaf node were deleted and the nodes with less than 50 instances were deleted; this is the reason why genomic NMLNP datasets have less labels and in some cases the hierarchy has less levels.

The results in the experiments for the genomic datasets were obtained by applying a stratified ten-fold cross-validation. For the image datasets, the results were obtained by selecting k training instances per image and testing with the rest, repeating this process ten times. For Caltech-101, $k = 10$ and for Caltech-256, $k = 5$. This decision was taken due to time and resources restrictions; since these datasets have many instances and classes, and many more attributes. Also, most of the works regarding these datasets report results depending on the number of training images per class.

4.2. Evaluation measures

In the case of HMC the definition of the evaluation measures is not straightforward since the predictions can be partially correct, for that reason specific measures for multi-label classification [18] and HMC have been proposed [15,8]. We present the most common ones.

N is the number of instances in the training set. The labels of an instance are represented in a vector of size $|L|$ where the predicted/real labels are marked with 1 and the rest with 0. Y_i represents the real set of labels and $\hat{Y}_i$ the predicted set of labels.

Please cite this article in press as: M. Ramírez-Corona et al., Hierarchical multilabel classification based on path evaluation, International Journal of Approximate Reasoning (2015), http://dx.doi.org/10.1016/j.ijar.2015.07.008

Exact-Match: Eq. (6) shows the effectiveness of the method to return complete correct paths.

$$ExactMatch = \frac{1}{N}\sum_{i=1}^{N} 1_{Y_i=\hat{Y}_i} \tag{6}$$

Accuracy [12]: Eq. (7) is the ratio of the size of the intersection and union of the predicted and actual label sets.

$$Accuracy = \frac{1}{N}\sum_{i=1}^{N} \frac{\left|Y_i \cap \hat{Y}_i\right|}{\left|Y_i \cup \hat{Y}_i\right|} \tag{7}$$

This measure is less harsh than exact match because it considers the correctly classified labels but also ignores the fact that the classification difficulty tends to increase for deeper levels of the class.

Hamming-Loss and Hamming-Accuracy [6]: Eq. (8) evaluates the frequency that an example-label pair is misclassified.

$$HammingLoss = \frac{1}{N|L|}\sum_{i=1}^{N} \left|Y_i \oplus \hat{Y}_i\right| \tag{8}$$

where $\oplus$ is the *exclusive or* operator. Hamming accuracy (H-Accuracy) is defined as: $H\text{-}Accuracy = 1 - HammingLoss$.

F1-measure [22]: For multi-label classification, is defined as the scalar F1-measure (Eq. (9)) but redefining precision and recall:

- *Precision*: The fraction of predicted labels which are actually part of the true set of labels $\frac{|z_i \wedge \hat{z}_i|}{|\hat{z}_i|}$; and
- *Recall*: The fraction of true labels which are also predicted $\frac{|z_i \wedge \hat{z}_i|}{|z_i|}$.

$$F1 = \frac{2 \times precision \times recall}{precision + recall} \tag{9}$$

We have specified a vector z instead of the y_i vector, because in the multi-label context there are several ways to average this measure, for this article we used two of them:

F1-macro D (Eq. (10)) is averaged by instances (macro-averaged); N vectors of $z_i \equiv y_i$.

$$\text{F1-macro D} = \frac{1}{N}\sum_{i=0}^{N} F1(z_i, \hat{z}_i) \tag{10}$$

F1-macro L (Eq. (11)) is averaged by label (macro-averaged); $|L|$ vectors of $z_i \equiv [y_i^1, \ldots, y_i^N]$.

$$\text{F1-macro L} = \frac{1}{|L|}\sum_{i=0}^{|L|} F1(z_i, \hat{z}_i) \tag{11}$$

Precision and recall measures do take into account the hierarchical relationships between classes, since the information from the parents is used by the classifiers. The F1-macro L measure is a per class mean performance measure, the F1-macro D measure is a per instance mean performance measure.

Gain-Loose Balance: In addition to the previous measures, we propose a new measure classed **Gain-Loose Balance**. It is proposed because we noticed that previous measures gave the best score to a pruning method that predicts just the root node for every instance. Gain-Loose Balance (GLB) avoids this bias toward conservative predictions; it determines the rewards and penalties using the number of siblings of the node and the depth of the node in the hierarchy.

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.

Please cite this article in press as: M. Ramírez-Corona et al., Hierarchical multilabel classification based on path evaluation, International Journal of Approximate Reasoning (2015), http://dx.doi.org/10.1016/j.ijar.2015.07.008

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.

Equation (12) 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_i represents the number of siblings plus one of the node i that is being evaluated, and w_i is the weight of the node (see Equation (3)). The first term represents the gains and the second the looses. There are two kinds of losses, false positives and false negatives.

$$GLB = \frac{\sum_{i=0}^{n_p}(1-\frac{1}{N_i})(1-w_i)}{\sum_{i=0}^{n_t}(1-\frac{1}{N_i})(1-w_i)} - \left(\sum_{i=0}^{n_{fp}}\frac{1}{N_i}w_i + \sum_{i=0}^{n_{fn}}\frac{1}{N_i}w_i\right) \tag{12}$$

Gain-Loose Balance ranges from a maximum of $max = 1$ (when the predicted path is equal to the real path) to a minimum of $min = -\frac{max\,L}{2}$ with $N = 2$:

$$min = -2\sum_{i=1}^{max\,L}\frac{1}{N}w_i = -2\sum_{i=1}^{max\,L}\frac{1}{2}\left(1-\frac{i}{max\,L+1}\right) \tag{13}$$

$$= -2\left(\sum_{i=1}^{max\,L}\frac{1}{2} - \frac{1}{2(max\,L+1)}\sum_{i=1}^{max\,L} i\right) \tag{14}$$

$$= -max\,L + \frac{max\,L}{2} = -\frac{max\,L}{2} \tag{15}$$

where $max\,L$ is the maximum number of levels in the hierarchy. In the worst case scenario the node has just two siblings and $N = 2$. w_i is the weight of the node as defined in Equation (3).

As we know the maximum and minimum values of the GLB measure we transformed it into a (0, 1) range maintaining the ratio. Then the Normalized GLB (NGLB) is defined as:

$$NGLB = \frac{(GLB - min)}{max - min} \tag{16}$$

5. Experiments

We performed a set of experiments to: (i) compare different base classifiers and select one for the rest of the experiments; (ii) determine the effect of depth over the classification performance, comparing CPE against other methods for DAG hierarchies; (iii) compare the CPE method against the Flat and Multi-label approaches; (iv) test the usefulness of the pruning phase of CPE in MLNP datasets; and (v) compare the NMLNP version of our algorithm against other NMLNP methods.

To find statistical significance when comparing two methods, we used a one-tail Wilcoxon test; and when comparing more than two methods, we used a Friedman test using Tukey's honestly significant difference criterion; both with a confidence level of 95%. In all the experiments (expect for the first one) we used Random Forest as base classifier. All the evaluation measures are expressed in percentages.

5.1. Base classifier

An experiment was designed to analyze the effect of the base classifier in the performance of CPE, and to select the base classifier that best suits the datasets used to test our method. The experiment was performed over the 18 datasets using a ten-fold cross-validation scheme. We compared the performance of our method with four different hierarchical measures (see Section 4.2), using as base classifier five common methods:

1. Decision Tables (DT). Accuracy was the measure used to evaluate the attribute combinations which were selected by Best First method.
2. C4.5. The confidence factor used for pruning was 0.25. The minimum number of instances per leaf was set to 2.
3. Support Vector Machines (SVMs). The kernel used was a PolyKernel.
4. Naïve Bayes (NB).
5. Random Forest (RF). Generated 10 trees.

The results are depicted in Table 2. The best results are marked with bold letters.

Please cite this article in press as: M. Ramírez-Corona et al., Hierarchical multilabel classification based on path evaluation, International Journal of Approximate Reasoning (2015), http://dx.doi.org/10.1016/j.ijar.2015.07.008

Table 2
Performance of the different base classifiers along the evaluation metrics.

Base classifier	DT	C4.5	SVM	NB	RF
Accuracy	19.50	22.03	19.42	26.09	**30.69**
Exact Match	8.34	7.10	11.42	17.50	**20.26**
F1-macro D	25.68	30.08	24.52	31.48	**36.67**
F1-macro L	3.91	2.86	3.77	13.45	**14.02**

Discussion

We observe that Random Forest obtains the best performance in all the evaluation measures, with NB in second place. This could be because RF is able to deal with large, unbalanced data sets, and works well in the presence of noise, which is the case in most of these datasets. Thus, we used Random Forest as base classifier for the rest of the experiments.

5.2. Effect of the hierarchy's depth over classification performance

An experiment was set to determine the effect of the depth of the hierarchy in the classification performance of our method. We used MLNP datasets (Table 1(a)) to be sure that the evaluated instances were the same for each hierarchy depth and for this reason the pruning phase of CPE was not applied. We used DAG datasets for this experiment because they were the ones with deeper levels (up to 11). The DAG datasets were pruned from 11 to 2 levels maintaining the same instances, this means that the starting structures are DAGs but in shallower levels became trees due to the elimination of nodes.

We compared CPE against two methods that could handle DAG hierarchies:

1. Top-Down LCPN Corrected (TD-C). TD, proposed by Koller et al. [14], is a top-down approach that trains an 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. TD-C is an extension of TD for DAG hierarchies. When a leaf node is reached TD-C appends all the paths to that node to the final prediction. Random Forest was used as base classifier.
2. HIROM, proposed by Bi et al. [3], is also a local method that combines the local predictions to search for the optimal consistent multi-label classification using a greedy strategy. Using Bayesian decision theory, it selects the optimal subset of classes in the hierarchy by minimizing the conditional risk. The used base classifier was Support Vector Machines as reported by the authors. This is a recent work that can handle DAG hierarchies.

The results are depicted in Fig. 4. Each plot represents a different evaluation measure performance (y-axis) and the number of levels (x-axis); the lines represent the performance along the different classification methods averaged along all the datasets.

When there are less levels, there is also a decrease in the number of labels and, in general, it is easier to classify. The performance of all the methods decreases at a lower rate in the deeper hierarchies because less nodes/labels are pruned in the deeper levels.

Discussion

The classification performance of all the methods in accuracy, exact match, F1-macro D and F1-macro L metrics drop off quickly on the first two or three levels, then it steadily falls up to the 4th or 5th levels and then it stabilizes. We expected Hamming accuracy to behave as the other metrics but it increased with the number of levels. Hamming accuracy comes from the Hamming loss function (Eq. (8)) that evaluates how many times a set of labels for an example is misclassified normalized by the number of labels. The growth in the measure when there are more levels can be due to this normalization, while there are more labels an error cost less than an error when there are fewer labels.

Other methods perform similar to CPE with shallow hierarchies, but CPE stabilizes sooner when the number of levels increases which allows our method to outperform the rest of them. F1-macro L is the only measure where CPE does not obtain good results; this can be caused by some labels with few associated examples that are not well predicted.

From this analysis we consider that our method works well with deep hierarchies, since its performance is less affected when there are many levels in the hierarchy.

5.3. Comparison against flat and multi-label classification methods

In this experiment we contrasted our approach with two alternative classification schemes that do not consider explicitly the hierarchy:

1. *Flat*: A multi-class classifier that predicts only the leaf nodes of the hierarchy.
2. *Chained Classifiers (CC)*: A multi-label classifier that builds a classifier for each label; in the prediction phase it incorporates the results of previous classifiers (in a chain) as additional attributes, and then selects a subset of labels as the output (this inspired our use of the parent label as an additional attribute) [18].

The results are averaged along all the three sets of databases and reported in percentages for five evaluation measures. The results for the MLNP data sets are summarized in Table 3.

Please cite this article in press as: M. Ramírez-Corona et al., Hierarchical multilabel classification based on path evaluation, International Journal of Approximate Reasoning (2015), http://dx.doi.org/10.1016/j.ijar.2015.07.008

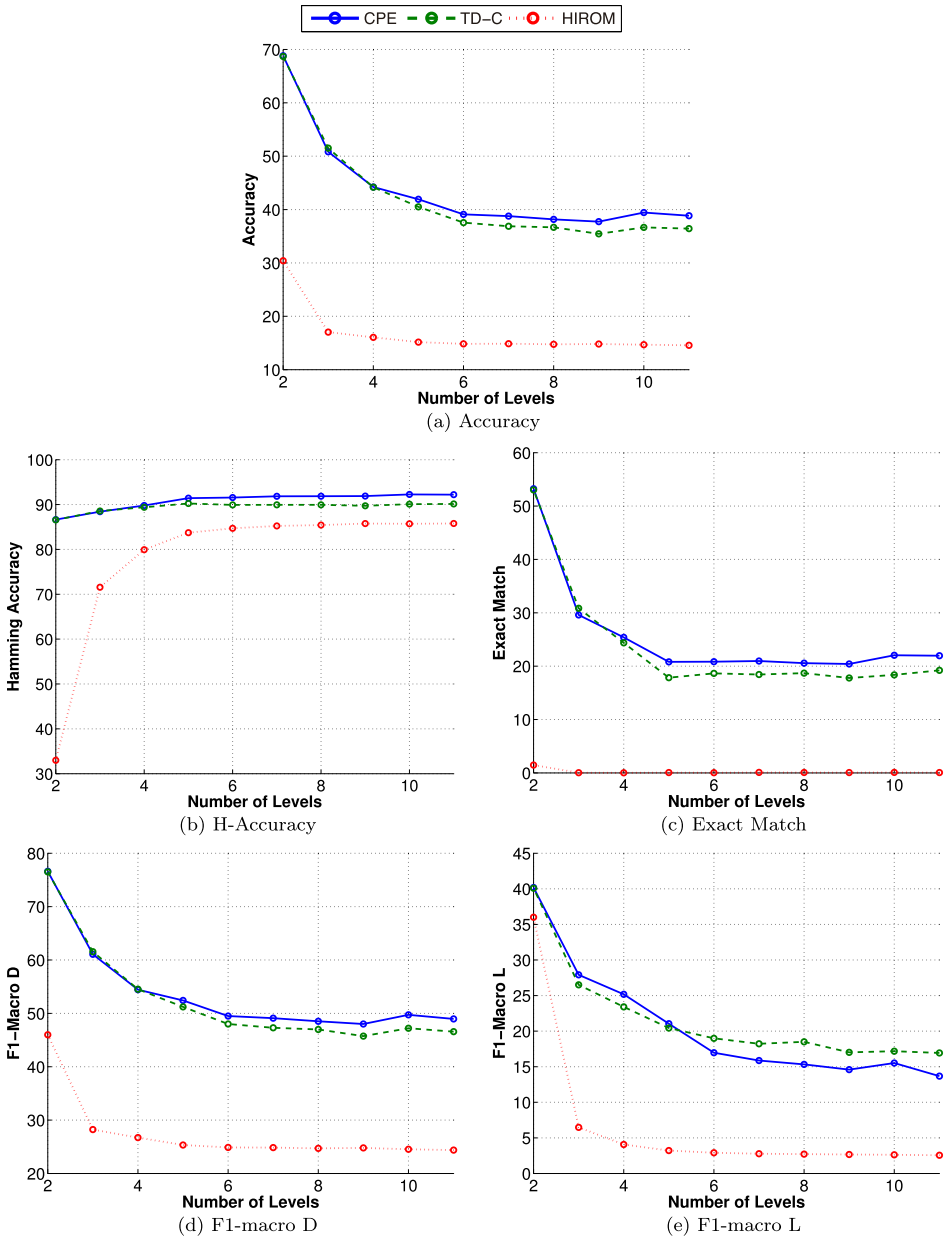


Fig. 4. Plots comparing the performance of CPE, TD-C and HIROM with hierarchies of different depths, using different metrics.

Discussion

CPE is competitive against the Flat approach in shallow not very populated hierarchies (FUN hierarchy has 4 levels and approximately 15 leaf nodes) although the Flat approach is superior in most of the metrics and has some significantly superior results. Exceptions are H-Accuracy and Exact Match, these measures take into account the errors in the predictions. This means that CPE is pruning when there is not a high prediction value and Flat is returning complete paths that may be incorrect.

In the case of deep not very populated hierarchies (GO datasets hierarchy has 11 levels and approximately 24 leaf nodes) in most cases CPE obtains better results than the Flat approach.

The small difference in the performance of CPE and the Flat approach in the genomic datasets is due to the fact that in presence of few leaf nodes the Flat approach tends to return good results, however, when there are more labels it becomes difficult to the Flat approach to differentiate between them with a single classifier, as can be seen in the results of the Caltech datasets.

The Caltech datasets have 256 and 84 leaf nodes, in every measure CPE surpasses the Flat approach with statistical significance. It can be seen that CPE is clearly superior to the Flat approach for large hierarchies.

Please cite this article in press as: M. Ramírez-Corona et al., Hierarchical multilabel classification based on path evaluation, International Journal of Approximate Reasoning (2015), http://dx.doi.org/10.1016/j.ijar.2015.07.008

Table 3
Comparison of CPE against the Flat and CC multi-label approaches averaged along the MLNP datasets (mean [std]). The best results are marked in bold. The number of datasets with significantly different results among the different datasets is marked in the super-index.

Evaluation measure	CPE	Flat	CC
(a) Average of the 10 FUN datasets			
Accuracy	23.22 [5.43]	**24.35 [4.67]**^{1}	15.1 [4.52]^{8}
H-Accuracy	90.11 [2.23]	89.4 [2.35]^{1}	**91.81 [2.96]**^{4}
Exact Match	**17.69 [4.67]**	17.60 [4.88]^{1}	7.92 [2.45]^{8}
F1-macro D	25.94 [6.05]	**27.99 [4.46]**^{1}	17.93 [6]^{7}
F1-macro L	13.47 [4.34]	**14.76 [4.08]**^{1}	9.94 [2.56]^{6}
(b) Average of the 8 GO datasets			
Accuracy	**38.79 [5.76]**	37.93 [5.22]	38.40 [3.01]^{5}
H-Accuracy	92.19 [0.64]	91.01 [0.86]	**93.64 [0.91]**
Exact Match	**21.98 [6.81]**	21.97 [5.79]	9.83 [2.53]^{8}
F1-macro D	48.96 [4.94]	47.64 [4.66]	**51.59 [3.34]**^{4}
F1-macro L	13.95 [5.69]	**16.95 [6.73]**^{2}	9.83 [3.88]^{5}

Evaluation measure	CPE	Flat
(c) Average of the 2 Caltech datasets		
Accuracy	**26.67 [10.61]**^{2}	18.36 [4.09]
H-Accuracy	**97.41 [1.27]**^{2}	48.71 [66.44]
Exact Match	**10.02 [10.03]**^{2}	6.19 [6.07]
F1-macro D	**34.96 [9.68]**^{2}	25.55 [2.13]
F1-macro L	**12.63 [11.72]**^{2}	7.43 [7.16]

Table 4
Comparing CPE-NMLNP against CPE-MLNP using MLNP datasets (mean [std]). The best results are marked in bold letter. The number of datasets with significantly superior results among the different datasets is marked in the super-index.

Evaluation measure	Non-pruned	Pruned
(a) Average of the 10 FUN datasets		
Accuracy	**23.22 [5.43]**	22.68 [6.27]
H-Accuracy	**90.11 [2.23]**	90.00 [2.27]
Exact Match	**17.69 [4.67]**	17.20 [4.95]
F1-macro D	**25.94 [6.05]**	25.40 [6.75]
F1-macro L	13.47 [4.34]^{1}	**13.51 [4.66]**
NGLB	**61.69 [2.66]**	61.33 [2.84]
(b) Average of the 8 GO datasets		
Accuracy	38.79 [5.76]	**39.37 [5.86]**
H-Accuracy	92.19 [0.64]	**92.24 [0.64]**
Exact Match	21.98 [6.81]	**22.53 [6.73]**
F1-macro D	48.96 [4.94]	**49.34 [5.01]**
F1-macro L	13.95 [5.69]	**14.00 [4.8]**
NGLB	78.23 [2.03]	**78.38 [1.98]**
(c) Average of the 2 Caltech datasets		
Accuracy	26.67 [10.61]	**26.73 [10.43]**
H-Accuracy	**97.41 [1.27]**	97.40 [1.27]
Exact Match	**10.02 [10.03]**	**10.02 [9.69]**^{1}
F1-macro D	34.96 [9.68]	**35.00 [9.46]**
F1-macro L	12.63 [11.72]	**12.75 [11.58]**^{1}
NGLB	32.81 [10.85]	**32.87 [10.59]**

CPE is also very competitive against the multi-label chain classifier. They both have fairly similar results in all measures except for Exact Match where CPE is clearly superior to CC.

Given the computational requirements of CC, it was not feasible to evaluate it with the Caltech datasets, although we expect a similar effect as with the Flat approach, the performance will decrease and CPE will be superior.

We conclude that taking in consideration the hierarchy, such as CPE does, has advantages with large hierarchies with many leaf nodes.

5.4. CPE-NMLNP vs CPE-MLNP

We also tested the performance of our method with and without our pruning method in datasets where the most specific label is in the leaf nodes of the hierarchy (MLNP datasets). Results are summarized in Table 4.

Discussion

The version of CPE that carries out the pruning phase and the version without pruning obtain almost the same performance when classifying MLNP datasets. This suggests that our method is pruning less nodes when the datasets are MLNP

Please cite this article in press as: M. Ramírez-Corona et al., Hierarchical multilabel classification based on path evaluation, International Journal of Approximate Reasoning (2015), http://dx.doi.org/10.1016/j.ijar.2015.07.008

Table 5
Comparing CPE against other NMLNP methods (mean [std]) for the genomic tree-structured datasets. The best results are marked in bold letter. The number of datasets with significantly different results among the different datasets is marked in the super-index.

Evaluation measure	CPE	HIROM	TPR	wTPR
Accuracy	**17.10 [4.7]**	3.11 [0.97]^{10}	12.70 [3]^{4}	12.85 [3.15]^{4}
H-Accuracy	**91.86 [2.12]**	87.93 [3.33]^{10}	88.58 [2.02]^{6}	91.83 [2.62]
Exact Match	**11.16 [3.3]**	2.07 [0.91]^{9}	5.18 [2.25]^{10}	6.27 [2.57]^{2}
F1-macro D	**19.87 [5.31]**	3.56 [0.99]^{10}	15.97 [3.24]	15.52 [3.29]^{7}
F1-macro L	11.51 [3.96]	9.48 [28.3]^{9}	11.87 [2.96]	**11.64 [3.4]**
NGLB	**59.41 [1.75]**	40.97 [2.41]^{10}	51.23 [3.62]^{10}	57.23 [2.47]

Table 6
Comparing CPE against other NMLNP methods (mean [std]) for the image classification tree-structured datasets. The best results are marked in bold letter. The number of datasets with significantly different results from the total datasets compared with CPE is marked in super-index.

Evaluation measure	CPE	HIROM	TPR	wTPR
Accuracy	**26.70 [10.36]**	12.63 [8.29]^{1}	22.79 [0.67]^{2}	25.52 [4.99]
H-Accuracy	97.40 [1.28]	95.86 [0.62]	95.94 [3.95]^{1}	**98.12 [1.15]**^{1}
Exact Match	**10.00 [9.65]**	0.04 [0.06]^{2}	0.95 [0.93]^{1}	1.87 [2.45]^{1}
F1-macro D	34.95 [9.44]	19.87 [13.18]^{1}	33.25 [0.43]^{2}	**35.92 [3.71]**^{1}
F1-macro L	**12.82 [11.63]**	0.79 [0.16]^{2}	8.71 [9.02]	6.87 [7.94]^{2}
NGLB	32.82 [10.56]	12.77 [7.09]^{2}	**34.34 [15.2]**	25.48 [7.5]^{1}

Table 7
Comparing CPE against other NMLNP methods (mean [std]) in 8 genomic DAG datasets. The best results are marked in bold letter. The number of datasets with significantly superior results from the total datasets is marked in super-index.

Evaluation measure	CPE	HIROM
Accuracy	**31.90 [3.45]**^{8}	14.89 [0.45]
H-Accuracy	**93.09 [0.43]**^{8}	84.93 [0.26]
Exact Match	**8.55 [3.31]**^{8}	0.00 [0.00]
F1-macro D	**44.40 [3.18]**^{8}	24.79 [0.58]
F1-macro L	**8.49 [2.41]**^{8}	3.32 [0.2]
NGLB	**80.16 [1.24]**^{8}	64.05 [0.38]

which means it can be used for both MLNP and NMLNP because it has about the same performance. The pruning is even avoiding some errors specially in deep (GO) and very populated (Caltech) structures where it obtains a slight (significant in few measures and datasets) advantage.

5.5. Comparison against other NMLNP methods

The proposed method, CPE-NMLNP, was evaluated experimentally with a number of tree and DAG structured hierarchies and compared with various hierarchical classification techniques. The results were obtained by a stratified 10-fold cross-validation.

Tree structured datasets. We used twelve hierarchical datasets and compared CPE against three HMC-NMLNP methods:

1. HIROM [3] (see Section 5.2).
2. True Path Rule (TPR) [23]. This is also a local approach the incorporates inconsistency correction based on the output of all classifiers. The positive decisions for a node influence the decisions of the parents (bottom-up) turning them positive, and the negative predictions of a node turn all its descendants negative. In the experiments we use a threshold of 0.5.
3. Weighted True Path Rule (wTPR) [23]. It extends TPR by incorporating a weight to the nodes in the hierarchy, depending on the positive and negative predicted children of the node. In the experiments we use a threshold of 0.5 and a parent weight $w_p = 0.5$.

Table 5 summarizes the results with the genomic (FUN) datasets, and Table 6 with the image classification (Caltech) datasets. In these tables we compare more than two methods, the super-indexes represent the number of datasets with significantly different results (instead of the number of datasets with significantly superior results, Table 4).

DAG structured datasets. We used eight hierarchical datasets and compared CPE against only one HMC-NMLNP method, HIROM [3], as the other approaches such as TPR cannot deal with DAG hierarchies. The results are depicted in Table 7. As in the previous two tables, the super-indexes represent the number of datasets with significantly different results.

Discussion

In shallow hierarchies (FUN, Caltech) the best results are obtained in Exact Match since CPE outperformed every method by several percentile points and in many cases the differences are significant; this means CPE returned more correct com-

Please cite this article in press as: M. Ramírez-Corona et al., Hierarchical multilabel classification based on path evaluation, International Journal of Approximate Reasoning (2015), http://dx.doi.org/10.1016/j.ijar.2015.07.008

plete paths. Compared with HIROM, CPE obtained statistically significant better results in every measure. Compared with TPR, CPE obtained less errors because in H-Accuracy and NGLB in many cases the differences are significant. Compared with wTPR, CPE obtained better F1-macro D and worst F1-macro L, which means CPE is obtaining more instances well predicted but wTPR gets better results along all classes.

Regarding DAG structured datasets, CPE always obtained the best results with statistical significance when compared against HIROM.

6. Conclusions and future work

We presented a novel approach, CPE, for hierarchical classification that can handle both, tree and DAG structures, and which performs non-mandatory leaf node prediction. CPE selects the best path in the hierarchy based on a score, incorporating information from the hierarchy by including the parent label, as in chain classifiers. The path score combines the predictions of the local classifiers, which depend on the relations of the nodes and the positions in the hierarchy. A pruning phase is included to eliminate less probable branches based on the probability of predicting the children of a node. Also, a new metric was introduced that avoids conservative classifications in the case of NMLNP. Experiments with 20 tree and DAG hierarchies showed that: (i) CPE works better in deep, populated hierarchies, (ii) CPE is better than the Flat approach when there are many leaf nodes, (iii) CPE is useful for MLNP and NMLNP datasets, and (iv) CPE is competitive when compared against other state-of-the-art methods for shallow hierarchies and superior for DAGs.

As future work, we plan to add an attribute selection mechanism depending on the node so that each local classifier can adapt to the characteristics to the level in the hierarchy, e.g., when classifying animals in the first hierarchical levels, the shape is important to differentiate, for instance, between a duck and a dog, but in deeper levels where the task is to differentiate, for instance, breads of dogs, local characteristics could be more useful. Also we would like to include information about the number of children to classify in the weight of each node, and extend our method for multiple path prediction.

Appendix A. A complete dataset hierarchy

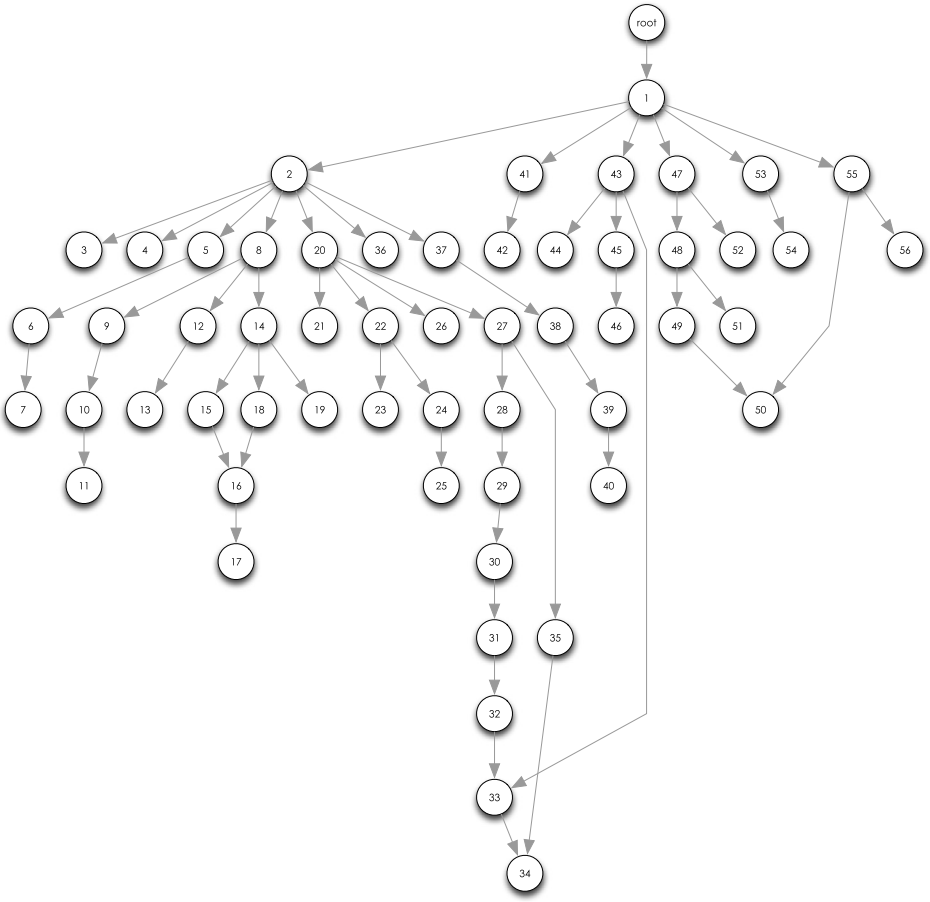


Fig. 5. Hierarchy of the cellcycle_GO dataset hierarchy.

Please cite this article in press as: M. Ramírez-Corona et al., Hierarchical multilabel classification based on path evaluation, International Journal of Approximate Reasoning (2015), http://dx.doi.org/10.1016/j.ijar.2015.07.008

References

[1] M. Ashburner, C.A. Ball, J.A. Blake, Gene ontology: tool for the unification of biology, Nat. Genet. 25 (2000) 25–29.
[2] W. Bi, J.T. Kwok, Multi-label classification on tree-and DAG-structured hierarchies, in: 28th International Conference on Machine Learning (ICML-11), 2011, pp. 17–24.
[3] W. Bi, J.T. Kwok, Hierarchical multilabel classification with minimum Bayes risk, in: Data Mining (ICDM), 2012 IEEE 12th International Conference, December 2012.
[4] H. Blockeel, M. Bruynooghe, Hierarchical multi-classification, in: ACM SIGKDD 2002 Workshop on Multi-Relational Data Mining (MRDM 2002), 2002, pp. 21–35.
[5] H. Blockeel, L. Schietgat, J. Struyf, S. Džeroski, A. Clare, Decision Trees for Hierarchical Multilabel Classification: A Case Study in Functional Genomics, Springer, Berlin, Heidelberg, 2006.
[6] W. Cheng, E. Hüllermeier, K.J. Dembczynski, Bayes optimal multilabel classification via probabilistic classifier chains, in: 27th International Conference on Machine Learning (ICML-10), 2010, pp. 279–286.
[7] A. Clare, R.D. King, Predicting gene function in Saccharomyces cerevisiae, Bioinformatics (2003) ii42–ii49.
[8] E.P. Costa, A.C. Lorena, A.C.P.L.F. Carvalho, A.A. Freitas, A review of performance evaluation measures for hierarchical classifiers, in: Evaluation Methods for Machine Learning II: Papers from the AAAI-2007 Workshop, 2007, pp. 1–6.
[9] E.P. Costa, A.C. Lorena, A.C.P.L.F. Carvalho, A.A. Freitas, N. Holden, Comparing several approaches for hierarchical classification of proteins with decision trees, in: Advances in Bioinformatics and Computational Biology, Springer, 2007, pp. 126–137.
[10] I. Dimitrovski, D. Kocev, S. Loskovska, S. Džeroski, Detection of visual concepts and annotation of images using ensembles of trees for hierarchical multi-label classification, in: Recognizing Patterns in Signals, Speech, Images and Videos, Springer, 2010, pp. 152–161.
[11] Li Fei-Fei, R. Fergus, P. Perona, Learning generative visual models from few training examples: an incremental bayesian approach tested on 101 object categories, Comput. Vis. Image Underst. 106 (1) (2007) 59–70.
[12] S. Godbole, S. Sarawagi, Discriminative methods for multi-labeled classification, in: Advances in Knowledge Discovery and Data Mining, Springer, 2004, pp. 22–30.
[13] Gregory Griffin, Alex Holub, Pietro Perona, Caltech-256 object category dataset, 2007.
[14] D. Koller, M. Sahami, Hierarchically classifying documents using very few words, Stanford InfoLab, 1997.
[15] A. Kosmopoulos, I. Partalas, Evaluation measures for hierarchical classification: a unified view and novel approaches, arXiv preprint, 2013.
[16] M. Ramírez, L.E. Sucar, E.F. Morales, Multi-label classification for tree and directed acyclic graphs hierarchies, in: Probabilistic Graphical Models, 2014, pp. 409–425.
[17] J. Read, B. Pfahringer, G. Holmes, E. Frank, Classifier chains for multi-label classification, Mach. Learn. (2011) 254–269.
[18] Jesse Read, Scalable multi-label classification, PhD thesis, University of Waikato, 2010.
[19] 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, Nucleic Acids Res. 32 (18) (2004) 5539–5545.
[20] C.N. Silla Jr., A.A. Freitas, A survey of hierarchical classification across different application domains, Data Min. Knowl. Discov. 22 (April 2010) 31–72, http://link.springer.com/10.1007/s10618-010-0175-9.
[21] L.E. Sucar, C. Bielza, E.F. Morales, P. Hernandez-Leal, J.H. Zaragoza, P. Larrañaga, Multi-label classification with bayesian network-based chain classifiers, Pattern Recognit. Lett. 41 (2014) 14–22.
[22] G. Tsoumakas, I. Katakis, Multi-label classification: an overview, Int. J. Data Warehous. Min. 3 (2007) 1–13, ISSN 15483924.
[23] G. Valentini, True Path Rule Hierarchical Ensembles, Multiple Classifier Systems, vol. 5519, Springer, 2009, pp. 232–241.
[24] G. Valentini, N. Cesa-Bianchi, HCGene: a software tool to support the hierarchical classification of genes, Bioinformatics 24 (5) (2008) 729–731.
[25] G. Valentini, M. Re, Weighted true path rule: a multilabel hierarchical algorithm for gene function prediction, in: European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases: 1st International Workshop on Learning from Multi-Label Data, 2009, pp. 132–145.
[26] A. Vedaldi, B. Fulkerson, VLFeat: an open and portable library of computer vision algorithms, http://www.vlfeat.org/, 2008.
[27] C. Vens, J. Struyf, L. Schietgat, S. Džeroski, H. Blockeel, Decision trees for hierarchical multi-label classification, Mach. Learn. 73 (2) (2008) 185–214.
[28] J.H. Zaragoza, L.E. Sucar, E.F. Morales, C. Bielza, P. Larra, Bayesian chain classifiers for multidimensional classification, in: Twenty-Second International Joint Conference on Artificial Intelligence, IJCAI 11, 2011.
[29] M.L. Zhang, Z.H. Zhou, A review on multi-label learning algorithms, IEEE Trans. Knowl. Data Eng. 26 (8) (2014) 1819–1837.

Please cite this article in press as: M. Ramírez-Corona et al., Hierarchical multilabel classification based on path evaluation, International Journal of Approximate Reasoning (2015), http://dx.doi.org/10.1016/j.ijar.2015.07.008