Hierarchical MultiClass AdaBoost

Charalampos Chelmis
Department of Computer Science
University at Albany, SUNY
Albany, New York, USA
cchelmis@albany.edu

Wenting Qi
Department of Computer Science
University at Albany, SUNY
Albany, New York, USA
wqi@albany.edu

***Abstract*—One of the most challenging machine learning problems is a particular case of classification in which classes are hierarchically structured and data instances can be assigned multiple labels residing in a path of the hierarchy. In this paper, we propose hierarchy–aware multiclass AdaBoost, allowing for the first time weak classifiers in an ensemble learning setting to be trained for hierarchical multiclass classification while incorporating a hierarchy–aware loss function directly into the training process. Experimental results on numerous real–world datasets show that, despite its simplicity, the proposed algorithm outperforms all baselines, arising as the state of the art in hierarchical multiclass classification.**

***Index Terms*—hierarchical classification, ensemble learning, learning with constrains**

I. Introduction

Ensemble learning is the process by which base classifiers are trained and combined to improve classification accuracy in challenging classification tasks [1]–[5]. In this work, we focus on Adaboost [6], one of the most influential supervised learning algorithms belonging to this branch of meta–learning methods. As a simple and interpretable modeling tool, Adaboost has been the method of choice when any–time solutions are required for classification tasks involving large data sets [7].

To date, numerous multi–class extensions of AdaBoost have been proposed for typical classification problems, where a data instance is associated with a single class from a set of disjoint classes [8]–[10]. Recently, however, the AI community has sifted its focus to more challenging learning probelms, in which classes are not disjoint but organized into a hierarchical structure [11]–[14]. Depending on the domain, the hierarchical structure may assume the form of a tree or a directed acyclic graph [15], and data instances are associated with several classes laying on a path in the class hierarchy. This task is known as hierarchical multiclass classification [14], and has numerous real–world applications, including, but not limited to, text classification [16]–[19], image annotation [13], and bioinformatics [11], [20], [21].

Early approaches treated the hierarchical multiclass classification problem as a sequence of several separate classification tasks, ignoring the hierarchical structure [22]. Subsequent methods can be divided into two main branches, namely (i) local, and (ii) global methods [14]. Local methods generate a hierarchy of classifiers in a top–down manner, in which each classifier is responsible for the prediction of either particular nodes or levels. On the other hand, global approaches focus on training a single classifier, capable of associating data instances with classes in the hierarchy as a whole. Both local and global methods have advantages and disadvantages. Specifically, local methods are more likely to capture dependencies among classes from the hierarchy, however, they suffer from high computational complexity due to the number of learners, and the inconsistency problem, i.e., an error early on is propagated down the hierarchy. On the other hand, global approaches are typically less computationally intensive than local approaches, however, they result in a more complex model, which is less likely to capture dependencies among classes from the hierarchy.

Contributions. Drawing inspiration from the success of AdaBoost and state–of–the–art hierarchical multiclass classification methods, we propose hierarchical multiclass Adaboost to train an ensemble of weak learners that learn to obtain consistent paths based on a hierarchy–aware loss function. Our main contributions can be summarized as follows:

- We present hierarchy–aware Hamming loss as a way of introducing hierarchical constraints into AdaBoost.MH [23].
- We propose a non–uniform weighting scheme that sets the weights of individual data instances asymmetrically to penalize upper–level misclassifications more heavily.
- We derive a new algorithm that directly extends AdaBoost.MH to the hierarchical multiclass case.
- We experimentally evaluate the effectiveness and explanatory power of our approach using real–world datasets. To ensure the reproducibility of our work, we make our source code available at https://github.com/IDIASLab/Hierarchical-Multiclass-AdaBoost.

II. Prior and Related Work

AdaBoost. Boosting was first proposed by [24] as a way of combining base classifiers to produce an accurate classification model. AdaBoost for binary classification, was introduced by [6], and was later popularized by [23]. Among the AdaBoost variants, Adaboost.SAMME directly extends AdaBoost to the multiclass case without reducing it to multiple two–class problems [25]. However, by assigning a single label out of

This material is based upon work supported by the National Science Foundation under Grant No. ECCS–1737443.

a set of classes to each data instance, Adaboost.SAMME is not readily applicable to hierarchical multiclass classification, where each data instance may be assigned multiple labels according to the hierarchy. We therefore focus on multiclass AdaBoost.MH [23], as it is explicitly designed to minimize the Hamming loss. This is instrumental in our formulation to introduce the hierarchical constraints into the learning process.

Hierarchical MultiClass Classification. Existing hierarchical multiclass classification methods can be categorized by (i) the type of hierarchical structure considered (i.e., tree or directed acyclic graph), (ii) the process by which the hierarchy is explored (i.e., local or global), (iii) how deep in the hierarchical structure classification is to be performed (i.e., always at a leaf node as opposed to any level) [14], (iv) how many paths can a data instance be associated with (i.e., exactly one as opposed to multiple paths, which is the case of multi–label classification [12], [14]), (v) the weak learner used to train local classifiers (e.g., Naive Bayes or SVM), and (vi) the loss function. Next, we summarize the most representative approaches from each category.

Top–down methods (e.g., [11]) start by constructing a tree of classifiers during training. In the binary case, a classifier is trained for each node; in the multiclass case, a multi–valued classifier is trained per parent node. During testing, a classifier labels an instance and passes the feature representation of the instance to the respective classifier one level lower in the hierarchy. The process is repeated until either a threshold (e.g., top k) or a leaf is reached. The key limitation of such methods is error–propagation, i.e., a misclassification at some node, results in an incorect prediction along the whole path from than node down to the corresponding leaf. *Global* approaches such as [26] are less likely to learn from the class dependencies, and often result in overfitted models. *Hybrid* schemes (e.g., [27]) train local classifiers in a top–down approach but then enforce consistency constraints (e.g., for a class to be set, it's parent node in the hierarchy must also be set) to obtain the set of labels during classification.

[18] introduced hierarchical loss, namely H–loss, as a performance measure for hierarchical classification problems, which was subsequently used to approximate the Bayes optimal classifier with respect to this metric [28]. Although H–loss emphasized on the importance of avoiding misclassifications at the upper hierarchical levels, it could lead to misleading predictions as not all misclassifications in the hierarchy are penalized. Specifically, H–loss only counts the first missclassification along a prediction path from the root to leaf nodes. To address this limitation, [29] introduced the hierarchical multilabel classification loss function (HMC–loss) and derived the optimal prediction rule by minimizing the condisional risk with respect to this loss. By tuning its parameters, HMC–loss can be reduced to Hamming loss, the most frequently used loss function in multilabel classification [30], or the path length before the first missclassification [31].

Our proposed approach can be considered to be a global method, where a base learner for all classes is trained at each iteration, and multiple learners are being trained over time to "learn" different aspects of a tree–structured hierarchy, and avoid overfitting. Classification is performed at the leaf nodes, while emphasizing the importance of predicting a consistent path with minimum loss.

III. Preliminaries

A. Problem Definition

For a description of the hierarchical multiclass classification problem, let $\mathcal{T}$ denote the tree–structured class hierarchy of m classes in total, where nodes $j \in \mathcal{T}$ are indexed as $0, 1, 2, \ldots, m-1$ in a top to bottom manner (i.e., 0 is for the root, 1 indicates its leftmost child, and so forth). We use $pa(j)$ to denote node j's unique parent, $anc(j)$ the set of its ancestors, and $sibl(j)$ its siblings. We further denote the number of j's ancestors as $|anc(j)|$, and that of its siblings as $|sibl(j)|$. Additionally, let the training data be $\mathcal{D} = \{(\mathbf{x}(i), \mathbf{y}(i)), \ldots, (\mathbf{x}(n), \mathbf{y}(n))\}$, where $\mathbf{x}(i) \in \mathbb{R}^d$ are feature vectors, $0 \leq i \leq n$, and label vectors $\mathbf{y}(i) = [y_0(i), y_1(i), y_2(i), \ldots, y_{m-1}(i)] \in \{-1, 1\}^m$ indicate the memberships of $\mathbf{x}(i)$ to each of the categories in $\mathcal{T}$. Then, $n \times d$ matrix $\mathbf{X}$ is the feature matrix, and $\mathbf{Y}$ is the $n \times m$ label matrix. Finally, we use $\hat{\mathbf{y}}(i)$ to represent the predicted multiclass vector of the $i-$th data instance, and t as the index of a weak classifier.

Given $\mathcal{D}$ and $\mathcal{T}$, the goal of learning is to infer a multiclass classification model Ω as a sum of T base classifiers, which can be used to predict the hierarchical categories of unseen data instances, with the additional constraint that the label of any non–root node j can be 1 if and only if all of its ancestors are labeled positive.

IV. Hierarchical MultiClass AdaBoost

We propose `H–Ada.MH`, a new algorithm for hierarchical multiclass classification that builds upon the Adaboost.MH framework [23] by (i) introducing hierarchical constraints and (ii) asymmetrically penalizing upper–level misclassifications more heavily. We discuss the technical details of `H--Ada.MH` in this section.

A. Hierarchy–aware Hamming Loss

Let $\mathcal{Y}$ denote the set of labels in the dataset, and $\mathbb{Y}(i) \subseteq \mathcal{Y}$ be the set of labels predicted for $\mathbf{x}(i)$. Then, the predicted multilabel $\hat{\mathbf{y}}(i) = [\hat{y}_1, \ldots, \hat{y}_{m-1}]$ is defined as:

$$\hat{y}_j = \begin{cases} 1, & \text{if } j \in \mathbb{Y}(i), \\ -1, & \text{if } j \notin \mathbb{Y}(i). \end{cases} \tag{1}$$

In the case of single–label prediction, the purpose is to find a hypothesis $H : \mathbf{X} \to \mathcal{Y}$, such that given a new data instance $(\mathbf{x}, \mathbf{y})$, the probability of $H(\mathbf{x}) \notin \mathcal{Y}$ is minimized. Note that in this case, $H(\mathbf{x})$ is a scalar measuring the probability that one label is predicted incorrectly. With the definition of the multilabel vector in Eq. (1), the multiclass classification problem can be defined as finding a model $H : \mathbf{X} \times 2^m$, such that the probability of predicted labels differing from the observed labels is minimized.

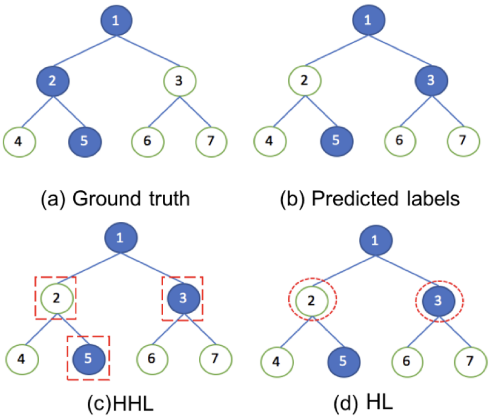

Fig. 1. Toy example illustrating the difference between Hamming loss (HL) and the proposed Hierarchy-aware Hamming loss (HHL). Positive nodes are highlighted in dark blue. Misclassified nodes are marked with red circles and red squares for HL and HHL respectively.

Suppose the data follows distribution D. Hamming loss can be used to measure the performance of a classifier as $\ell_{HL}(\mathbf{y}, H) = \frac{1}{m} E_{(\mathbf{x},\mathbf{y})\sim D}\left[|H(\mathbf{x}) \ominus \mathbf{y}|\right]$, where $\ominus$ denotes symmetric difference [23]. Intuitively, $HL(H)$ can be interpreted as the average loss of m binary classifiers, with $\mathbf{y}$ and $H(\mathbf{x})$ representing the m binary observed and predicted labels respectively, and a loss being recorded when $H(\mathbf{x})_j \neq \mathbf{y}_j$.

Our proposed Hierarchy–aware Hamming loss (HHL for short) incorporates hierarchy constrains as follows:

$$\ell_{HHL}(\mathbf{y}, \hat{\mathbf{y}}) = \sum I_p(y_j, \hat{y}_j), \tag{2}$$

where the summation is over all indices $0 \leq i \leq n$ and $j \in \mathcal{T}$,

$$I_p(y_j, \hat{y}_j) = \begin{cases} 0 & \text{if } y_j = \hat{y}_j \text{ and } \mathbf{y}_{anc(j)} = \hat{\mathbf{y}}_{anc(j)}, \\ 1 & \text{if } y_j \neq \hat{y}_j \text{ or } \mathbf{y}_{anc(j)} \neq \hat{\mathbf{y}}_{anc(j)}, \end{cases} \tag{3}$$

and any non–root node j can be labeled positive only if all of its ancestors are labeled positive [29], i.e.,

$$y_j = 1 \Rightarrow \mathbf{y}_{anc(j)} = 1. \tag{4}$$

For illustration purposes, Figure 1 shows an example in which node 5 is labeled correctly, as opposed to its parent node 2. Such prediction is not allowed when HHL is used, since the hierarchical constraint introduced in Eq. 4 is not met for node 5.

B. Hierarchical Weight Initialization

In Adaboost.MH, initial observation weights for each data i with respect to class j are set to $w(i,j) = \frac{1}{nm}$, thus equally weighting each of the m classes. When classes are however related by a tree–structured hierarchy $\mathcal{T}$, the the prediction for each class is no longer independent. Instead, class predictions for nodes laying at lower levels (i.e., more specific concepts) in $\mathcal{T}$ depend on the predictions for their ancestors (i.e., more generic concepts), according to the hierarchy constraint imposed by Eq. 4.

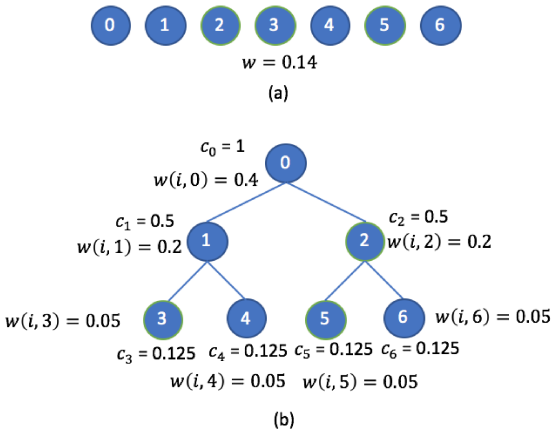

Fig. 2. Toy example illustrating weight initialization for (a) Adaboost.MH, and (b) Hierarchical Adaboost.MH.

To incorporate structural information in Adaboost.MH, and to penalize misclassifications at the upper hierarchy levels more heavily than those at the lower levels, we propose to set weights asymmetrically to

$$w_h(i,j) = \frac{c_j}{n \sum_{k=0}^{m-1} c_k}, \tag{5}$$

where

$$c_j = \begin{cases} 1, & j = 0 \\ \frac{c_{pa(j)}}{|sibl(j)|}, & j > 0, \end{cases} \tag{6}$$

and $c_{pa(j)}$ is the hierarchical weight of j's parent.

Figure 2 demonstrates the difference between Adaboost.MH's weighting scheme, and the proposed asymmetric weighting approach. For all classes, $n = 1$. Classes are equally weighted (i.e., $w = 0.14$) in the case of Adaboost.MH, as shown in Fig. 2(a). Instead, as Fig. 2(b) illustrates, weights decrease progressively by depth with the proposed asymetric weighting scheme.

C. Hierarchy–aware Multiclass Adaboost

Next, we describe a new algorithm, which facilitates the generalization of the traditional AdaBoost framework for hierarchical multiclass classification. We refer to our proposed algorithm as `H-Ada.MH` (c.f. Algorithm 1).

`H-Ada.MH` proceeds to train $T = \{t_0, t_1, \ldots, t_T\}$ weak learners as follows. Starting with a training sample, weighted by $w_0(i,j)$, the first base learner (e.g., random forest) t_0 is trained to produce a base hypothesis h_{t_0} for each class $j \in \mathcal{T}$ top to bottom, according to the tree structure. The predicted label for j is verified against its ancestor, i.e., $pa(j)$ in $\mathcal{T}$, and is set to -1 if Eq. 4 is not satisfied. In general, h_t can be obtained as:

$$h_t(\mathbf{x}(i), j) = \begin{cases} h_t(\mathbf{x}(i), j), & h_t(\mathbf{x}(i), pa(j)) = 1, \\ -1, & h_t(\mathbf{x}(i), pa(j)) = -1. \end{cases} \tag{7}$$

Algorithm 1 `H–Ada.MH`

Input: Dataset $\mathcal{D}$, Tree–structured hierarchy $\mathcal{T}$, Number of base learners T to be trained

Initialize observation weights $w_0(i,j)$ for each $\bar{y}_j(i)$ using Eq. (5);
repeat
  Train base learner t using weight $w_t(i,j)$;
  Obtain weak hypothesis $h_t(\boldsymbol{x(i)},j)$;
  if $pa(j) = NULL$ or $h_t(\boldsymbol{x(i)}, pa(j)) = 1$ **then**
    $h_t(\boldsymbol{x(i)},j) = h_t(\boldsymbol{x(i)},j)$;
  else
    $h_t(\boldsymbol{x(i)},j) = -1$;
  end if
  Compute base coefficient for learner t using Eq. (8);
  Update data weight $w_{t+1}(i,j)$ with Eq. (10);
until T base learners have been trained.
return final hypothesis using Eq. (12)

TABLE I
SUMMARY OF DATASETS. THE TOTAL NUMBER OF CLASSES, AND LEAF NODES, DEPTH OF TREE HIERARCHY, NUMBER OF FEATURES, AND AVERAGE NUMBER OF DATA INSTANCES PER LEAF NODE ARE REPRESENTED BY $|C|$, $|l|$, d, $|A|$, AND x_c RESPECTIVELY.

Dataset	Train	Test	$\lvert C\rvert/\lvert l\rvert$	d	$\lvert A\rvert$
ImageCLEF07D	10000	1006	22/9	3	80
ImageCLEF07A	10000	1006	26/11	3	80
Diatoms	2065	1054	343/258	2	371

After obtaining a base hypothesis for all training data instances, the base coefficient for learner t is computed as:

$$\alpha_t = \frac{1}{2} ln(\frac{1+r_t}{1-r_t}), \tag{8}$$

where

$$r_t = \sum_{i,j} w_t(i,j) y_j(i) h_t(\mathbf{x}(i), j), \tag{9}$$

and the weight $w_{t+1}(i,j)$ of that training data is boosted:

$$w_{t+1}(i,j) = \frac{w_t(i,j) exp(-\alpha_t y_j(i) h_t(\mathbf{x}(i), j)}{Z_t}, \tag{10}$$

where

$$Z_t = \sum_{i,j} w_t(i,j) \exp\left(-\alpha_t y_j(i) h_t(\mathbf{x}(i), j)\right). \tag{11}$$

The process is repeated until T base learners have been trained. The final hypothesis is obtained as follows:

$$H(\boldsymbol{x}, j) = sign(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}; j)). \tag{12}$$

V. EXPERIMENTS

A. Datasets

We conduct experiments on three real–world, freely available datasets, whose class hierarchy is structured as trees.

- **ImageCLEF07D and ImageCLEF07A** [13]: X–ray images extracted from the 2007 ImageCLEF competition.

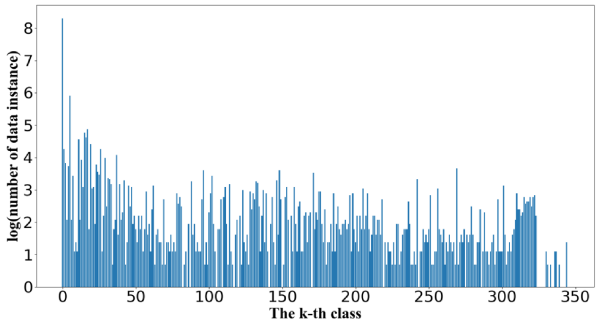

Fig. 3. Number of classes per data instance in the Diatoms dataset.

- **Diatoms** [16]: image dataset used to evaluate algorithms for automatic identification of diatoms based on shape and ornamentation.

Table I summarizes these datasets. The total number of classes is substantially larger in the Diatoms dataset as opposed to the other two datasets. Additionally, the average number of data instances per class in the Diatoms dataset (c.f. Fig. 3) is quite small (7.53 ± 7.245), making the integration of dependencies between classes critical for accurate classification.

B. Experimental Setup

Experiments are conducted on a commodity IOS laptop with a 3.1GHZ Intel core i7. CPU and 16GB 1867 MHZ DDR3 memory. For fair comparison, all baseline parameters are tuned to achieve the best performance. For the same reason, all datasets are pre–split into training and testing according to their original specifications. For both `H–Ada.MH` and Ada.MH, we use a one–layer decision tree as the base learner, and set the number of base learners, T, to 600 for the ImageCLEF07D and ImageCLEF07A datasets, and to 200 for the Diatoms dataset.

C. Baselines

To demonstrate the effectiveness of `H–Ada.MH`, we compare it with the following methods:

- **Adaboost.MH** [23]: AdaBoost extension for multiclass classification based on Hamming loss. Adaboost.MH ignores the hierarchical relationships among classes, treating a hierarchical multiclass classification problem as a flat multilabel classification task.
- **Local–level** [11]: A top–down method, according to which a classifier in each level predicts among classes belonging to that level. We use Random forests as the base classifier for each level.
- **Global** [26]: A global approach that builds a single classifier to discriminate between all categories simultaneously.

Note that recent **Deep Neural Network** architectures cannot be used as a baseline for two main reasons. First, publicly available datasets comprise a rather limited amount of training samples and number of features. Similarly, the classes are very sparse, making the optimization of such architectures

challenging. Second, deep neural networks are not currently interpretable, making the explainability of their predictions difficult. Similarly, we exclude from our comparison methods for hierarchical **multilabel** classification [12], [32]–[35], in which data instances may be associated with several distinct paths of the class hierarchy at the same time.

D. Evaluation Metrics

- **Precision/Recall oriented metrics**: Given a class $j \in \mathcal{T}$, let TP_j, FP_j, and FN_j be the number of true positives, false positives, and false negatives, respectively. The precision and recall for $\mathcal{T}$ can then be computed as [33]:

$$P=\frac{\sum_{j\in\mathcal{T}} TP_j}{\sum_{j\in\mathcal{T}} TP_j+\sum_{j\in\mathcal{T}} FP_j}, \quad R=\frac{\sum_{j\in\mathcal{T}} TP_j}{\sum_{j\in\mathcal{T}} TP_j+\sum_{j\in\mathcal{T}} FN_j}. \tag{13}$$

  By combining P and R, we can then compute **Macro–F1**, i.e., the arithmetic mean of per–class F1–scores, where $F1 = 2 \cdot \frac{P \cdot R}{P+R}$. On the other hand, **Micro–F1** [34] measures classification performance by counting the true positive, false negatives and false positives globally, thus accounting for class imbalance [36].
- **Hierarchical Classification Metrics**
  - **Hierarchical Top–k accuracy** [29]: Top–k accuracy is used to quantify the effectiveness of a classifier in the first k predicted labels, where k is the depth of the tree–structure hierarchy. This metric can be computed as $\frac{1}{k}$(# of true positives in the top–k labels of $\hat{\mathbf{y}}$).
  - **Hierarchical F–measure** [37]: Let $\hat{P}(i)$ be the set consisting of the labels from each layer predicted for data i and their ancestors' classes, and let $\hat{T}(i)$ be the set consisting of the true labels from each layer for data i and all their ancestors' classes. The hierarchical precision (hP) and recall (hR) are defined as:

$$hP = \frac{\sum_i |\hat{P}_i \cap \hat{T}_i|}{\sum_i |\hat{P}_i|}, \quad hR = \frac{\sum_i |\hat{P}_i \cap \hat{T}_i|}{\sum_i |\hat{T}_i|}. \tag{14}$$

    The hierarchical F score (H-F) is then computed similar to F1 but using the hP and hR scores, respectively.
  - **HMC–loss** [38]: The Hierarchical Multiclass Classification loss can be computed as $l(\mathbf{y}, \hat{\mathbf{y}}) = \alpha \sum_{j:y_j=1 \wedge \hat{y}_j=-1} c_j + \beta \sum_{j:y_j=-1 \wedge \hat{y}_j=1} c_j$, where c_j is a fixed cost that is assigned to the misclassified node j defined by Eq (5). Following the methodology used in [38] for parameter selection, we compute $\lambda = \frac{n_-}{n_+}$, where n_- is the number of negative training labels and n_+ is the number of positive training labels. Parameters α and β, such that $\alpha = \lambda\beta$ and $\alpha + \beta = 2$ are set in the same way as in [38].

E. Results

We begin by comparing `H–Ada.MH` with Adaboost.MH. Our goal is to show that the performance of AdaBoost.MH improves with the proposed hierarchy–aware loss function and asymmetric weighting scheme. Figure 4 shows the results.

TABLE II
PERFORMANCE COMPARISON.

Dataset	Metric	H-Ada.MH	Local	Global
ImageCLEF07A	Micro-F1	**0.7143**	0.569	0.6633
	Macro-F1	**0.5096**	0.1504	0.470
	H-F	**0.7445**	0.573	0.691
	HMC	**0.075**	0.137	0.079
ImageCLEF07D	Micro-F1	**0.7216**	0.646	0.644
	Macro-F1	0.3992	0.1546	**0.4552**
	H-F	**0.7607**	0.662	0.670
	HMC	**0.129**	0.2047	0.1948
Diatoms	Micro-F1	**0.487**	0.174	0.205
	Macro-F1	**0.5587**	0.0138	0.2391
	H-F	**0.4542**	0.1176	0.1812
	HMC	**0.012**	0.0174	0.0187

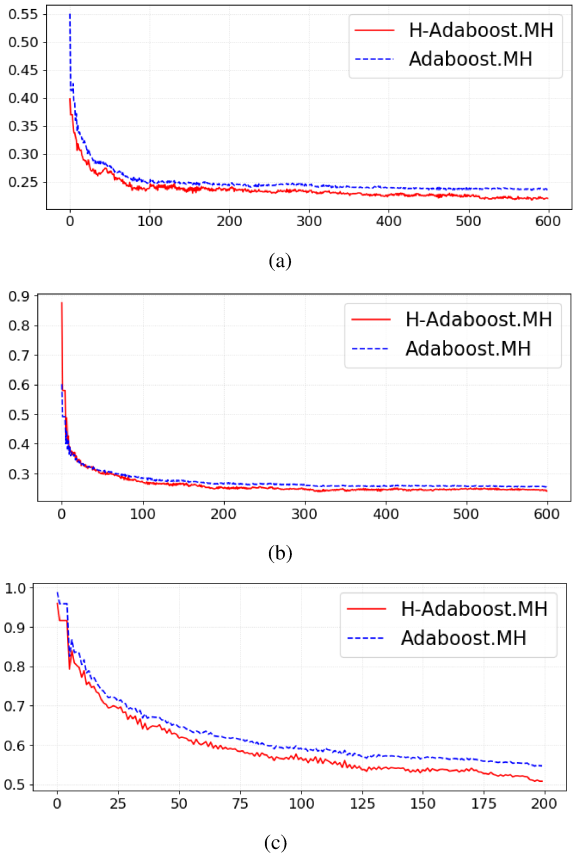

Fig. 4. Comparison of `H–Adab.MH` with Adaboost.MH. in (a) ImageCLEF07D, (b) ImageCLEF07A, and (c) Diatoms. The test error ($y-$axis) in the test dataset is shown as a function of the number of base learners ($x-$axis).

As expected `H–Ada.MH` achieves the smallest loss in all cases. The competitive advantage of `H–Ada.MH` becomes more prominent in the Diatoms dataset, where the number of classes dramatically increases. Misclassifications near the bottom of the hierarchy are avoided by `H–Ada.MH` since, unlike Ada.MH, it accounts for the structural relationships among classes while making predictions.

Next, we compare `H-Ada.MH` with the hierarchical multiclass classification baselines. The results are show in Table II. With the exception of one metric for the ImageCLEF07D

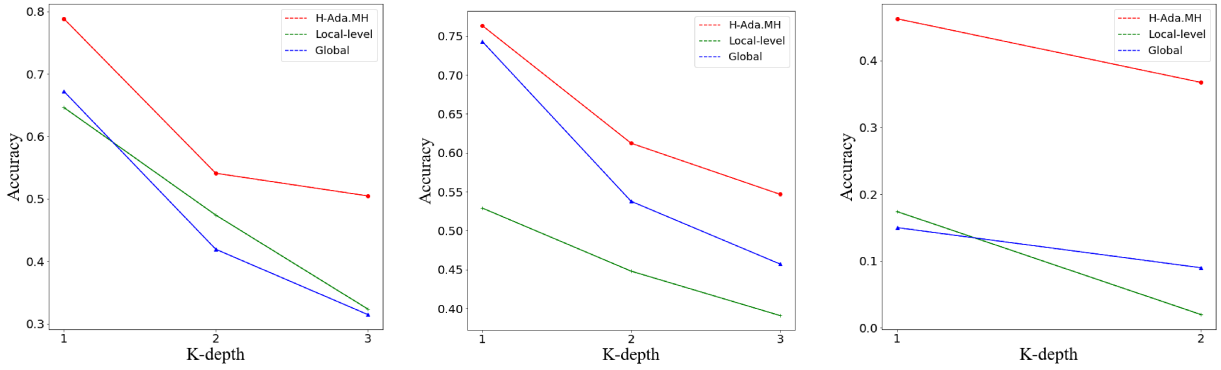

Fig. 5. Top–k accuracy comparison in (a) ImageCLEF07D, (b) ImageCLEF07A, and (c) Diatoms datasets.

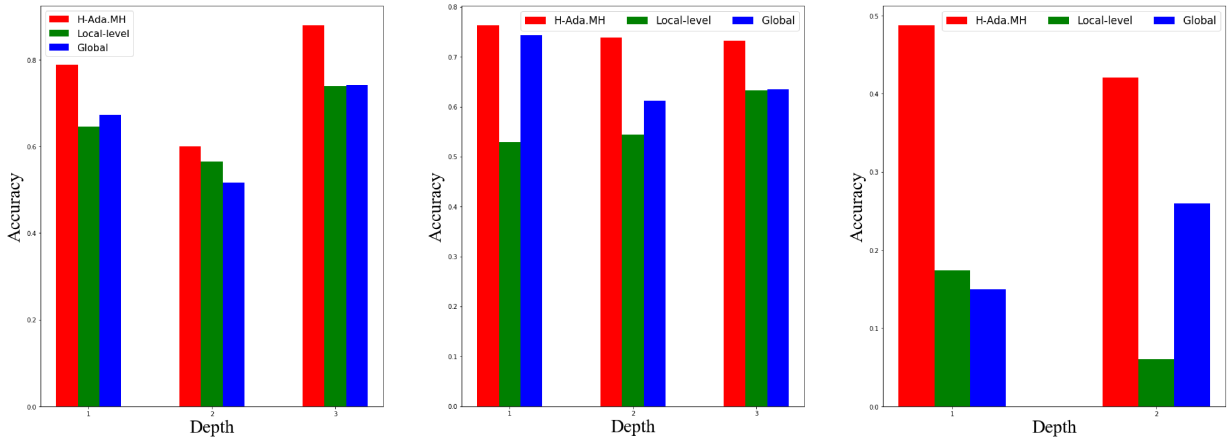

Fig. 6. Per–level accuracy comparison for (a) ImageCLEF07D, (b) ImageCLEF07A, and (c) Diatoms datasets.

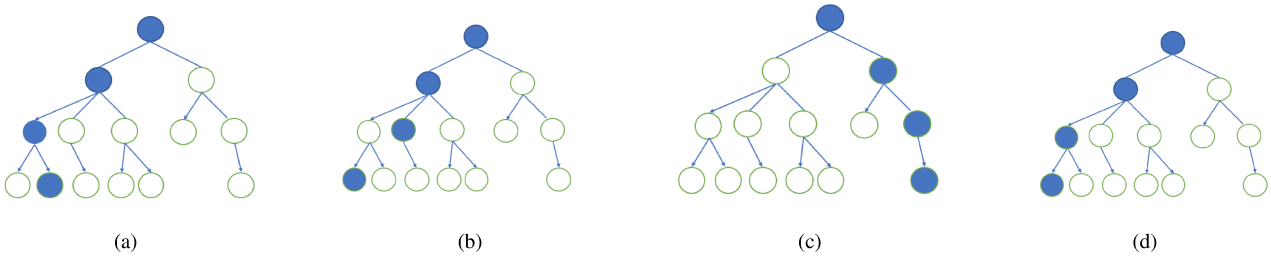

Fig. 7. Ground truth (a) and corresponding predictions ((b) Global, (c) Local, and (d) `H–Ada.MH` for a sample drawn from the ImageCLFE07A dataset.

dataset, the proposed approach outperforms the baselines.

As the goal is to predict all classes in the entire hierarchy, labeling classes correctly at each level is critical. Thus, to better show the superiority of `H-Ada.MH`, our discussion focuses next on Top–k, H–F and HMC loss, all of which have been used to measure hierarchical classification performance.

The advantage of `H–Ada.MH` seems to be smaller with respect to HMC in the ImageCLEF07A dataset, as errors in the upper levels contribute more to the HMC loss. Although this result may be misleading initially, `H–Ada.MH` indeed achieves better performance as it (i) can better discriminate regardless of depth (c.f. Fig 6), and (ii) leads to more consistent labels, i.e., predicted classes form a path from the root to a leaf node that satisfies the tree–structure constraint (Fig. 5).

The Top–k accuracy results are showed in Figure 5, whereas Figure 6 shows classification accuracy for each level in the hierarchy. In all cases, `H–Ada.MH` outperforms the other methods on all levels of the hierarchy. Moreover, although the performance of all methods tends to decrease when the hierarchy deepens, `H–Ada.MH` not only retains superior performance, but also broadens its superiority with the baselines as depth increases. Specifically, as depth increases, the number

of classes in a level increase rapidly (e.g., there are 85 and 258 classes in levels 1 and 2, respectively, in the Diatoms dataset), adversely impacting performance. Since H–Ada.MH considers the dependencies among different levels of the hierarchical structure, it is less affected by the number of classes in a level (as opposed to the baselines).

To further illustrate the difference between H–Ada.MH and the baselines, Figure 7 shows the prediction for a test sample from the ImageCLFEF07A dataset. Apparently, no single method can obtain the corerct solution, with the local method being completely off. Compared to the Global baseline (Fig. 7(b)), H–Ada.MH (Fig. 7(c)) (i) returns a consistent path, and (ii) makes only one mistake at the leaf node (i.e., the most specific level of the hierarchy).

In summary, the experimental results all indicate that H–Ada.MH can classify data instances more effectively than the baselines by learning multiple base classifiers that are explicitly trained to model dependencies among classes, and subsequently used to combine predictions at each level in the class hierarchy in a way that conforms to the overall hierarchical structure.

VI. Conclusion

We presented H–Ada.MH, a new algorithm for hierarchical multiclass classification. Specifically, we first introduced hierarchical constraints into AdaBoost.MH based on a hierarchy–aware Hamming loss function. Then, we devised an asymmetric weighting scheme to penalize errors at the upper levels of the hierarchy, and therefore avoid propagating misclassification down a path. Our experiments on three real–world datasets demonstrated the effectiveness and intuitiveness of H–Ada.MH.

In the future, we wish to address a problem setting where the class hierarchy is organized as a directed acyclic graph instead of a tree. Subsequently, we would like to explore the more challenging problem of multi–label classification, where each data instance can be assigned to multiple paths of the class hierarchy at the same time. Finally, we plan to test the performance of H–Ada.MH on other domains, such as document categorization and bioinformatics.

VII. Ethics Statement

The work presented in this paper is the result of our original work, and is not currently under consideration for publication elsewhere. The names and affiliations of those who actually did the work and contributed to the paper in a meaningful way will be included upon acceptance.

Beyond meeting our professional conduct obligations, it is difficult for us to anticipate or predict how the proposed general framework presented here will be used in the real–world. Therefore, the potential harm, if any, caused by irresponsible use of the proposed framework cannot be easily quantified. In its core, H–Ada.MH produces probable, yet uncertain outcomes (i.e., labels) after being trained on a specific dataset. If a biased dataset is used to train an ensemble with H–Ada.MH, then the outcomes of such classifier may be used in a way that may not be ethically neutral. Recognizing this limitation is important. The fact however that H–Ada.MH produces paths which are open and accessible to humans for interpretation, may be beneficial in identifying biases in the training data itself.

References

[1] R. T. Clemen, "Combining forecasts: A review and annotated bibliography," *International journal of forecasting*, vol. 5, no. 4, pp. 559–583, 1989.

[2] L. K. Hansen and P. Salamon, "Neural network ensembles," *IEEE transactions on pattern analysis and machine intelligence*, vol. 12, no. 10, pp. 993–1001, 1990.

[3] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[4] S. Hashem, "Optimal linear combinations of neural networks," *Neural networks*, vol. 10, no. 4, pp. 599–614, 1997.

[5] D. Opitz and R. Maclin, "Popular ensemble methods: An empirical study," *Journal of artificial intelligence research*, vol. 11, pp. 169–198, 1999.

[6] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of computer and system sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[7] R. Caruana and A. Niculescu-Mizil, "An empirical comparison of supervised learning algorithms," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 161–168.

[8] Y. Freund, R. Schapire, and N. Abe, "A short introduction to boosting," *Journal-Japanese Society For Artificial Intelligence*, vol. 14, no. 771-780, p. 1612, 1999.

[9] E. L. Allwein, R. E. Schapire, and Y. Singer, "Reducing multiclass to binary: A unifying approach for margin classifiers," *Journal of machine learning research*, vol. 1, no. Dec, pp. 113–141, 2000.

[10] J. Friedman, T. Hastie, R. Tibshirani *et al.*, "Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors)," *The annals of statistics*, vol. 28, no. 2, pp. 337–407, 2000.

[11] A. Freitas and A. Carvalho, "A tutorial on hierarchical classification with applications in bioinformatics," in *Research and trends in data mining technologies and applications*. IGI Global, 2007, pp. 175–208.

[12] G. Tsoumakas, I. Katakis, and I. Vlahavas, "Mining multi-label data," in *Data mining and knowledge discovery handbook*. Springer, 2009, pp. 667–685.

[13] I. Dimitrovski, D. Kocev, S. Loskovska, and S. Džeroski, "Hierarchical annotation of medical images," *Pattern Recognition*, vol. 44, no. 10-11, pp. 2436–2449, 2011.

[14] C. N. Silla and A. A. Freitas, "A survey of hierarchical classification across different application domains," *Data Mining and Knowledge Discovery*, vol. 22, no. 1-2, pp. 31–72, 2011.

[15] F. Wu, J. Zhang, and V. Honavar, "Learning classifiers using hierarchically structured class taxonomies," in *International symposium on abstraction, reformulation, and approximation*. Springer, 2005, pp. 313–320.

[16] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li, "Rcv1: A new benchmark collection for text categorization research," *Journal of machine learning research*, vol. 5, no. Apr, pp. 361–397, 2004.

[17] L. Cai and T. Hofmann, "Hierarchical document categorization with support vector machines," in *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, 2004, pp. 78–87.

[18] N. Cesa-Bianchi, C. Gentile, and L. Zaniboni, "Incremental algorithms for hierarchical classification," *Journal of Machine Learning Research*, vol. 7, no. Jan, pp. 31–54, 2006.

[19] K. Kowsari, D. E. Brown, M. Heidarysafa, K. J. Meimandi, M. S. Gerber, and L. E. Barnes, "Hdltex: Hierarchical deep learning for text classification," in *2017 16th IEEE international conference on machine learning and applications (ICMLA)*. IEEE, 2017, pp. 364–371.

[20] N. Cesa-Bianchi and G. Valentini, "Hierarchical cost-sensitive algorithms for genome-wide gene function prediction," in *Machine Learning in Systems Biology*, 2009, pp. 14–29.

[21] A. Sokolov and A. Ben-Hur, "Hierarchical classification of gene ontology terms using the gostruct method," *Journal of bioinformatics and computational biology*, vol. 8, no. 02, pp. 357–376, 2010.

[22] C. J. Fall, A. Törcsvári, K. Benzineb, and G. Karetka, "Automated categorization in the international patent classification," in *Acm Sigir Forum*, vol. 37, no. 1. ACM New York, NY, USA, 2003, pp. 10–25.

[23] R. E. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated predictions," *Machine learning*, vol. 37, no. 3, pp. 297–336, 1999.

[24] M. Kearns and L. Valiant, "Cryptographic limitations on learning boolean formulae and finite automata," *Journal of the ACM (JACM)*, vol. 41, no. 1, pp. 67–95, 1994.

[25] T. Hastie, S. Rosset, J. Zhu, and H. Zou, "Multi-class adaboost," *Statistics and its Interface*, vol. 2, no. 3, pp. 349–360, 2009.

[26] C. N. Silla Jr and A. A. Freitas, "A global-model naive bayes approach to the hierarchical prediction of protein functions," in *2009 Ninth IEEE International Conference on Data Mining*. IEEE, 2009, pp. 992–997.

[27] J. N. Hernandez, L. E. Sucar, and E. F. Morales, "A hybrid global-local approach for hierarchical classification," in *The Twenty-Sixth International FLAIRS Conference*, 2013.

[28] N. Cesa-Bianchi, C. Gentile, and L. Zaniboni, "Hierarchical classification: combining bayes with svm," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 177–184.

[29] W. Bi and J. T. Kwok, "Bayes-optimal hierarchical multilabel classification," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 11, pp. 2907–2918, 2015.

[30] W. Cheng, E. Hüllermeier, and K. J. Dembczynski, "Bayes optimal multilabel classification via probabilistic classifier chains," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 279–286.

[31] O. Dekel, J. Keshet, and Y. Singer, "Large margin hierarchical classification," in *Proceedings of the twenty-first international conference on Machine learning*, 2004, p. 27.

[32] J. Rousu, C. Saunders, S. Szedmak, and J. Shawe-Taylor, "Kernel-based learning of hierarchical multilabel classification models," *Journal of Machine Learning Research*, vol. 7, no. Jul, pp. 1601–1626, 2006.

[33] C. Vens, J. Struyf, L. Schietgat, S. Džeroski, and H. Blockeel, "Decision trees for hierarchical multi-label classification," *Machine learning*, vol. 73, no. 2, p. 185, 2008.

[34] M.-L. Zhang and Z.-H. Zhou, "A review on multi-label learning algorithms," *IEEE transactions on knowledge and data engineering*, vol. 26, no. 8, pp. 1819–1837, 2013.

[35] J. Wehrmann, R. Cerri, and R. Barros, "Hierarchical multi-label classification networks," in *International Conference on Machine Learning*, 2018, pp. 5075–5084.

[36] B. Yang, J.-T. Sun, T. Wang, and Z. Chen, "Effective multi-label active learning for text classification," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2009, pp. 917–926.

[37] S. Kiritchenko, S. Matwin, A. F. Famili *et al.*, "Functional annotation of genes using hierarchical text categorization," in *Proc. of the ACL Workshop on Linking Biological Literature, Ontologies and Databases: Mining Biological Semantics*, 2005.

[38] W. Bi and J. T. Kwok, "Hierarchical multilabel classification with minimum bayes risk," in *2012 IEEE 12th International Conference on Data Mining*. IEEE, 2012, pp. 101–110.