Exploiting Label Dependency for Hierarchical Multi-label Classification

Noor Alaydie, Chandan K. Reddy, and Farshad Fotouhi

Department of Computer Science
Wayne State University, Detroit, MI, USA
alaydie@wayne.edu, reddy@cs.wayne.edu, fotouhi@wayne.edu

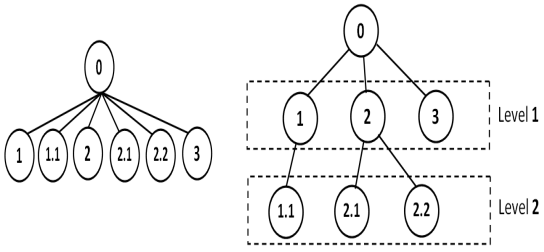
Abstract. Hierarchical multi-label classification is a variant of traditional classification in which the instances can belong to several labels, that are in turn organized in a hierarchy. Existing hierarchical multi-label classification algorithms ignore possible correlations between the labels. Moreover, most of the current methods predict instance labels in a "flat" fashion without employing the ontological structures among the classes. In this paper, we propose HiBLADE (Hierarchical multi-label Boosting with LAbel DEpendency), a novel algorithm that takes advantage of not only the pre-established hierarchical taxonomy of the classes, but also effectively exploits the hidden correlation among the classes that is not shown through the class hierarchy, thereby improving the quality of the predictions. According to our approach, first, the pre-defined hierarchical taxonomy of the labels is used to decide upon the training set for each classifier. Second, the dependencies of the children for each label in the hierarchy are captured and analyzed using Bayes method and instance-based similarity. Our experimental results on several real-world biomolecular datasets show that the proposed method can improve the performance of hierarchical multi-label classification.

Keywords: Hierarchical multi-label classification, correlation, boosting.

1 Introduction

Traditional classification tasks deal with assigning instances to a single label. In multi-label classification, the task is to find the set of labels that an instance can belong to rather than assigning a single label to a given instance. Hierarchical multi-label classification is a variant of traditional classification where the task is to assign instances to a set of labels where the labels are related through a hierarchical classification scheme. In other words, when an instance is labeled with a certain class, it should also be labeled with all of its superclasses, this is known as the *hierarchy constraint*.

Hierarchical multi-label classification is a widely studied problem in many domains such as functional genomics, text categorization, image annotation and object recognition. In functional genomics (which is the application that we focus on in this paper) the problem is the prediction of gene/protein functions. Biologists have a hierarchical organization of the functions that the genes can be assigned to. An individual gene or protein may be involved in more than one biological activity, and hence, there is a need for a prediction algorithm that is able to identify all the possible functions of a particular



(a) Flat (b) Hierarchical

Fig. 1. Flat versus Hierarchical classification. (a) Flat representation of the labels (b) Hierarchical representation of the same set of labels.

gene. There are two types of class hierarchy structures: a rooted tree structure, such as the MIPS's FunCat taxonomy [17], and a directed acyclic graph (DAG) structure, such as the Gene Ontology (GO) [7]. In this paper, we use the FunCat scheme.

Most of the existing research focuses on a "flat" classification approach, that operates on non-hierarchical classification schemes, where a binary classifier is constructed for each label separately as shown in Figure 1(a). This approach ignores the hierarchical structure of the classes shown in Figure 1(b). Reducing a hierarchical multi-label classification problem to a conventional classification problem allows the possibility of applying the existing methods. However, since the prediction of the class labels has to be performed independently, such transformations are not capable of exploiting the interdependencies and correlations between the labels [6]. Moreover, the flat classification algorithm fails to take advantage of the information inherent in the class hierarchy, and hence may be suboptimal in terms of efficiency and effectiveness [9].

1.1 Our Contributions

In this paper, we propose, HiBLADE, a hierarchical multi-label classification framework for modeling the pre-defined hierarchical taxonomy of the labels as well as for exploiting the existing correlations between different labels, that are not given by the taxonomical classification of the labels, to facilitate the learning process.

To the best of our knowledge, *there is no work related to the hierarchical multi-label classification setting that exploits the correlations between different labels other than the domain-based pre-established taxonomical classification of the classes*. Our intuition is that the domain-based taxonomical classification of the classes should be used as additional features while label dependencies are inferred from the data. Specifically, a novel approach to learn the label dependencies using a Bayesian framework and instance-based similarity is proposed. Bayesian framework is used to characterize the dependency relations among the labels represented by the directed acyclic graph (DAG) structure of the Bayesian network. As opposed to other hierarchical multi-label classification algorithms, our algorithm has the following advantages:

1. The underlying pre-defined taxonomy of the labels is explicitly expressed which allows us to gain further insights into the learning problem.

2. It is capable of addressing correlations and interdependencies among the children of a particular label using the Bayesian framework and instance-based similarity.
3. The use of a shared boosting model for the child labels for each label in the hierarchy using the obtained correlations leads to efficient and effective results.

The rest of the paper is organized as follows: related work is discussed in Section 2. Our proposed method, HiBLADE, is presented in Section 3. In Section 4, we report our experimental results on several biomolecular datasets. Then, we conclude and discuss further research directions in Section 5.

2 Related Work

Since conventional classification methods, such as binary classification and multi-class classification, were not designed to directly tackle the hierarchical classification problems, such algorithms are referred to as flat classification algorithms [18]. It is important to mention that flat classification and other similar approaches are not considered to be a hierarchical classification approach, as they create new (meta) classes instead of using pre-established taxonomies.

Different approaches have been proposed in the literature to tackle the hierarchical multi-label classification problem [4, 21, 16]. Generally, these approaches can be grouped into two categories: the local classifier methods and the global classifier methods. Moreover, most of the existing methods use a top-down class prediction strategy in the testing phase [3, 16, 18]. The local strategy treats any label independently, and thus ignores any possible correlation or interdependency between the labels. Therefore, some methods perform an additional step to correct inconsistent predictions. For example, in [3], a Bayesian framework is developed for correcting class-membership inconsistency for the separate class-wise models approach. In [1], a hierarchical multi-label boosting algorithm, named HML-Boosting, was proposed to exploit the hierarchical dependencies among the labels. HML-Boosting algorithm relies on the hierarchical information and utilizes the hierarchy to improve the prediction accuracy.

True path rule (TPR) is a rule that governs the annotation of GO and FunCat taxonomies. According to this rule, annotating a gene to a given class is automatically transferred to all of its ancestors to maintain the hierarchy constraint [12]. In [20], a true path ensemble method was proposed. In this method, a classifier is built for each functional class in the training phase. A bottom-up approach is followed in the testing phase to correct the class-membership inconsistency. In a modified version of TPR ($TPR - w$), a parent weight is introduced. The weight is used to explicitly modulate the contribution of the local predictions with the positive predictions coming from the descendant nodes. In [5], a hierarchical bottom-up Bayesian cost-sensitive ensemble (HBAYES-CS), is proposed. Basically, a calibrated classifier is trained at each node in the taxonomy. H-loss is used in the evaluation phase to predict the labels for a given node. In a recent work [2], we proposed a novel Hierarchical Bayesian iNtegration algorithm HiBiN, a general framework that uses Bayesian reasoning to integrate heterogeneous data sources for accurate gene function prediction. On the other hand, several research groups have studied the effective exploitation of correlation information among different labels in the context of multi-label learning. However, these approaches

do not assume the existence of any pre-defined taxonomical structure of the classes [6, 11, 14, 23, 24].

3 HiBLADE Algorithm

Let $\mathcal{X} = \Re^d$ be the d-dimensional input space and $\mathcal{Y} = \{y_1, y_2, ..., y_\mathcal{L}\}$ be the finite set of $\mathcal{L}$ possible labels. The hierarchical relationships among classes in $\mathcal{Y}$ are defined as follows: Given $y_1, y_2 \in \mathcal{Y}, y_1$ is the ancestor of y_2, denoted by $(\uparrow y_2) = y_1$, if and only if y_1 is a superclass of y_2.

Let a hierarchical multi-label training set $\mathcal{D} = \{< x_1, \mathcal{Y}_1 >, ..., < x_N, \mathcal{Y}_N >\}$, where $x_i \in \mathcal{X}$ is a feature vector for instance i and $\mathcal{Y}_i \subseteq \mathcal{Y}$ is the set of labels associated with x_i, such that $y_i \in \mathcal{Y}_i \Rightarrow y'_i \in \mathcal{Y}_i, \forall(\uparrow y_i) = y'_i$. Having $\mathcal{Q}$ as the quality criterion for evaluating the model based on the prediction accuracy, the objective function is defined as follows: a function $\mathfrak{f} : \mathcal{D} \rightarrow 2^y$. Here, 2^y is the power set of $\mathcal{Y}$, such that $\mathcal{Q}$ is maximized, and $y' \in \mathfrak{f}(x) \Rightarrow y \in \mathfrak{f}, \forall(\uparrow y') = y$. The function $\mathfrak{f}$ is represented here by the ***HiBLADE*** algorithm.

Hierarchical multi-label learning aims to model and predict $p(child\ class \mid parent\ class)$. Our goal is to make use of hierarchical dependencies as well as the extracted dependencies among the labels y_k where $1 \leq k \leq \mathcal{L}$ and $(\uparrow y_k) = y_m$ such that for each example we can better predict its labels. The problem then becomes how to identify and make use of such dependencies in an efficient way.

3.1 Training Scheme

The training of each classifier is performed locally. During classification, the classifier at each class will only be presented with examples that are positive at the parent class of the current class. Hence, the reached examples are positive examples to the current class and/or to the siblings of that class. In other words, the training for each classifier is performed by feeding as negative training examples, the positive examples at the parent of the current class that are not positive examples at the current class.

3.2 Extending the Features

The feature vector for each example is extended to include the class labels of the levels higher in the hierarchy than the current level as given in line 9 of Algorithm 1. More formally, the feature vector for each instance j that belongs to a class i will have the following form: $f_{i,j} =< x_{j,1}, ..., x_{j,d}, \bar{x}_{j,d+1}, ..., \bar{x}_{j,d+k} >$, where $< x_{j,1}, ..., x_{j,d} >$ is the original feature vector and $< \bar{x}_{j,d+1}, ..., \bar{x}_{j,d+k} >$ are the additional features.

3.3 Label Correlation

The other type of dependencies is modeled using Bayesian structure and instance-based similarity as shown in line 10 of Algorithm 1. For each class i, we get the children classes of class i and $sharedModels$ algorithm (shown in Algorithm 2) is invoked. In each boosting iteration t, the entire pool is searched for the best fitted model other

Algorithm 1 $HiBLADE$

Input: A pair $< \mathcal{Y}, \mathcal{L} >$ where $\mathcal{Y}$ is a tree-structured set of classes and $\mathcal{L}$ is the total number of classes of $\mathcal{Y}$.
Output: For each class $y_l \in \mathcal{Y}$, the final composite classifier $y_l = sign[F_l(x)]$.
Algorithm:
for $i = 1, ..., \mathcal{L}$ **do**
  if class i is a leaf class **then**
    Do nothing
  else
    Let $children(i) = y_{1i}, ..., y_{ki}$ be the k children classes of i
    Form the new feature vectors by adding the labels of the classes at the higher levels to the current feature vectors.
    Learn classifiers for $children(i)$ using the shared models Algorithm
  end if
end for

than the model that was built directly for that label and its corresponding combination weights, the best fitted model is called h_l^t. We refer to the best fitted model as the candidate model. The chosen model h_c^t is then updated based on the following formula:

$$\gamma_{ij} = \frac{\epsilon_{ii}}{\epsilon_{ii} + \epsilon_{ji}} * \beta_{ij} \tag{1}$$

where ϵ_{ji} is the error results from applying model h_j^t on the examples in class i and ϵ_{ii} is the error results from applying the model h_i^t on the examples in class i. β_{ij} controls the proportional contribution of Bayesian-based and instance-based similarities. β_{ij} is computed as follows:

$$\beta_{ij} = \phi * b_{ij} + (1 - \phi) * s_{ij} \tag{2}$$

where b_{ij} is the Bayesian correlation between class i and class j, and it is estimated as $b_{ij} = |i \cap j|/|j|$, where $|i \cap j|$ is the number of positive examples in class i and class j and $|j|$ is the number of positive examples in class j. s_{ij} is the instance-based similarity between class i and class j. Each instance from one class is compared to each other instance from the other class. In HiBLADE, s_{ij} is computed using the Euclidean distance between the positive examples in both classes that has the following formula:

$$s_{ij} = \sqrt{\sum_l (i_l - j_l)^2} \tag{3}$$

where l is the corresponding feature in the two vectors. s_{ij} is normalized to be in the range of $[0, 1]$. ϕ is a threshold parameter that has a value in the range $[0, 1]$. Setting ϕ to 0 means that only instance-based similarity is taken into consideration in the learning process. While setting it to 1 means that only Bayesian-based correlation is taken into consideration. On the other hand, any value of ϕ between 0 and 1 combines both types of correlation. It is important to emphasize that these computations are performed only for the class that is found to be the most useful class with respect to the current class.

In the general case, both classes, the current class and the candidate class, contribute to the final prediction. In other words, any value of ϵ_{ji} other than 0, indicates the level of contribution from the candidate class. More specifically, if the error of the candidate class, ϵ_{ji}, is greater than the error of the current class, ϵ_{ii}, the value of γ_{ij} will be small indicating that only a limited contribution of the candidate class is considered. In contrast, if the error of the current class, ϵ_{ii}, is greater than the error of the candidate class, ϵ_{ji}, then γ_{ij} will be high, and hence, the prediction decision will be dependable more on the candidate class. Finally, the models for the current class and the used candidate class are replaced by the new learned models. At the end, the composite classifiers F_c provide the prediction results.

Algorithm 2 shows the details of the shared models algorithm. The shared models algorithm takes as input the children classes of a particular class together with the feature vectors for the instances that are positive at the parent class. These instances will form the positive and negative examples for each one of the children classes. The algorithm begins by initializing a pool of M models, where M is the number of children classes, one for each class that is learned using a boosting-type algorithm such as ADABOOST. The number of base models to be generated is determined by T. In each iteration t and for each label in the set of the children labels, we look for the best fitted model, $h_l^t(x)$ and the corresponding combination weights, α_l^t. The contribution of the selected base model, $h_l^t(x)$, to the overall classifier, $F_c(x)$, depends on the current label. In other words, if the error, ϵ_{ji} of the candidate classifier is 0, this will be a perfect model for the current label. Hence, equation (1) will be reduced to $\gamma_{ij} = \beta_{ij}$. In this case, the contribution of that model depends on the level of correlation between the candidate class and the current one. On the other hand, if the current model is a perfect model, i.e., the error $\epsilon_{ii} = 0$, then equation 1 will be reduced to $\gamma_{ij} = 0$, which means that for the current iteration, there is no need to look at any other classifier.

Algorithm 2 $SharedModels$

Input: $D = \{(x_i, Y_i) : i = 1, ..., N\}$, where $x_i \in X$ is a feature vector for instance i and $Y_i \subset Y$ is the set of labels associated with x_i and M is the number of labels under study. ϕ: a threshold parameter.
Output: $y_c = sign[F_c(x)]$
Algorithm:
Set $F_c(x) = 0$ for each label $c = 1, .., M$
Initialize a pool of candidate shared models: $SM_p = h_1(.), ..., h_M(.)$ where $h_i(.)$ is a model learned on the label i using the boosting-type algorithm.
for $t = 1, ..., T$ **do**
  for $c = 1, ..., M$ **do**
    Find α_l^t and $h_l^t \in SM_p$ where $c \neq l$ that minimize the loss function on label c.
    $F_c(x) = F_c(x) + h_c^t(x) * \alpha_c^t * (1 - \gamma_{cl}) + h_l^t(x) * \alpha_l^t * \gamma_{cl}$
    Replace $h_c^t(x)$ and $h_l^t(x)$ in SM_p with the new learned models using the boosting-type algorithm.
  end for
end for

4 Experimental Details

We chose to demonstrate the performance of our algorithm for the prediction of gene functions in yeast using four bio-molecular datasets that were used in [20]. Valentini [20] pre-processed the datasets so that for each dataset, only genes that are annotated with FunCat taxonomy are selected. To make this paper self-contained, we briefly explain the data collection process and the pre-processing steps performed on the data. Uninformed features that have the same value for all of the examples are removed. Class "99" in FunCat corresponds to an "unclassified protein". Therefore, genes that are annotated only with that class are excluded. Finally, in order to have a good size of positive training examples for each class, selection has been performed to classes with at least 20 positive examples. Dataset characteristics are summarized in Table 1.

Table 1. The characteristics of the four bio-molecular datasets used in our experiments.

Dataset	Description	Samples	Features	classes
Gene-Expr	Gene expression data	4532	250	230
PPI-BG	PPI data from BioGRID	4531	5367	232
Pfam-1	Protein domain binary data	3529	4950	211
PPI-VM	PPI data from Von Mering experiments	2338	2559	177

The gene expression dataset, Gene-Expr, is obtained by merging the results of two studies, gene expression measures relative to 77 conditions and transcriptional responses of yeast to environmental stress measured on 173 conditions [10]. For each gene product in the protein-protein interaction dataset, PPI-BG, a binary vector is generated that implies the presence or absence of protein-protein interaction. Protein-protein interaction data have been downloaded from BioGRID database [19, 20]. In Pfam-1 dataset, a binary vector is generated for every gene product that reflects the presence or absence of 4950 protein domains obtained from Pfam (Protein families) database [8, 20]. For PPI-VM dataset, Von Mering experiments produced protein-protein data from yeast two-hybrid assay, mass spectrometry of purified complexes, correlated mRNA expression and genetic interactions [22].

Table 2. Per-level F_1 measure for Gene-Expr dataset using Flat, $HiBLADE_I$, $HiBLADE_C$ with $\phi = 0.5$ and $HiBLADE_B$ for boosting iterations=50.

Level	Flat	$HiBLADE_I$ $\phi = 0.0$	$HiBLADE_C$ $\phi = 0.5$	$HiBLADE_B$ $\phi = 1.0$
1	**0.3537**	0.2328	0.2301	0.2336
2	0.1980	0.4052	**0.4427**	0.4094
3	0.1000	0.3575	**0.4019**	0.3742
4	0.2000	0.2714	**0.3598**	0.2874

4.1 Evaluation Metrics

Classical evaluation measures such as precision, recall and F-measure are used by unstructured classification problems and thus, they are inadequate to address the hierarchical natures of the classes. Another approach that is used for the hierarchical multi-label learning is to use extended versions of the single label metrics (precision, recall and F-measure). To evaluate our algorithm, we adopted both, the classical and the hierarchical evaluation measures. F_1 measure considers the joint contribution of both precision (P) and recall (R). F_1 measure is defined as follows:

$$F_1 = \frac{2 \times P \times R}{P + R} = \frac{2TP}{2TP + FP + FN} \tag{4}$$

where TP stands for True Positive, TN for True Negative, FP for False Positive and FN for False Negative. When TP=FP=FN=0, we made F_1 measure to equal to 1 as the classifier has correctly classified all the examples as negative examples [9]. Hierarchical measures are defined as follows:

$$hP = \frac{1}{|l(P(x))|} \sum_{p \in l(P(x))} \frac{|C(x) \cap \uparrow p|}{|\uparrow p|} \tag{5}$$

$$hR = \frac{1}{|l(C(x))|} \sum_{c \in l(C(x))} \frac{|\uparrow c \cap P(x)|}{|\uparrow c|} \tag{6}$$

$$hF = \frac{2 \times hP \times hR}{hP + hR} \tag{7}$$

where hP, hR and hF stands for hierarchical precision, hierarchical recall and hierarchical F-measure, respectively. $P(x)$ is a subgraph formed by the predicted class labels for the instance x while $C(x)$ is a subgraph formed by the true class labels for the instance x. p is one of the predicted class labels and c is one of the true labels for instance x. $l(P(x))$ and $l(C(x))$ are the set of leaves in graphs $P(x)$ and $C(x)$, respectively. We also computed both micro-averaged hierarchical F-measure (hF_1^{μ}) and macro-averaged hierarchical F-measure hF_1^M. hF_1^{μ} is computed by computing hP and hR for each path in the hierarchical structure of the tree and then applying equation (7). On the other hand, hF_1^M is computed by calculating hF_1 for each path in the hierarchical structure of the classes independently and then averaging them. Having high hierarchical precision means that the predictor is capable of predicting the most general functions of the instance, while having high hierarchical recall indicates that the predictor is able to predict the most specific classes [20]. The hierarchical F-measure takes into account the partially correct paths in the overall taxonomy.

4.2 Experimental Results and Discussion

We analyzed the performance of the proposed framework at each level of the FunCat taxonomy, and we also compared the proposed method with four other methods that follow the local classifier approach, namely, HBAYES-CS, HTD, TPR and TPR-w. HBAYES-CS, TPR and TPR-w are described in the Related Work Section. HTD (Hierarchical Top-Down) is the baseline method that belongs to the local classifier strategy

Table 3. Per-level F_1 measure for PPI-BG dataset using Flat, $HiBLADE_I$, $HiBLADE_C$ with $\phi = 0.5$ and $HiBLADE_B$ for boosting iterations=50.

Level	Flat	$HiBLADE_I$ $\phi = 0.0$	$HiBLADE_C$ $\phi = 0.5$	$HiBLADE_B$ $\phi = 1.0$
1	0.0808	0.2014	0.1833	**0.2052**
2	0.0267	0.6904	0.6984	**0.6998**
3	0.0001	0.6446	0.6304	**0.6520**
4	0.0001	0.6743	0.6454	**0.6747**

and performs hierarchical classification in a top-down fashion. Since HiBLADE also belongs to the local classifier strategy, it is fair to have a comparison against a local classifier approach that does not consider any type of correlation between the labels. We also analyzed the effect of the proper choice of the threshold ϕ on the performance of the algorithm. The setup for the experiments is summarized as follows:

- Flat: This is the baseline method that does not take the hierarchical taxonomy of the classes into account and does not consider label dependencies. A classifier is built for each class independently of the others. We used AdaBoost as the base learner to form a baseline algorithm for the comparison with the other methods.
- $HiBLADE_I$: The proposed algorithm that considers Instance-based similarities only. Here ϕ is set to zero.
- $HiBLADE_B$: The proposed algorithm that considers classes correlation based on Bayesian probabilities only. Here ϕ is set to one.
- $HiBLADE_C$: The proposed algorithm that considers a combination of both instance-based similarity and classes correlation. Here ϕ is set to 0.5.

Table 4. Per-level F_1 measure for $Pfam-1$ dataset using Flat, $HiBLADE_I$, $HiBLADE_C$ with $\phi = 0.5$ and $HiBLADE_B$ for boosting iterations=50.

Level	Flat	$HiBLADE_I$ $\phi = 0.0$	$HiBLADE_C$ $\phi = 0.5$	$HiBLADE_B$ $\phi = 1.0$
1	**0.1133**	0.0924	0.0827	0.1085
2	0.0267	0.8524	**0.8702**	0.7273
3	0.1000	0.7473	**0.7946**	0.6824
4	0.2222	0.5122	**0.5135**	0.5085

First, we performed a level-wise analysis of the F-measure of the FunCat classification tree on the four datasets. In measuring the level-wise performance, level 1 reflects the root nodes while all other classes are at depth i, where $2 \leq i \leq 5$. We show the results for the top four levels in the hierarchy for the proposed method and the flat method. Moreover, we show the performance of the proposed framework with different ϕ's values while setting the number of boosting iterations to 50 iterations. Tables 2, 3, 4 and 5 show the results of per-level evaluation for Gene-Expr, PPI-BG, Pfam-1 and PPI-VM datasets, respectively. The most significant measures for each level are highlighted.

The proposed algorithm outperforms the flat classification method in most of the cases with significant differences in the performance measurements. The results in Tables 2, 3, 4 and 5 indicate that the deeper the level the better the performance of the proposed algorithm compared to the flat classification method. For example, in all of the datasets, the proposed algorithm outperformed the flat classification method in all the levels that are higher than level 1. This result is consistent with our understanding of both of the classification schemes. In other words, the proposed method and the flat classification method have a similar learning procedure for the classes in the first level. However, the proposed method achieved better results for the deeper levels in the hierarchy.

Table 5. Per-level F_1 measure for PPI-VM dataset using Flat, $HiBLADE_I$, $HiBLADE_C$ with $\phi = 0.5$ and $HiBLADE_B$ for boosting iterations=50.

Level	Flat	$HiBLADE_I$ $\phi = 0.0$	$HiBLADE_C$ $\phi = 0.5$	$HiBLADE_B$ $\phi = 1.0$
1	**0.1631**	0.1266	0.1029	0.1193
2	0.1786	0.6033	**0.6758**	0.6601
3	0.0001	0.5802	0.6822	**0.6957**
4	0.0001	**0.6931**	0.5246	0.5417

To get more insights into the best choice of ϕ threshold, we compare hierarchical precision, hierarchical recall, hierarchical F_1^μ measure and hierarchical F_1^M measure for Gene-Expr, PPI-BG, Pfam-1 and PPI-VM datasets for $\phi = 0.0, 0.5$ and 1.0, respectively, for 50 boosting iterations. Table 6 shows the results of the comparisons. The most significant measures are highlighted. As shown in Table 6, the combination of Bayesian-based correlation and instance-based similarity achieved the best performance results in most of the cases. For example, six of the highest performance values, in general, in this table are achieved when $\phi = 0.5$.

Table 6. Hierarchical precision, hierarchical recall, hierarchical F_1^M and hierarchical F_1^μ measures of HiBLADE for all the four datasets using boosting iterations =50.

Measure	Gene-Expr			PPI-BG		
	$\phi = 0.0$	$\phi = 0.5$	$\phi = 1.0$	$\phi = 0.0$	$\phi = 0.5$	$\phi = 1.0$
hP	0.820	0.808	**0.826**	0.878	**0.924**	0.875
hR	**0.644**	0.630	0.627	0.662	0.686	**0.701**
hF_1^M	**0.702**	0.689	0.692	0.735	**0.769**	0.756
hF_1^μ	**0.722**	0.708	0.712	0.755	**0.787**	0.778
Measure	Pfam-1			PPI-VM		
	$\phi = 0.0$	$\phi = 0.5$	$\phi = 1.0$	$\phi = 0.0$	$\phi = 0.5$	$\phi = 1.0$
hP	0.763	0.836	**0.875**	0.716	**0.748**	0.719
hR	0.625	**0.663**	0.637	0.542	0.551	**0.557**
hF_1^M	0.669	**0.720**	0.714	0.590	**0.605**	0.601
hF_1^μ	0.687	**0.740**	0.737	0.617	**0.635**	0.628

Furthermore, we conducted comparisons of hierarchical F-measure with HBAYES-CS, HTD, TPR and TPR-w methods. HBAYES-CS is using Guassian SVMs as the base learners, while HTD, TPR and TPR-w are using Linear SVMs as the base learners. Figure 2 shows the F-measure of the different methods. By exploiting the label dependencies, the classifiers performance are effected positively. Our results show that the proposed algorithm significantly outperforms the local learning algorithms. Although there

is no clear winner among the different versions of HiBLADE algorithm, HiBLADE always achieved significantly better results than the other methods.

5 Conclusion

In this paper, we proposed a hierarchical multi-label classification framework for incorporating information about the hierarchical relationships among the labels as well as the label correlations. The experimental results showed that the proposed algorithm, HiBLADE, outperforms the flat classification method and the local classifiers method that builds independent classifier for each class. For future work, we plan to generalize the proposed approach to general graph structures and develop more scalable solutions using some other recent proposed boosting strategies [13, 15].

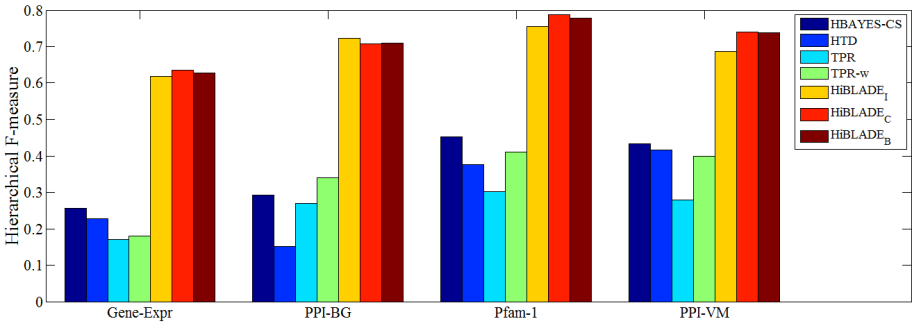


Fig. 2. Hierarchical F-measure comparison between HBAYES-CS, HTD, TPR, TPR-w, $HiBLADE_I$, $HiBLADE_C$ and $HiBLADE_B$. For the $HiBLADE$ algorithm, the number of boosting iterations is 50 and $\phi = 0.5$ for $HiBLADE_C$.

References

1. N. Alaydie, C. K. Reddy, and F. Fotouhi. Hierarchical boosting for gene function prediction. In *Proceedings of the 9th International Conference on Computational Systems Bioinformatics(CSB)*, pages 14–25, Stanford, CA, USA, August 2010.
2. N. Alaydie, C. K. Reddy, and F. Fotouhi. A Bayesian Integration Model of Heterogeneous Data Sources for Improved Gene Functional Inference. In *Proceedings of the ACM Conference on Bioinformatics, Computational Biology and Biomedicine (ACM-BCB)*, pages 376–380, Chicago, IL, USA, August 2011.
3. Z. Barutcuoglu, R. E. Schapire, and O. G. Troyanskaya. Hierarchical multi-label prediction of gene function. *Bioinformatics*, 22(7):830–836, Jan 2006.
4. Wei Bi and James Kwok. Multi-label classification on tree- and dag-structured hierarchies. In Lise Getoor and Tobias Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, ICML '11, pages 17–24, New York, NY, USA, June 2011. ACM.
5. N. Cesa-Bianchi and G. Valentini. Hierarchical cost-sensitive algorithms for genome-wide gene function prediction. In *Machine Learning in Systems Biology, Proceedings of the Third international workshop*, pages 25–34, Ljubljana, Slovenia, 2009.

6. Weiwei Cheng and Eyke Hüllermeier. Combining instance-based learning and logistic regression for multilabel classification. *Machine Learning*, 76(2–3):211–225, 2009.
7. The Gene Ontology Consortium. Gene ontology: tool for the unification of biology. *Nature Genetics*, 25(1):25–29, May 2000.
8. M. Deng, T. Chen, and F. Sun. An integrated probabilistic model for functional prediction of proteins. In *in Proc 7th Int Conf Comp Mol Biol*, pages 95–103, 2003.
9. A. Esuli, T. Fagni, and F. Sebastiani. Boosting multi-label hierarchical text categorization. *Information Retrieval*, 11:287–313, 2008.
10. A P Gasch, P T Spellman, C M Kao, O Carmel-Harel, M B Eisen, G Storz, D Botstein, and P O Brown. Genomic expression programs in the response of yeast cells to environmental changes. *Mol. Biol. Cell*, 11:4241–4257, 2000.
11. G. Jun and J. Ghosh. Multi-class Boosting with Class Hierarchies. In *Multiple Classifier Systems*, pages 32–41, 2009.
12. S. Mostafavi and Q. Morris. Using the gene ontology hierarchy when predicting gene function. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 22–26, Montreal, Canada, September 2009.
13. Indranil Palit and Chandan K. Reddy. Scalable and Parallel Boosting with Map-Reduce. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 2012. In press.
14. Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. Classifier Chains for Multi-label Classification. In *Proceedings of the 20th European Conference on Machine Learning (ECML 2009)*, Bled, Slovenia, 2009.
15. Chandan K. Reddy and Jin-Hyeong Park. Multi-resolution Boosting for Classification and Regression Problems. *Knowledge and Information Systems (KAIS)*, 29(2):435–456, November 2011.
16. J. Rousu, C. Saunders, S. Szedmak, and J. Shawe-Taylor. Kernel-Based Learning of Hierarchical Multilabel Classification Models. *The Journal of Machine Learning Research*, 7:1601–1626, 2006.
17. A Ruepp, A Zollner, D Maier, K Albermann, J Hani, M Mokrejs, I Tetko, U Güldener, G Mannhaupt, M Münsterktter, and HW Mewes. The FunCat, a functional annotation scheme for systematic classification of proteins from whole genomes. *Nucleic Acids Research*, 32(18):5539–5545, Oct 2004.
18. Carlos N. Silla, Jr. and Alex A. Freitas. A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery*, 22:31–72, January 2011.
19. C. Stark, B. Breitkreutz, T. Reguly, L. Boucher, A. Breitkreutz, and M. Tyers. BioGRID: a general repository for interaction datasets. *Nucleic Acids Research*, 34:D535–D539, 2006.
20. Giorgio Valentini. True path rule hierarchical ensembles for genome-wide gene function prediction. *IEEE ACM Transactions on Computational Biology and Bioinformatics*, 8(3):832–847, 2011.
21. C. Vens, J. Struyf, L. Schietgat, S. Džeroski, and H. Blockeel. Decision trees for hierarchical multi-label classification. *Machine Learning*, 73:185–214, 2008.
22. C. Von Mering, R. Krause, B. Snel, M. Cornell, S. Oliver, S. Fields, and P. Bork. Comparative assessment of large-scale data sets of protein-protein interactions. *Nature*, 417:399–403, 2002.
23. Rong Yan, Jelena Tesic, and John R. Smith. Model-Shared Subspace Boosting for Multi-label Classification. In *13th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD)*, pages 834–843, New York, NY, USA, 2007.
24. M.-L. Zhang and K. Zhang. Multi-label learning by exploiting label dependency. In *Proceedings of the 16th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'10)*, pages 999–1007, Washington D. C., USA, 2010.