Probabilistic Clustering for Hierarchical Multi-Label Classification of Protein Functions

Rodrigo C. Barros[1], Ricardo Cerri[1],
Alex A. Freitas[2], and André C. P. L. F. de Carvalho[1]

[1] Universidade de São Paulo, São Carlos-SP, Brazil
{rcbarros,cerri,andre}@icmc.usp.br
[2] University of Kent, Canterbury, Kent, UK
A.A.Freitas@kent.ac.uk

Abstract. Hierarchical Multi-Label Classification is a complex classification problem where the classes are hierarchically structured. This task is very common in protein function prediction, where each protein can have more than one function, which in turn can have more than one sub-function. In this paper, we propose a novel hierarchical multi-label classification algorithm for protein function prediction, namely HMC-PC. It is based on probabilistic clustering, and it makes use of cluster membership probabilities in order to generate the predicted class vector. We perform an extensive empirical analysis in which we compare our new approach to four different hierarchical multi-label classification algorithms, in protein function datasets structured both as trees and directed acyclic graphs. We show that HMC-PC achieves superior or comparable results compared to the state-of-the-art method for hierarchical multi-label classification.

Keywords: Hierarchical Multi-Label Classification; Protein Function Prediction; Probabilistic Clustering

1 Introduction

Classification is the well-known machine learning task whose goal is to assign instances to predefined categories. Classification algorithms are given as input a set of N n-dimensional training instances $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$, as well as their respective set of class labels $C = \{C_{\mathbf{x}_1}, C_{\mathbf{x}_2}, ..., C_{\mathbf{x}_N}\}$, in which $C_{\mathbf{x}_i} \in \{C_1, ...C_k\}$ in a k-class problem.

The vast majority of classification problems require the association of each instance with a single class, which means $C_{\mathbf{x}_i}$ is a single categorical value in $\{C_1, ...C_k\}$. This particular kind of problem is regarded as *flat* or *non-hierarchical* classification. Notwithstanding, there are problems in which the classes are organized in a hierarchical structure — a tree or a directed acyclic graph (DAG) — and each instance may be associated to multiple classes in multiple paths of this hierarchy. The difference between the tree and DAG hierarchies is that, in a

tree, each class can have only one superclass, which implies there is just one path between the root node and the class node. In a DAG hierarchy, each class can have more than one superclass at the same time, which means that there can be multiple paths from the root node to a class node. As an example, consider the dotted nodes in Fig. 1. While in Fig. 1(a) there is just one possible path between the root node and the dotted nodes (1/2 and 2/2/1) (tree structure), we can see that in Fig. 1(b) (DAG structure) we can reach the dotted node at the second level by two paths (1/2 and 2/2). The same can be observed at the dotted node located in the third level, which can be reached by two different paths (2/1/1 and 2/2/1).

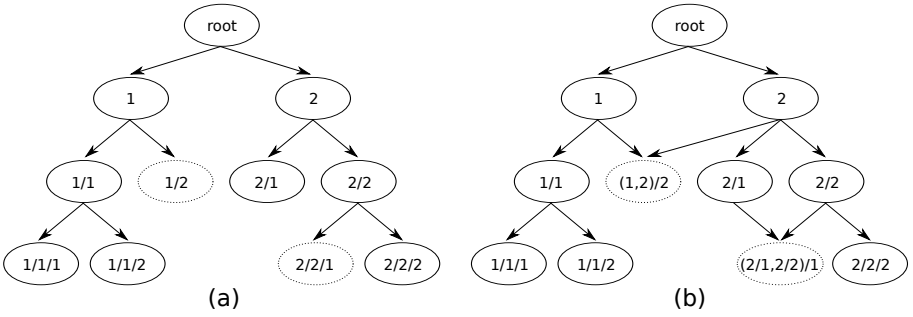

Fig. 1. Example of class hierarchy.

Either structured as a DAG or tree, this particularly complex problem is known as hierarchical multi-label classification (HMC), and is the primary subject of this paper. In a HMC problem, the set of class labels can be represented as a matrix $\mathbf{V} = \{\mathbf{v}_{\mathbf{x}_1}, \mathbf{v}_{\mathbf{x}_2}, ..., \mathbf{v}_{\mathbf{x}_N}\}$, in which $\mathbf{v}_{\mathbf{x}_i}$ is the c-dimensional binary class vector associated with instance $\mathbf{x}_i$, in a hierarchy with c nodes (classes). Each position of the class vector $\mathbf{v}_{\mathbf{x}_i}$ represents a class, and it is set to 1 if $\mathbf{x}_i$ belongs to that respective class, or 0 otherwise.

Two important examples of HMC problems are the tasks of text classification [29, 30, 25] and protein function prediction [4, 6, 31]. The latter is an increasingly important research field by itself, given the recent availability of unknown proteins for analysis and determination of their respective biological functions. Protein function prediction can be seen as a predictive problem in which each protein is a dataset instance, whereas different protein features are used as predictive attributes, and ultimately the goal is to classify these proteins according to different functions they can perform. Since a protein can perform multiple functions, and these functions are usually organized in a hierarchical structure (*e.g.*, the FunCat [26] and Gene Ontology [3] protein functional-definition schemes), the protein function prediction can be regarded as a typical HMC problem.

There has been an increasing number of machine learning approaches for hierarchical multi-label classification of protein functions. Roughly speaking, these

approaches can be divided into *local* and *global* approaches. In the local approach, conventional (flat) classification algorithms such as decision trees or support vector machines are trained to produce a hierarchy of classifiers, which are then executed following a top-down strategy to classify unlabeled instances [11]. In this approach, local information about the class hierarchy is used during the training process of each base classifier. Such local information can be used in different ways, depending on how the local classifiers are induced [28]. Differently from the local approach, the global approach induces a single classifier using all classes of the hierarchy at once. After the training process, the classification of a new instance occurs in just one step [31]. As global methods induce a single classifier to consider the specificities of the classification problem, they usually do not make use of conventional classification algorithms, unless these are heavily adapted to consider the hierarchy of classes.

In this paper, we propose a novel global hierarchical multi-label classification algorithm that is based on probabilistic clustering, namely Hierarchical Multi-label Classification with Probabilistic Clustering (HMC-PC). HMC-PC works according to the following assumption: instances that belong to a given cluster have similar class vectors, and hence the training instances are clustered following an expectation-maximization scheme [13], and the average class vector of the training instances from a given cluster is used to classify new unseen instances associated to the same cluster. The cluster membership probabilities are also used to tune the average class vector in each cluster. HMC-PC offers the advantages of the global methods, namely: (a) reduced time complexity when compared to the execution of multiple classifiers in the local approach; and (b) it does not suffer the problem of error propagation across levels, since the classification of a given hierarchical level is not done separately from the other levels. Finally, we show that HMC-PC presents competitive results when compared with the state of the art decision-tree-based method Clus-HMC [31].

This paper is organized as follows. Section 2 discusses related work on machine learning approaches for protein function prediction. Section 3 details HMC-PC, our novel global algorithm for hierarchical multi-label classification. Section 4 depicts the methodology employed for the experimental analysis of protein function prediction, which is in turn presented in Section 5. We present our conclusions and future work opportunities in Section 6.

2 Related Work

One of the first HMC algorithms was proposed by Clare and King [10], namely HMC4.5, which is a global method based on decision-tree induction algorithms. It is a variant of C4.5 [24] with modifications in the calculation of class entropy. The proposed modification uses the sum of the number of bits needed to describe membership or non-membership of each class, and also the information related to the size of the tree rooted by a given class. The method was used in tree-structured hierarchies.

In [18] and [19], the authors proposed a global method for the classification of Gene Ontology (GO) [3] genes based on the classification of documents from the MedLine repository that describe these genes. This method expands the sets of classes by including their ancestor classes and then applies the AdaBoost algorithm [27] in the modified dataset. Inconsistent predictions that may have occurred are corrected.

In [4], the authors proposed a local method that uses a hierarchy of SVM classifiers for the prediction of gene functions structured according to the GO. The classifiers are trained separately for each class, and the predictions are then combined using Bayesian Networks [16], aiming at finding the most probable consistent set of predictions.

In the work of Vens et al. [31], three methods based on the concept of Predictive Clustering Trees (PCT) were extensively compared. The authors make use of the global Clus-HMC method [5] that induces a single decision tree to cope with the entire hierarchical multi-label classification problem. They compared its performance with two local methods. The first one, Clus-SC, induces an independent decision tree for each class of the hierarchy, ignoring the relationships between classes. The second, Clus-HSC, explores the hierarchical relationships between the classes to induce a decision tree for each class. The authors employed the above-mentioned methods to hierarchies structured both as trees and DAGs, and discussed the modifications needed so that the algorithms could cope with both types of hierarchical structures. While in [31] the authors used the Euclidean distance to calculate the similarities and dissimilarities between instances in the decision tree, Aleksovski et al. [2] expanded this study by investigating the use of other distance measures, namely Jaccard, SimGIC, and ImageClef.

In [22], Otero et al. proposed *h*Ant-Miner, a global method for hierarchical single-label classification using Ant Colony Optimization (ACO) [15, 14]. The authors later extended this method [23] to allow multi-label classification, considering both tree- and DAG-structured hierarchies.

Cerri et al. [8] proposed a global method that employs a Genetic Algorithm (GA) to produce HMC rules. The GA evolves the antecedents of classification rules, in order to optimize the level of coverage of each antecedent. The employed fitness (evaluation) function gives a better reward to rules with the antecedents that cover a higher number of instances. Then, the set of optimized antecedents is selected to build the corresponding consequent of the rules (set of classes to be predicted). The method was used in hierarchies structured as trees.

Both in [21] and [1], the authors propose methods that employ clustering as a substep of classification, though these approaches only deal with flat multi-label data and not with hierarchical multi-label data.

Finally, Cerri et al. [7, 9] proposed a local approach that employs a sequence of connected artificial neural networks for protein function prediction. Each network is associated to a hierarchical level, and the output of the network in level l is used as the input of the network in level $l + 1$. A strategy for avoiding inconsistent predictions is employed, since a given neural network may predict a

class whose superclass had not been predicted before. The method is tested over a hierarchy structured as a tree.

3 HMC-PC

In this paper, we propose Hierarchical Multi-label Classification with Probabilistic Clustering (HMC-PC), which is a novel global HMC algorithm. The general rationale behind HMC-PC is the assumption that we can discover the most fitting probability distribution for each particular group of training instances, and that those instances that were generated by the same distribution also share similar class vectors. Hence, once we have discovered the set of k distributions from the training set, a new unseen instance can be easily classified by performing two procedures: (i) discovering the distribution that most probably has generated it — *i.e.*, discovering which cluster it belongs to; and (ii) assigning to this new instance the average class vector of the training instances that were generated by the same distribution (cluster).

HMC-PC can roughly be divided into three mains steps:(i) cluster generation; (ii) class vector generation; and (iii) classification.

1. Cluster generation: the training dataset is arranged into different clusters following a probabilistic expectation-maximization (EM) scheme [13];
2. Class vector generation: for each cluster, the class vectors of the training instances that surpass a given probability threshold are averaged, and later used to classify unseen instances;
3. Classification: each test instance is assigned to the cluster it most probably belongs to. Then, the cluster's average class vector generated in the previous step is assigned to the test instance as the final prediction.

3.1 Cluster Generation

The first step of HMC-PC is to generate clusters from the training dataset $\mathbf{X}$, which is comprised of n attributes and N instances. In this step, the class vector of each instance $\mathbf{x}_i \in \mathbf{X}$, $i = 1...N$ is not used during cluster generation.

Each cluster in HMC-PC is assumed to be generated by a distinct Gaussian probability distribution. HMC-PC clustering iterates over the steps of expectation and maximization, much the same as the well-known EM algorithm [13]. We make the further naïve assumption of attribute independence, which means we are only interested in the diagonal of the covariance matrix $\mathbf{\Sigma}_i$ from the i^{th} Gaussian. This assumption is intended to speed-up the algorithm, avoiding the cost of computing the inverse of $\mathbf{\Sigma}_i$, which is usually $O(n^3)$.

In the expectation step, the cluster membership of each instance $\mathbf{x}_i$ regarding each Gaussian distribution (cluster) C_j is computed, assuming the parameters of each of the k distributions are already known:

$$Pr[C_j|\mathbf{x}_i] = \frac{Pr[\mathbf{x}_i|C_j] \times Pr[C_j]}{Pr[\mathbf{x}_i]} \quad (1)$$

where $Pr[\mathbf{x}_i|C_j] = \mathcal{N}(\mathbf{x}_i|\mu_j, \mathbf{\Sigma}_j)$, and $Pr[C_j]$ is estimated as $\sum_{i=1}^{N} Pr[C_j|\mathbf{x}_i]/N$. Note that $Pr[\mathbf{x}_i]$ can simply be replaced by the sum of $Pr[\mathbf{x_i}|C_j] \times Pr[C_j]$ for the k distributions.

The maximization step is performed by simply updating the parameters of the k distributions, taking into account the recently computed values of $Pr[C|\mathbf{x}]$:

$$N_j = \sum_{i=1}^{N} Pr[C_j|\mathbf{x}_i] \tag{2}$$

$$Pr[C_j] = \frac{N_j}{N} \tag{3}$$

$$\mu_j = \frac{1}{N_j} \sum_{i=1}^{N} Pr[C_j|\mathbf{x}_i] \times \mathbf{x}_i \tag{4}$$

$$\mathbf{\Sigma}_j = \frac{1}{N_j} \sum_{i=1}^{N} Pr[C_j|\mathbf{x}_i] \times (\mathbf{x}_i - \mu_j)(\mathbf{x}_i - \mu_j)^T \tag{5}$$

The iteration between the expectation and maximization steps is performed until one of the two conditions occurs:

- the maximum number of 100 iterations is reached; or
- the difference between the log-likelihood of two consecutive steps is smaller than 1×10^{-6}.

The log-likelihood is computed after each expectation step:

$$\mathcal{LL} = \sum_{i=1}^{N} \ln \left(\sum_{j=1}^{k} Pr[C_j] \times Pr[\mathbf{x_i}|C_j] \right) \tag{6}$$

Since we have to assume that either the cluster memberships $Pr[C_j|\mathbf{x}_i]$ or the distribution parameters $\mu_j, \mathbf{\Sigma}_j$ are informed before the beginning of the expectation-maximization iterations, HMC-PC executes the well-known k-means algorithm [20] 10 times varying the random initialization. The partition with the smallest value of the squared error is employed to initialize the parameters of the k distributions.

The only problem that remains is the definition of the number of Gaussian distributions (clusters). In order to avoid the use of a user-defined parameter, we propose the following methodology for automatically defining the value of k:

1. Set $k = 1$;
2. Run the expectation-maximization iterations with 10-fold cross-validation over the *training set*, *i.e.*, in each run EM is applied to 9 of the 10 training folds and the log-likelihood is assessed on the hold-out fold, averaging the results over the 10 runs.
3. If the log-likelihood has increased, increase the value of k by 1 and the procedure continues in step 2.

After the algorithm has converged and the final parameters of the k clusters are known, the N training instances are assigned to their most probable cluster, *i.e.*:

$$C_{\mathbf{x}_i} = \underset{C_j}{\operatorname{argmax}} \left(\frac{Pr[\mathbf{x}_i|C_j] \times Pr[G_j]}{Pr[\mathbf{x}_i]} \right) \tag{7}$$

3.2 Class Vector Generation

Once the training instances have been distributed throughout the clusters, HMC-PC generates one class vector per cluster. The rationale behind this step is that a future test instance will be assigned to its most probable cluster according to Eq. (7), and then classified according to the class vector generated from that cluster. HMC-PC offers two strategies to generate one class vector per cluster:

1. The class vector of cluster C_j is generated as the average class vector of the training instances that were assigned to cluster C_j, *i.e.*:

$$\bar{\mathbf{v}}_{C_j} = \frac{1}{N} \sum_{\mathbf{x}_i \in C_j} \mathbf{v}_{\mathbf{x}_i} \tag{8}$$

2. The class vector of cluster C_j is generated as the average class vector of the training instances that were assigned to cluster C_j and whose cluster membership probability surpass a given previously-defined threshold Δ_j, *i.e.*:

$$\bar{\mathbf{v}}_{C_j} = \frac{1}{N} \sum_{\substack{\mathbf{x}_i \in C_j \wedge \\ Pr[C_j|\mathbf{x}_i] \geq \Delta_j}} \mathbf{v}_{\mathbf{x}_i} \tag{9}$$

Note that strategy 1 is a special case of strategy 2 in which $\Delta = 0$ for all clusters. The second strategy, on the other hand, makes use of the cluster memberships to define the average class vectors. The disadvantage of the second strategy is the need of defining threshold values Δ for each cluster. In order to overcome this problem, we propose an adaptive threshold selection strategy as follows. First, the training set is divided into two subsets: sub-training and validation. The sub-training set is used as before to generate the clusters, and its instances are distributed throughout the discovered clusters. Next, we also distribute the validation instances to their most probable cluster, also according to Eq. (7). Then, for each cluster, we evaluate the classification performance of the validation instances with the area under the precision-recall curve (AUPRC, more details in Section 4) by building the average class vector following Eq. (9). For that, we have to try different values of Δ_j, *i.e.*, $\{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$. The average class vector built according to the threshold value that yielded the largest AUPRC value is then chosen to classify the test instances that are assigned to cluster C_j.

The pseudo-code of HMC-PC with the adaptive threshold selection strategy is presented in Alg. 1. The main difference between the adaptive threshold

Algorithm 1 HMC-PC with adaptive threshold selection.

Require: `Training dataset` $\mathbf{X}$
Require: `Threshold set` $ts = \{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$
1: Divide $\mathbf{X}$ in sub-training $\mathbf{X}^t$ and validation $\mathbf{X}^v$ sets
2: $k \leftarrow CV(\mathbf{X}^t)$
3: $partition \leftarrow EM(\mathbf{X}^t, k)$
4: **for** $\mathbf{x}_i \in X^t$ **do**
5: $\quad C_{\mathbf{x}_i} \leftarrow$ Eq. (7)
6: **end for**
7: **for** $\mathbf{x}_i \in X^v$ **do**
8: $\quad C_{\mathbf{x}_i} \leftarrow$ Eq. (7)
9: **end for**
10: **for** cluster $C_j \in partition$ **do**
11: $\quad bestAUPRC \leftarrow 0$
12: $\quad$ **for** $\Delta_j \in ts$ **do**
13: $\quad\quad \bar{\mathbf{v}}_{C_j} \leftarrow$ Eq. (9)
14: $\quad\quad AUPRC \leftarrow classify(\{\mathbf{x^v}_i | \mathbf{x^v}_i \in C_j\}, \bar{\mathbf{v}}_{C_j})$
15: $\quad\quad$ **if** $AUPRC \geq bestAUPRC$ **then**
16: $\quad\quad\quad bestAUPRC \leftarrow AUPRC$
17: $\quad\quad\quad \mathbf{thresholds}_j \leftarrow \Delta_j$
18: $\quad\quad$ **end if**
19: $\quad$ **end for**
20: **end for**
21: $partition \leftarrow EM(\mathbf{X}^t \cup \mathbf{X}^v, k)$
22: **return** $\mathbf{thresholds}, partition$

strategy and the threshold-free strategy is the need for a validation set to automatically select the value of Δ_j for cluster C_j. Note that both training and validation data are distributed throughout the clusters (lines 5 and 8). The main loop in line 10 performs the adaptive threshold selection by evaluating the validation data that were assigned to a given cluster C_j with regard to the different threshold values. The algorithm stores in vector **thresholds** the optimized threshold value per cluster. These thresholds were the ones that maximized the AUPRC generated by Eq. (9) (line 13). Therefore, even with the threshold Δ possibly assuming different values for each cluster, the user is not required to set any ad-hoc parameter during the whole execution of HMC-PC. Finally, the method performs the expectation-maximization steps once again (line 21) with the full training set (sub-training + validation).

3.3 Classification

The last step of HMC-PC is to classify unseen instances. The classification process is straightforward: (i) assign each test instance to its most probable cluster according to Eq. (7); (ii) assuming test instance $\mathbf{x}_i$ was assigned to cluster C_j, make use of class vector $\bar{\mathbf{v}}_{C_j}$ computed from the training instances that belong to cluster C_j and have cluster membership probability greater than $\mathbf{thresholds}_j$ as the class prediction for test instance $\mathbf{x}_i$.

4 Experimental Methodology

4.1 Baseline Algorithms

We employ four of the methods reviewed in Section 2 as the baseline algorithms during the experiments performed in this work. We make use of the global decision-tree based method Clus-HMC, which is considered the state-of-the-art method in the literature, since it obtained the best results so far, and we also make use of its local variants Clus-HSC and Clus-SC. The three methods are detailed in [31]. We also employ the Ant Colony Optimization-based method *hm*Ant-Miner [23], which is a global method that obtained competitive results when compared to Clus-HMC. We decided to select these algorithms because they were all applied to the same protein function prediction datasets used in the experiments (both for tree and DAG structures). In addition, they all produce the same type of output provided by HMC-PC, and their performance were analyzed with the same evaluation measure we use in this paper.

We evaluated HMC-PC's performance with the two alternative class-vector generation mechanisms presented in Section 3.2. The first version will be regarded as HMC-PC whereas the second version will be regarded as HMC-PC$_{\Delta}$.

4.2 Datasets

Ten freely-available numeric datasets[3] related to protein function prediction are used in the experiments, namely: cellcycle, derisi, eisen, gasch1, and gasch2 (FunCat-annotated and Gene Ontology-annotated). The option for all-numeric datasets is because the current version of HMC-PC can only cope with numeric attributes. Dealing with nominal attributes is a topic left for future work.

These datasets are related to issues like phenotype data and gene expression levels. They are organized according to two different class hierarchy structures: tree structure (FunCat-annotated data sets) and directed acyclic graph structure (Gene Ontology-annotated data sets).

Table 1 summarizes the main characteristics of the training, validation, and test datasets employed in the experiments. In the particular case of *hm*Ant-Miner and HMC-PC, the training and validation datasets are merged and used together to generate the predictive models. The PCT-based methods and HMC-PC$_{\Delta}$ make use of the validation datasets to optimize parameters during their executions.

A detailed description of each dataset can be found in [31]. For executing HMC-PC in these datasets, all missing values were replaced with the mean value of the respective attribute.

4.3 Evaluation Measures and Statistical Analysis

Considering that all algorithms tested in this paper output a vector of class probabilities for each instance being predicted, we make use of the area under the average PR-curve ($AU(\overline{PRC})$) as the evaluation measure to compare them.

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

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

Structure	Dataset	$\|A\|$	$\|C\|$	Training		Validation		Test	
				Total	Multi	Total	Multi	Total	Multi
Tree	Cellcycle	77	499	1628	1323	848	673	1281	1059
	Derisi	61	499	1608	1309	842	671	1275	1055
	Eisen	79	461	1058	900	529	441	837	719
	Gasch1	173	499	1634	1325	846	672	1284	1059
	Gasch2	52	499	1639	1328	849	674	1291	1064
DAG	Cellcycle	77	4125	1625	1625	848	848	1278	1278
	Derisi	61	4119	1605	1605	842	842	1272	1272
	Eisen	79	3573	1055	1055	528	528	835	835
	Gasch1	173	4125	1631	1631	846	846	1281	1281
	Gasch2	52	4131	1636	1636	849	849	1288	1288

To obtain a PR-curve for a given algorithm, different thresholds ranging within $[0, 1]$ are applied to the outputs of the methods, and thus different values of precision and recall are obtained, one for each threshold value. Each threshold value then represents a point within the PR space. The union of these points forms a PR-curve. In order to calculate the area below the PR-curve, the PR-points must be interpolated [12]. This interpolation guarantees that the area below the curve is not artificially increased, which would happen if the curves were constructed just connecting the points without interpolation. Given a threshold value, a precision-recall point $(\overline{Prec}, \overline{Rec})$ in the PR-space can be obtained through Eq. (10) and (11), corresponding to the micro-average of precision and recall, where i ranges from 1 to c, and TP, FP, and FN stand, respectively, for the number of true positives, false positives, and false negatives.

$$\overline{Prec} = \frac{\sum_i TP_i}{\sum_i TP_i + \sum_i FP_i} \quad (10) \qquad \overline{Rec} = \frac{\sum_i TP_i}{\sum_i TP_i + \sum_i FN_i} \quad (11)$$

In order to provide some reassurance about the validity and non-randomness of the results, we employed the Friedman and Holm statistical tests, recommended for comparisons when a control classifier is compared against other classifiers [17]. We employed a confidence level of 95% in the statistical tests.

5 Results and Discussion

Table 2 presents the comparison among the two HMC-PC versions and the baseline methods Clus-HMC, Clus-HSC, Clus-SC, and *hm*Ant-Miner. Given that *hm*Ant-Miner is a non-deterministic method, its results are averages over 15 executions. Both versions of HMC-PC and the PCT-based methods are deterministic algorithms and thus require a single execution. We highlight in bold the best absolute values for each dataset, and we provide at the end of the table the average rank of each method, following the Friedman statistical test.

Table 2. $AU(\overline{PRC})$ values for the comparison of HMC-PC with *hm*Ant-Miner, Clus-HMC, Clus-HSC, Clus-SC.

		HMC-PC	HMC-PC$_\Delta$	Clus-HMC	Clus-HSC	Clus-SC	*hm*-Ant-Miner
DAG							
	Cellcycle	0.368	0.369	0.357	**0.371**	0.252	0.332
	Derisi	0.341	0.344	**0.355**	0.349	0.218	0.334
	Eisen	0.396	**0.398**	0.380	0.365	0.270	0.376
	Gasch1	0.381	**0.382**	0.371	0.351	0.239	0.356
	Gasch2	0.369	0.367	0.365	**0.378**	0.267	0.344
Tree							
	Cellcycle	**0.200**	0.187	0.172	0.111	0.106	0.154
	Derisi	0.163	0.163	**0.175**	0.094	0.089	0.161
	Eisen	0.211	**0.214**	0.204	0.127	0.132	0.180
	Gasch1	**0.212**	0.210	0.205	0.106	0.104	0.175
	Gasch2	0.196	0.**197**	0.195	0.121	0.119	0.152
	Average Rank	2.15	1.85	2.80	4.00	5.90	4.30

We can observe that both HMC-PC versions are the best-ranked among all methods. It is interesting to see that the threshold-based version HMC-PC$_\Delta$, which takes into consideration the cluster membership probability to build the average class vectors, is slightly better ranked than HMC-PC. This is coherent with our initial hypothesis that the cluster membership probabilities can be used to tune the cluster class vector generation process, improving the prediction of unseen instances. Indeed, by making use of the cluster memberships, HMC-PC$_\Delta$ was capable of detecting the training instances that could bring more precise information within each cluster.

Regardless of the HMC-PC version employed, we can see that it provides better results than *hm*-Ant-Miner for all ten datasets. The same can be said regarding Clus-SC, which is outperformed by either version of HMC-PC by a large margin.

In the comparison against Clus-HSC, we can notice that HMC-PC and HMC-PC$_\Delta$ outperform it in 7 out of 10 datasets, and they are outperformed by it in the remaining three. It should be noticed that the performance of Clus-HSC in the datasets structured as a tree is very poor, which seems to be a problem that both local PCT-based methods share.

Finally, when comparing HMC-PC and HMC-PC$_\Delta$ with Clus-HMC, we can see that both versions outperform Clus-HMC in 8 out of 10 datasets, being outperformed in only two datasets, which is quite a considerable difference considering the fact that Clus-HMC is the best-performing method in the literature.

For assessing the statistical significance of the results, we first consider the p-value provided by the Friedman test: 3.15×10^{-6}, which states that the null hypotheses in which all methods perform similarly should be rejected. Then, we take the best-ranked method as the *control* algorithm, and a set of pairwise adjusted comparisons according to Holm's procedure are performed.

Table 3 presents the p-values and adjusted α values for the Holm pos-hoc pairwise comparisons, bearing in mind that HMC-PC$_\Delta$ is the control algorithm. The statistical test rejects those hypotheses that have a p-value ≤ 0.025. Note

Table 3. Holm's procedure for $\alpha = 0.05$. HMC-PC$_\Delta$ is the control algorithm.

i	Algorithm	$z = (R_0 - R_i)/SE$	p-value	Holm's adjusted α
5	Clus-SC	4.84	1.29×10^{-6}	0.0100
4	hm-Ant-Miner	2.93	3.40×10^{-3}	0.0125
3	Clus-HSC	2.57	1.02×10^{-2}	0.0167
2	Clus-HMC	1.14	2.56×10^{-1}	0.0250
1	HMC-PC	0.36	0.72×10^{-1}	0.0500

that HMC-PC$_\Delta$ outperforms with statistical significance all methods but Clus-HMC and HMC-PC.

Since Clus-HMC is the best-performing baseline algorithm, we now compare it with HMC-PC$_\Delta$ in specific classes of the hierarchy in order to examine their behavior when predicting classes at different hierarchical levels. For selecting these specific classes, we used the following methodology: we selected ten classes from each dataset in which Clus-HMC presented the best per-class AUPRC values in the training set. We compare the test per-class AUPRC values between HMC-PC$_\Delta$ and Clus-HMC in the DAG-structured datasets and in the tree-structured datasets (Table 4).

By careful inspection of Table 4, we can observe that in the datasets in which HMC-PC$_\Delta$ outperformed Clus-HMC in $AU(\overline{PRC})$, it also outperformed Clus-HMC in the majority of the classes regarding the per-class AUPRC. The only exception was the Gasch2 dataset organized as a DAG, in which Clus-HMC and HMC-PC$_\Delta$ tied 5-5 in the 10 selected classes. Overall, HMC-PC$_\Delta$'s good performance is consistent across hierarchical levels.

For exemplifying this scenario, we can notice that HMC-PC outperformed Clus-HMC in several classes that lie deep in the hierarchy. Recall that these classes are associated with more specific protein functions, and the more specific the function, the more useful the information about the protein. Also, recall that the deeper the class, the fewer the number of instances assigned to it. As an example, we can cite the case of the GO term (class) `GO:0006412` in datasets Cellcycle, Eisen, Gasch1, and Gasch2 (GO-annotated), in which HMC-PC$_\Delta$ consistently outperforms Clus-HMC. Figure 2 shows how deep in the DAG-structured hierarchy the GO term `GO:0006412` lies.

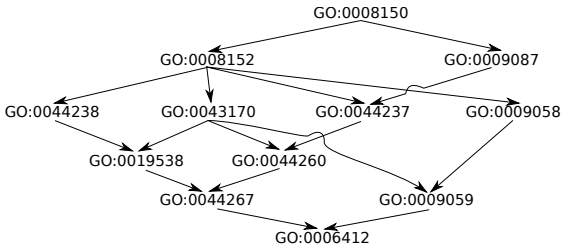

Fig. 2. Hierarchical paths that lead to term `GO:0006412`.

Table 4. Per-class AUPRC values for 10 specific classes in each dataset. Classes that start with "GO:" belong to DAG-structured datasets, whereas the remaining belong to tree-structured datasets.

Dataset	Class	Clus-HMC	HMC-PC$_\Delta$	Class	Clus-HMC	HMC-PC$_\Delta$
Cellcycle	GO:0044464	**0.967**	0.961	GO:0044424	0.898	**0.921**
	GO:0009987	0.860	**0.876**	GO:0008152	0.726	**0.767**
	GO:0003735	0.404	**0.649**	GO:0044237	0.676	**0.699**
	GO:0044238	0.650	**0.677**	GO:0044444	0.586	**0.624**
	GO:0043170	0.580	**0.632**	GO:0006412	0.361	**0.599**
Eisen	GO:0044464	0.981	**0.983**	GO:0044424	0.926	**0.929**
	GO:0009987	**0.916**	0.909	GO:0003735	0.692	**0.713**
	GO:0008152	0.808	**0.840**	GO:0044445	0.636	**0.687**
	GO:0006412	0.607	**0.652**	GO:0044237	0.751	**0.770**
	GO:0044238	0.716	**0.726**	GO:0043170	0.664	**0.720**
Derisi	GO:0044464	0.965	**0.974**	GO:0044424	**0.888**	0.880
	GO:0009987	**0.849**	0.838	GO:0008152	**0.736**	0.731
	GO:0044237	**0.674**	0.668	GO:0044238	**0.643**	0.640
	GO:0044444	**0.582**	0.555	GO:0003735	**0.462**	0.293
	GO:0043170	**0.585**	0.556	GO:0043226	0.525	**0.540**
Gasch1	GO:0044464	0.963	**0.978**	GO:0044424	0.912	**0.927**
	GO:0009987	0.855	**0.875**	GO:0003735	**0.659**	0.614
	GO:0008152	0.733	**0.754**	GO:0044237	0.674	**0.699**
	GO:0006412	0.574	**0.583**	GO:0044238	0.660	**0.671**
	GO:0044444	**0.647**	0.639	GO:0043170	0.616	**0.659**
Gasch2	GO:0044464	**0.966**	0.961	GO:0044424	0.910	**0.926**
	GO:0009987	**0.863**	0.850	GO:0003735	**0.622**	0.609
	GO:0008152	**0.741**	0.739	GO:0044237	**0.696**	0.682
	GO:0006412	0.521	**0.536**	GO:0044238	0.667	**0.668**
	GO:0043170	0.655	**0.669**	GO:0044422	0.601	**0.619**
Cellcycle	1	0.402	**0.432**	12.01	0.330	**0.425**
	10	0.335	**0.349**	12.01.01	0.326	**0.351**
	10.01	0.195	**0.200**	14	0.303	**0.329**
	11	**0.371**	0.351	16	0.261	**0.276**
	12	0.304	**0.424**	20	0.252	**0.254**
Eisen	1	0.392	**0.462**	12.01	0.582	**0.650**
	2	**0.420**	0.301	12.01.01	0.609	**0.722**
	10	0.352	**0.364**	14	0.396	**0.370**
	11	**0.394**	0.369	14.13.01	**0.177**	0.082
	12	0.508	**0.650**	16	**0.270**	0.250
Derisi	1	0.376	**0.400**	12.01	**0.376**	0.235
	2	**0.238**	0.215	12.01.01	**0.385**	0.234
	10	0.240	**0.278**	14	**0.280**	0.278
	11	0.264	**0.337**	16	**0.237**	0.232
	12	**0.366**	0.259	20	**0.259**	0.248
Gasch1	1	0.444	**0.454**	12.01	**0.635**	0.590
	2	**0.285**	0.190	12.01.01	**0.660**	0.593
	10	0.326	**0.367**	14	0.329	**0.362**
	11	0.363	**0.439**	16	**0.279**	0.259
	12	0.566	**0.587**	20	0.300	**0.302**
Gasch2	1	0.451	**0.470**	12.01.01	**0.627**	0.478
	10	0.286	**0.290**	14	0.342	**0.352**
	11	**0.409**	0.400	16	0.248	**0.283**
	12	**0.562**	0.546	20	0.253	**0.261**
	12.01	**0.634**	0.525	42	0.236	**0.251**

We conclude by stating that HMC-PC seems to be a good alternative to Clus-HMC for the following reasons: (i) it performs better in the majority of the datasets; (ii) it is a parameter-free algorithm; (iii) it provides good AUPRC values for both shallow and deep classes in the hierarchy; and (iv) its time complexity is linear in all its input variables.

6 Conclusions

In this paper, we present a novel global hierarchical multi-label classification algorithm based on probabilistic clustering for the task of protein function prediction. The method is named Hierarchical Multi-Label Classification with Probabilistic Clustering (HMC-PC). We present two different versions of HMC-PC, namely HMC-PC and HMC-PC$_{\Delta}$.

HMC-PC works by clustering the protein function datasets in k clusters following an expectation-maximization scheme [13]. Then, for each of the k clusters, the average class vector is generated based on the training instances that were (hard-)assigned to each cluster. The choice of which instances will be used to define the per-cluster average class vector is based on the probabilities of cluster membership. The threshold-free version of HMC-PC assumes all instances that were assigned to a given cluster should be used for generating the cluster's average class vector, whereas HMC-PC$_{\Delta}$ employs an adaptive threshold selection strategy based on validation data to select the best value for Δ in each cluster.

We performed experiments using ten protein function prediction datasets (five of them structured as trees and five of them structured as DAGs). We compared HMC-PC versions with four well-known HMC algorithms: two decision tree-based local methods, namely Clus-HSC and Clus-SC; one decision tree-based global method, namely Clus-HMC; and one global method based on Ant Colony Optimization, namely *hm*Ant-Miner. Among all the methods previously proposed in the literature, Clus-HMC has been considered so far the state-of-the-art method for hierarchical multi-label classification [31]. We evaluated the methods using the area under the average PR-curve ($AU(\overline{PRC})$).

The comparison with the baseline methods shows that HMC-PC — particularly its threshold-based version HMC-PC$_{\Delta}$ — outperforms them in the majority of the datasets. We also compared HMC-PC$_{\Delta}$ and Clus-HMC in individual hierarchical classes, and showed that HMC-PC$_{\Delta}$ often obtained the best performance, including in classes that lie deep in the class hierarchy. This is particularly important, since deep class predictions tend to be more useful to biololgists than shallow class predictions.

As future work, we intend to extend HMC-PC so it can also deal with categorical attributes. We also intend to perform a deeper analysis to verify whether the thresholds in the different clusters are correlated to each other in any sense. Finally, we plan to investigate the use of the complete covariation matrix, so we do not have to make the naïve assumption of attribute independence. Nevertheless, such a modification will lead to an increased time complexity, since the

complexity of finding the inverse of a $n \times n$ matrix is $O(n^3)$, whereas the current version of HMC-PC is linear in all its input variables.

Acknowledgment

The authors would like to thank Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), Brazil, for funding this research.

References

1. Ahmed, M.S.: Clustering guided multi-label text classification. Ph.D. thesis, University of Texas at Dallas (2012)
2. Aleksovski, D., Kocev, D., Dzeroski, S.: Evaluation of distance measures for hierarchical multilabel classification in functional genomics. In: 1st Workshop on Learning from Multi-Label Data (MLD) held in conjunction with ECML/PKDD. pp. 5–16 (2009)
3. Ashburner, M., et al.: Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. Nature Genetics 25, 25–29 (2000)
4. Barutcuoglu, Z., Schapire, R.E., Troyanskaya, O.G.: Hierarchical multi-label prediction of gene function. Bioinformatics 22, 830–836 (2006)
5. Blockeel, H., Bruynooghe, M., Dzeroski, S., Ramon, J., Struyf, J.: Hierarchical multi-classification. In: Workshop on Multi-Relational Data Mining. pp. 21–35 (2002)
6. Blockeel, H., Schietgat, L., Struyf, J., Dzeroski, S., Clare, A.: Decision trees for hierarchical multilabel classification: A case study in functional genomics. In: Knowledge Discovery in Databases. pp. 18–29 (2006)
7. Cerri, R., Barros, R.C., Carvalho, A.C.P.L.F.: Hierarchical multi-label classification for protein function prediction: A local approach based on neural networks. In: Intelligent Systems Design and Applications (ISDA). pp. 337 –343 (nov 2011)
8. Cerri, R., Barros, R.C., Carvalho, A.C.P.L.F.: A genetic algorithm for hierarchical multi-label classification. In: Proceedings of the 27th Annual ACM Symposium on Applied Computing. pp. 250–255. SAC '12, ACM, New York, NY, USA (2012)
9. Cerri, R., Barros, R.C., Carvalho, A.C.P.L.F.: Hierarchical multi-label classification using local neural networks. Journal of Computer and System Sciences. In press. (2013)
10. Clare, A., King, R.D.: Predicting gene function in saccharomyces cerevisiae. Bioinformatics 19, 42–49 (2003)
11. Costa, E.P., Lorena, A.C., Carvalho, A.C.P.L.F., Freitas, A.A.: Comparing several approaches for hierarchical classification of proteins with decision trees. In: II Brazilian Symposium on Bioinformatics. Lecture Notes in Bioinformatics, vol. 4643, pp. 126–137. Springer-Verlag, Berlin, Heidelberg (2007)
12. Davis, J., Goadrich, M.: The relationship between precision-recall and roc curves. In: International Conference on Machine Learning. pp. 233–240 (2006)
13. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum Likelihood from Incomplete Data via the EM Algorithm. Journal of the Royal Statistical Society 39(1), 1–38 (1977)
14. Dorigo, M.: Optimization, Learning and Natural Algorithms. Ph.D. thesis, Dipartimento di Elettronica, Politecnico di Milano, IT (1992)

15. Dorigo, M., Maniezzo, V., Colorni, A.: Positive feedback as a search strategy. Tech. rep., Dipartimento di Elettronica, Politecnico di Milano, IT (1991)
16. Friedman, N., Geiger, D., Goldszmidt, M.: Bayesian network classifiers. Machine Learning 29(2-3), 131–163 (1997)
17. García, S., Fernández, A., Luengo, J., Herrera, F.: Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. Information Sciences 180(10), 2044–2064 (May 2010)
18. Kiritchenko, S., Matwin, S., Famili, A.: 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)
19. Kiritchenko, S., Matwin, S., Nock, R., Famili, A.: Learning and evaluation in the presence of class hierarchies: Application to text categorization. In: Lamontagne, L., Marchand, M. (eds.) Advances in Artificial Intelligence, Lecture Notes in Computer Science, vol. 4013, pp. 395–406. Springer Berlin Heidelberg (2006)
20. Lloyd, S.P.: Least squares quantization in pcm. IEEE Transactions on Information Theory 28(2), 129–137 (1982)
21. Nasierding, G., Tsoumakas, G., Kouzani, A.Z.: Clustering based multi-label classification for image annotation and retrieval. In: IEEE International Conference on Systems, Man and Cybernetics. pp. 4514–4519 (2009)
22. Otero, F.E.B., Freitas, A.A., Johnson, C.: A hierarchical classification ant colony algorithm for predicting gene ontology terms. In: European Conference on Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics. Lecture Notes in Computer Science, vol. 5483, pp. 68–79. Springer (2009)
23. Otero, F.E.B., Freitas, A.A., Johnson, C.: A hierarchical multi-label classification ant colony algorithm for protein function prediction. Memetic Computing 2, 165–181 (2010)
24. Quinlan, J.R.: C4.5: programs for machine learning. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1993)
25. Rousu, J., Saunders, C., Szedmak, S., Shawe-Taylor, J.: Kernel-based learning of hierarchical multilabel classification models. Journal of Machine Learning Research 7, 1601–1626 (2006)
26. Ruepp, A., Zollner, A., Maier, D., Albermann, K., Hani, J., Mokrejs, M., Tetko, I., Güldener, U., Mannhaupt, G., Münsterkötter, M., Mewes, H.W.: The funcat, a functional annotation scheme for systematic classification of proteins from whole genomes. Nucleic Acids Research 32(18), 5539–5545 (October 2004)
27. Schapire, R.E., Singer, Y.: Improved boosting algorithms using confidence-rated predictions. In: Machine Learning. vol. 37, pp. 297–336. Kluwer Academic Publishers, Hingham, MA, USA (1999)
28. Silla, C., Freitas, A.A.: A survey of hierarchical classification across different application domains. Data Mining and Knowledge Discovery 22, 31–72 (2011)
29. Sun, A., Lim, E.P.: Hierarchical text classification and evaluation. In: Fourth IEEE International Conference on Data Mining. pp. 521–528 (2001)
30. Sun, A., Lim, E.P., Ng, W.K., Srivastava, J.: Blocking Reduction Strategies in Hierarchical Text Classification. IEEE Transactions on Knowledge and Data Engineering 16, 1305–1308 (2004)
31. Vens, C., Struyf, J., Schietgat, L., Džeroski, S., Blockeel, H.: Decision trees for hierarchical multi-label classification. Machine Learning 73, 185–214 (2008)