An Adaptation of Binary Relevance for Multi-Label Classification applied to Functional Genomics

Erica Akemi Tanaka[1] **and José Augusto Baranauskas**[1]

[1]Faculdade de Filosofia, Ciências e Letras de Ribeirão Preto,
Universidade de São Paulo, Brasil

Abstract. *Many classification problems, especially in the field of bioinformatics are associated with more than one class, known as multi-label classification problems. In this study we propose a new adaptation for the Binary Relevance method taking into account the correlation among labels, focusing on the interpretability of the model, not only its performance. The experimental results shown that our proposal has a performance comparable to other methods as the same time it provides an interpretable model from the multi-label problem.*

1. Introduction

Since the advance of hardware and software, the automated sequencing of DNA fragments became possible. The amount of biological data available is increasing, which increases also the need for computational tools for processing and knowledge extraction. As a result, machine learning techniques are widely used to predict gene functions; then the best predictions can be tested in the lab to validate these results [Schietgat et al. 2010]. However, the prediction of gene function is complex considering the fact a single gene can have multiple functions. In this case, multi-label classification seems to be appropriated.

There are several reasons to investigate and propose new multi-label classification techniques, especially in bioinformatics or bio-related research fields. For example, the *Gene Ontology*[1] is an example of a multi-label problem, in which genes and proteins may have more than one function or feature. Another example is the MIPS Functional Catalogue [Ruepp et al. 2004], in which genes and proteins can belong to more than one functional class. Therefore, the research and development of computational techniques to classify multi-label problems using proteins, genes and other biological and medical data is very important: with this knowledge one can develop new drugs, treat diseases, help in diagnostics and among other purposes.

However, traditional algorithms are unable to handle a set of multi-label instances, since these algorithms were designed to predict a single label. A simpler solution is to transform the original dataset into a set of instances in which each set contains all attributes, and only one of the labels to be predicted, the method known as *Binary Relevance* (BR). However, studies have shown that this approach is not a good solution [Clare and King 2001, Suzuki et al. 2001], since each label is treated individually, ie, generates one classifier for each label ignoring the correlations between them. Intuitively, an algorithm that finds a classifier to more than one label, can capture some correlations between them and find a classifier more simple (for example, a smaller number of rules). Because of this problem, it is important to develop techniques that use the method *Binary Relevance* but they can capture the correlations between the labels.

[1]`http://www.geneontology.org/`

The purpose of this study is to present a new adaptation of the method *Binary Relevance* using decision trees to treat multi-label problems. Decision trees are symbolic learning models that can be analyzed as set of rules, in order to improve the understanding of the knowledge extracted. For this reason, the method proposed here was designed to capture correlations between labels, a feature *Binary-Relevance* doesn't take into account, and consequently upgrade the ability to generalize. Furthermore, our proposal also takes into account the interpretability not only the performance, then our proposal tries to reduce the number of induced trees for expert interpretation, in the best hypothesis build only one model that classifies all labels.

This work is organized as follows: Section 3 presents the basic concepts of multi-label classification. Section 2 describes some related studies in the literature. Section 4 presents our multi-label learning algorithm. Section 5 describes the experimental methodology, results and discussion. Finally, Section 6 presents the conclusions from this study.

2. Related Work

Different techniques have been proposed in the literature for treating multi-label classification problems. In some of them single-label classifiers can be combined to treat multi-label classification problems. Other techniques modify single-label classifiers, changing their algorithms to allow using them in multi-label problems.

In [Cherman et al. 2010] is proposed a method BR + which is an extension of the method BR in which considers the relationship between the labels and is also constructed c binary classification problems analogously to BR. The difference are in descriptors attributes that have all attributes X and contains all the labels as descriptors, except the own label to be predicted.

In [Clare and King 2001] presents a study using decision trees for hierarchical multi-label classification to analyze information of *S. cerevisiae* and try to predict new gene functions. To analyze these data, resampling strategies were developed and modified the algorithm C4.5.

In [Alves et al. 2008] is proposed approach called MHCAIS (Multi-label Hierarchical Classification with an Artificial Immune System) is a adapted algorithm for multi-label and hierarchical classification. The first version of MHCAIS build a global classifier to predict all labels, while the second version of a building one classifier for each label. In both versions the classifier is expressed as a set of IF-THEN rules, which has the advantage of represent knowledge understandable to users biologists.

In [Blockeel et al. 1998] a tool called Clus uses concepts from *Predictive Clustering Trees* (PCT). Decision trees are constructed where each node corresponds to a group of instances from the dataset. PCT is a clustering approach that adapts the basic top-down induction of decision trees for clustering. The procedure used for the construction of PCT is similar to other induction algorithms of decision trees such as C4.5 [Quinlan 1993] or CART [Breiman et al. 1984].

In [Blockeel et al. 2006], Clus-HMC refers to the use of Clus as a multi-label hierarchical classification system that learns a tree to classify all labels and Clus-SC generates one decision tree for each label.

3. Multi-Label Classification

Basically, the classification task is to discover knowledge that can be used to predict the class of an instance, whose class is unknown, based on the values of the attributes that describe such an instance. In this sense there are two versions of the classification task, according to the number of labels to be predicted for each instance: (a) Single-label Classification and (b) Multi-label Classification. Single-label classification refers to the classification task, where there is only one label (the target concept) to be predicted. The basic principles of multi-label classification are similar to single-label classification, however the multi-label classification have two or more concept labels to be predicted. In the case of symbolic models expressed as rules, a multi-label classification rule contains two or more conclusions, each one involving a different label.

Let X be the domain of instances to be classified, Y be the set of labels, and H be the set of classifiers for $f : X \rightarrow Y$, where f is unknown. The goal is to find the classifier $h \in H$ maximizing the probability of $h(x) = y$, where $y \in Y$ is the ground truth label of x [Shen et al. 2004].

Table 1 shows the *attribute-value* representation modified to deal with multi-label problems. A dataset is characterized by N instances $z_1, z_2, \ldots, z_N$, each containing m attributes $X_1, X_2, \ldots, X_m$ and c labels $Y_1, Y_2, \ldots, Y_c$. On this table, the row i refers to the i-th instance ($i = 1, 2, \ldots, N$); the entry x_{ij} refers the value of j-th attribute ($j = 1, 2, \ldots, m$) of instance i and output y_{ik} refers to the value of k-th label ($k = 1, 2, \ldots, c$) of instance i.

Table 1. Set of instances in the format attribute-value for multi-label problems

	X_1	X_2	$\cdots$	X_m	Y_1	Y_2	$\cdots$	Y_c
z_1	x_{11}	x_{12}	$\cdots$	x_{1m}	y_{11}	y_{12}	$\cdots$	y_{1c}
z_2	x_{21}	x_{22}	$\cdots$	x_{2m}	y_{21}	y_{22}	$\cdots$	y_{2c}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
z_N	x_{N1}	x_{N2}	$\ldots$	x_{Nm}	y_{N1}	y_{N2}	$\cdots$	y_{Nc}

As can be seen, instances are tuples $\vec{z}_i = (x_{i1}, x_{i2}, \ldots, x_{im}, y_{i1}, y_{i2}, \ldots, y_{ic}) = (\vec{x}_i, \vec{y}_i)$ also denoted by $z_i = (x_i, y_i)$, where it is implicit the fact that z_i, x_i and y_i are vectors. It is observed that each y_i is a member of the set $Y_1 \times Y_2 \times \ldots \times Y_c$, and $Y_i \in \{0, 1\}$, ie each label has two classes.

4. The BR-CT Methodology

Before introducing our algorithm, some notation is necessary:

- D: the full dataset with all attributes and labels $\{X_1, \ldots, X_c, Y_1, \ldots, Y_c\}$;
- D_l: the labels dataset, defined as $D_l \equiv D \backslash \{X_1, \ldots, X_m\}$;
- D_a: the attributes dataset, where $D_a \equiv D \backslash \{Y_1, \ldots, Y_c\}$;
- D_l^i: an specific labels dataset, defined as $D_l^i \equiv \{Y_1, \ldots, Y_{i-1}, Y_{i+1}, \ldots, Y_c\} \cup \{Y_i\}$, where $\{Y_1, \ldots, Y_{i-1}, Y_{i+1}, \ldots, Y_c\}$ represents learning attributes and $\{Y_i\}$ the target class;

- D_a^i: a dataset containing all attributes and the label Y_i that represents a target class, defined as $D^i \equiv D_a \cup \{Y_i\}$;
- R_j^t: j-th rule from tree t, where $R_j^t \equiv B^t \rightarrow E^t$, the logic implication (if B^t then E^t).

The methodology to deal with multi-label problems proposed here can be seen on Algorithm 1. It can be divided into three steps. The first step (Lines 3-10), performs the induction of c decision trees, taking into account only the labels (Line 4). In this situation, for each label Y_i ($i = 1, \ldots, c$) a decision tree A_i is induced, using as attributes the $c - 1$ remaining labels ($Y_1, \ldots, Y_{i-1}, Y_{i+1}, \ldots, Y_c$) and the label Y_i as the target label. After that, the c decision trees were used to generate a graph G containing initially c nodes where each node i represent the label Y_i. For each A_i, an edge connecting labels Y_i and Y_j is added to G if labels Y_i and Y_j are connected in A_i (Line 7). Figure 1(a) shows an example on how the graph is built from a set of trees $A_1, \ldots, A_4$.

Algorithm 1 Binary Relevance with Correlation Labels - BR-CL

Require: multi-label dataset D containig m attributes $X_1, \ldots, X_m$ and c labels $Y_1, \ldots, Y_c$
Ensure: ExtendedTrees
1: $G \leftarrow \{Y_1, \ldots, Y_c\}$
2: $Extended \leftarrow \emptyset$
3: **for** $i \leftarrow 1$ **to** c **do**
4: $A_i \leftarrow$ BuildDecisionTree(D_l^i)
5: **for** $w \leftarrow 1$ **to** c **do**
6: **if** $Y_w \subset A_i$ **then**
7: $G \leftarrow G \cup \{(Y_i, Y_w)\}$
8: **end if**
9: **end for**
10: **end for**
11: **for** $i \leftarrow 1$ **to** c **do**
12: $T_i \leftarrow$ BuildDecisionTree(D_a^i)
13: $S \leftarrow A_i$
14: $T_i' \leftarrow T_i$
15: **loop**
16: $SR \leftarrow$ SelectAllRules(S), where $E_j^{T_i'} = E_k^S$
17: $R^{T_i'} \leftarrow$ BuildRules(SR), in form $B_j^{T_i'} \rightarrow E_j^{T_i'} \wedge B_k^S$
18: $L(R^{T_i'}) \leftarrow$ calculate laplace of $R^{T_i'}$
19: $\Omega \leftarrow$ select the rule with the largest $L(R^{T_i'})$ precision
20: Extended $\leftarrow$ Extended $\cup \{Y_1, Y_{i-1}, Y_{i+1}, \ldots, Y_c\} \cap \Omega$
21: $T_i' \leftarrow T_i' \cup$ Extended
22: **if** There are labels to be considered from $\{Y_1, \ldots, Y_{i-1}, Y_{i+1}, \ldots, Y_c\}$ **then**
23: $SL \leftarrow$ select one label y considering the best accuracy of A_y from Extended
24: $S \leftarrow A_{SL}$
25: **else**
26: exit loop
27: **end if**
28: **end loop**
29: **end for**
30: ExtendedTrees $\leftarrow \emptyset$
31: **for** $j \leftarrow 1$ **to** $C(G)$ **do**
32: ExtendedTrees $\leftarrow$ ExtendedTrees $\cup$ {select T_i' with the best HammingLoss}
33: **end for**
34: **return** ExtendedTrees

This step tries to find groups of related labels, being represented by a connected component in G. Let us denote $C(G)$ the number of connect components in graph G. At the end of this step, there are three possible situations: (1) $C(G) = 1$, all labels are related to each other, and therefore there is only one connected component in G that contains all the labels; (2) $C(G) = c$, no labels are related to each other, then G contains c connected components; and (3) $1 < C(G) < c$, there is some relationship between some labels.

In the second stage (Lines 11-29) is carried out the induction of tree T_i, but taking into account all attributes $X_1, \ldots, X_m$ and just one label Y_i at a time (Line 12). After that, each decision tree T_i is extended (Line 15), i.e., a process is performed in order that each tree T_i can predict more than one label. Is used the connected component of G in which contains the node Y_i since the tree T_i is related to the label Y_i whereas it is the label to be predicted by T_i. This results in new trees T_i' that have a list of labels at each leaf node, that is, if all labels are correlated (first case above) then the tree will be extended to include all the labels on its leaves. If there are two or more connected component (third case), the tree is extended only for labels that are part of its component in G. For that, firstly the tree A_i is selected to start the extension S for tree T_i, where $S \leftarrow A_i$ (Line 13).

For each rule j in T_i, a rule is created up to the root level of the tree. For each rule j from T_i are then selected all k rules S, where $E_j^{T_i} = E_k^S$ (Line 16). After that, all k rules $R_k^{T_i}$ are built in logical form $R_k^{T_i} \equiv B^{T_i} \rightarrow E^{T_i} \wedge B^S$ (Line 17), i.e., premise and conclusion of k-th rule of T_i are united with premise of rule S.

Then, it is calculated laplace precision metric (Line 18) for all de rules $R_k^{T_i}$ to find which rule is more accurate. After that, a best rule is selected with the largest laplace value, this rule will be used to extend the rule j of the tree T_i (Line 19). The Laplace 1 [Clark and Niblett 1989] is defined in (1), where $N(B \rightarrow E)$ is the number of instances satisfying premise and conclusion, $N(B)$ is the number of instances which satisfies only the premise, and $\widehat{k}$ is the number of classes in the domain of Y_i. In our experiments, since $Y_i \in \{0, 1\}$ then $\widehat{k} = 2$:

$$L(R_k^{T_i'}) = \frac{\sum_{i=1}^{N} N(B \rightarrow E) + 1}{\sum_{i=1}^{N} N(B) + \widehat{k}} \tag{1}$$

Figure 1(b) shows how the extension a tree is performed, where the computation of laplace values is done for each $R_k^{T_i}$, choosing the largest value, as mentioned earlier. The example shows how extend the first rule in tree T_1.

If not all labels have been extended from T_i components (Line 22), then the process continues the extension on other tree. This tree is selected considering only the trees in Extended, where Extended is a subset from $\{A_1, \ldots, A_{i-1}, A_{i+1}, \ldots, A_c\}$. Only if Y_i appeared in the rule selected then A_i is considered as a part of Extended (Line 24). Otherwise if the extension of the tree T_i is finish (Line 26) the loop is exited.

In the third stage (Lines 30-33) occurs the selection of the tree with lower HammingLoss rate per component(Line 32). This step allows you to select the best tree for each component and thus decrease the number of trees to be analyzed since the result of the algorithm shows only $C(G)$ trees, each of which is the best tree by component.

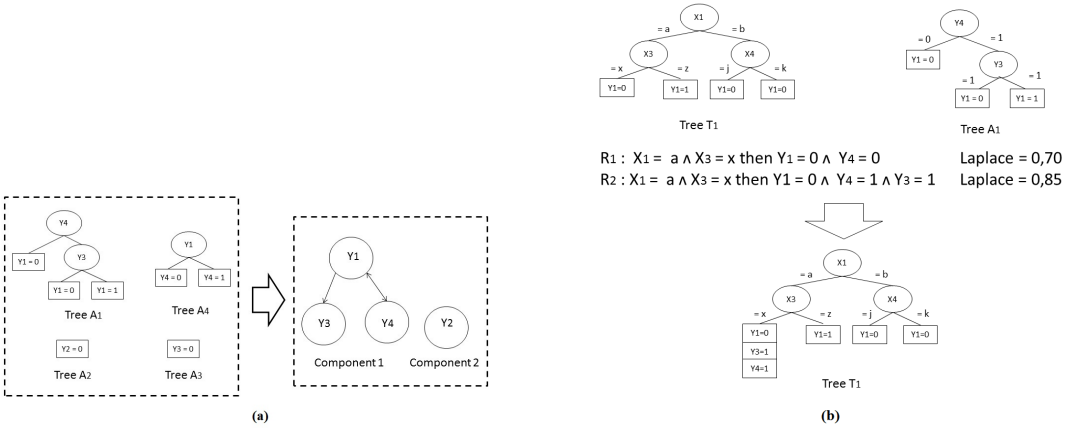

Figure 1. Figure 1(a) shown the transformation of trees A_i (left) in graph G (right) and Figure 1(b) shown a extending Tree T_1

5. Experimental Metodology

5.1. Experimental Setup

The datasets used in experiments reported here are from the functional genomic field, are available from Catholic University of Leuven[2] related to *Saccharomyces cerevisiae*. Were considered 16 labels in the experiments which are structured hierarchically according to the catalog developed by MIPS Funcat [Mewes et al. 2004] available on 24/04/2002[3], considering only the first level of the hierarchy.

A pre-processing of datasets was necessary to transform them into non-hierarchical data. In this case, instead a hierarchical attribute-class, a binary vector was created in which each position corresponds to a main category contained in the class of hierarchical dataset. Then each instance was transformed from hierarchical class to non-hierarchical considering only the first level of the hierarchy.

The experiments reported were performed using the Weka library [Witten and Frank 1999]. In the proposed method, the decision trees were based on the algorithm J48 [Quinlan 1993] with *default* settings. We have evaluated our algorithm as well as five other methods from the Mulam library: Binary Relevance, Label Powerset [Tsoumakas et al. 2010], RAkEL (RAndom k-labELsets) [Tsoumakas and Vlahavas 2007], MLkNN (Multi-Label k-Nearest Neighbours) [Zhang and Zhou 2007] and BPMLL (Back-Propagation Multi-Label Learning) [Zhang 2006].

The method Label Powerset is based on a combination of more than one label to create a new label, but the number of labels can increase considerably and some may end up with a few instances. The method RAkEL constructs a ensemble of LP classifiers and each classifier is trained with a small subset of random k labels. The algorithm ML-KNN is based on the kNN algorithm: for each test instance, its k nearest neighbors in the training set are identified then, according to statistical information gained from the label sets of these neighboring instances, maximum a posteriori principle is utilized to

[2]http://dtai.cs.kuleuven.be/clus/hmc-ens/

[3]http://www.aber.ac.uk/ dcswww/Research/bio/dss/yeastpreds/yeast/classes.txt

determine the label set for the test instance. The algorithm BPMLL is an adaptation of the popular algorithm back-propagation for multi-label learning, the main modification of this algorithm is the introduction of a new error function that considers multiple labels.

For methods BR, LP and RAkEL the algorithm J48 was used with minimal number of objects equals 2, no binary split and no pruning method settings and for methods MLkNN and BPMLL the default settings was used too. Another method includes Clus with reduce variance as heuristic, no binary split, no pruning method and minimal weight equals 2.

Moreover, in the course of research and development methodology first it was decided to select one tree by component, being one with better accuracy and then extend only those selected; however, a preliminary test carried out with a simple dataset showed not necessarily the tree selected get the best HammingLoss if all the trees were extended before selecting. Therefore, to statistically analyze these two possibilities were tested both options, BR-CTa for which the selection is made before the extension and BR-CTb for which the selection is made after the extension.

To analyze the performance 10-fold cross-validation was performed for each method and each dataset, recording the metrics HammingLoss (2) and F-measure described previously. To analyze significant results the Friedman test [Friedman 1940] was used, considering a significance level of 5%, and the Benjamini-Hochberg as the *post-hoc* test [Benjamini and Hochberg 1995].

$$HammingLoss(h,t) = \frac{1}{|N|} \times \sum_{i=1}^{N} \frac{|y_i \Delta \hat{y}_i|}{|Y|} \tag{2}$$

5.2. Results and Discussion

The results of HammingLoss measure and F-measure are shown on Tables 2 and 3, respectively. It is also shown the rank of each approaches (between curly brackets), and the average rank obtained by the Friedman test considering a significance level of 5%.

Table 2. HammingLoss

Dataset	BR-CT	BR	LP	RAkEL	MLkNN	BKMLL	Clus
derisi	0.098(2.5)	0.100(4.0)	0.134(6.0)	0.1085(.0)	0.098(2.5)	0.173(7.0)	0.032(1.0)
seq	0.107(4.0)	0.116(6.0)	0.129(7.0)	0.105(3.0)	0.097(2.0)	0.110(5.0)	0.028(1.0)
pheno	0.101(5.0)	0.097(1.5)	0.116(6.0)	0.097(1.5)	0.099(4.0)	0.203(7.0)	0.098(3.0)
gasch2	0.113(4.5)	0.107(3.0)	0.135(7.0)	0.109(6.0)	0.100(2.0)	0.113(4.5)	0.046(1.0)
expr	0.112(4.0)	0.120(6.0)	0.134(7.0)	0.108(3.0)	0.101(2.0)	0.113(5.0)	0.032(1.0)
church	0.112(5.0)	0.105(3.0)	0.133(6.5)	0.107(4.0)	0.103(2.0)	0.113(6.5)	0.061(1.0)
gasch1	0.113(4.5)	0.126(6.0)	0.133(7.0)	0.111(3.0)	0.099(2.0)	0.113(4.5)	0.036(1.0)
cellcycle	0.115(5.0)	0.117(6.0)	0.140(7.0)	0.111(3.0)	0.102(2.0)	0.113(4.0)	0.044(1.0)
spo	0.112(5.0)	0.108(3.0)	0.141(7.0)	0.111(4.0)	0.102(2.0)	0.113(6.0)	0.036(1.0)
eisen	0.119(4.0)	0.120(5.0)	0.132(7.0)	0.106(3.0)	0.094(2.0)	0.126(6.0)	0.036(1.0)
Average Rank	4.450	4.350	6.800	3.350	2.250	5.600	1.200

Considering HammingLoss, it is possible to note Clus had the best average rank, and the BR-CT had the fifth best rank, but almost the same performance as the BR. In the

Table 3. F-measure

Dataset	BR-CT	BR	LP	RAkEL	MLkNN	BPMLL	Clus
derisi	0.402(4.0)	0.417(2.0)	0.372(6.0)	0.384(5.0)	0.405(3.0)	0.482(2.0)	0.869(1.0)
seq	0.517(2.0)	0.457(4.0)	0.412(6.0)	0.495(3.0)	0.435(5.0)	0.001(7.0)	0.885(1.0)
pheno	0.394(1.5)	0.394(1.5)	0.377(4.0)	0.149(7.0)	0.356(5.0)	0.268(6.0)	0.387(3.0)
gasch2	0.461(2.0)	0.424(4.0)	0.387(6.0)	0.428(3.0)	0.409(5.0)	0.001(7.0)	0.839(1.0)
expr	0.457(2.0)	0.429(3.0)	0.389(6.0)	0.420(5.0)	0.422(4.0)	0.001(7.0)	0.857(1.0)
church	0.395(3.0)	0.353(6.0)	0.402(2.0)	0.390(4.0)	0.369(5.0)	0.001(7.0)	0.689(1.0)
gasch1	0.462(2.0)	0.422(4.0)	0.4056(.0)	0.459(3.0)	0.421(5.0)	0.001(7.0)	0.867(1.0)
cellcycle	0.465(2.0)	0.426(4.0)	0.373(6.0)	0.444(3.0)	0.403(5.0)	0.001(7.0)	0.838(1.0)
spo	0.392(4.0)	0.379(5.0)	0.372(6.0)	0.398(3.0)	0.407(2.0)	0.001(7.0)	0.874(1.0)
eisen	0.452(6.0)	0.498(4.0)	0.473(5.0)	0.543(2.0)	0.515(3.0)	0.001(7.0)	0.877(1.0)
Average Rank	2.950	3.850	5.400	3.900	4.300	6.400	1.200

Table 4. Number of Leaves

Conjunto de exemplos	BR-CTa	BR-CTb	Clus
derisi	7	7	1462
seq	90	16	1420
pheno	12	12	13
gasch2	62	22	1252
expr	62	4	1172
church	21	15	965
gasch1	80	76	1218
cellcycle	80	32	1233
spo	72	65	1396
eisen	87	55	771

context of F-measure the algorithm with the best average rank was Clus too, and BR-CT had the second best rank.

In Table 4 is shows the number of leaves (rules) results of BR-CTa, BR-CTb and Clus methods. Analyzing the table we can see that the number of leaves of the tree resulting of Clus method is far superior against the number of leaves generated by the set of trees of the proposed method (BR-CTa and BR-CTb). So, although the method Clus has best performance, the BR-CT methods builds smaller trees facilitating its interpretability.

The result of *post-hoc* test is shown on Table 5 considering HammingLoss and F-measure metrics, in which symbol $\triangle$ ($\blacktriangle$) means that the BR-CT is better (significantly) than the classifier at the row whereas the symbol $\triangledown$ ($\blacktriangledown$) means that BR-CT is worse (significantly) than the classifier at the row.

Table 5. Benjamini-Hochberg *post-hoc* Test - BR-CT versus all

	HammingLoss	F-measure
BR	$\triangledown$	$\triangle$
LP	$\blacktriangle$	$\blacktriangle$
RAkEL	$\triangledown$	$\triangle$
MLkNN	$\blacktriangledown$	$\triangle$
BPMLL	$\triangle$	$\blacktriangle$
Clus	$\blacktriangledown$	$\triangledown$

As can be seen, taking into account the HammingLoss our proposal had a good performance in regard LP (significantly) and BPMLL, and had a bad performance with BR, RAkEL, MLkNN and Clus being significantly in the last two. Otherwise taking into account the F-measure our proposal had a good performance against all approaches except for Clus, and two of which are significantly better (LP and BPMLL). But our methodology takes into account the comprehensiveness not only the performance, and the MLkNN can not be interpreted by the specialist and the model generated by Clus has a high complexity, as they usually are very large, which can be seen in Table 4.

6. Conclusions

In this paper a study of multi-label classification problem has been conducted. In such kind of problem, there is more than one label to be predicted, i.e., an instance may be related to more than one label at the same time making the classification task more difficult. In order to improve performance and comprehensibility of the extracted model, in this study we had proposed an adaptation for Binary Relevance method in order to overcome the BR disadvantage: we consider the correlations between the labels and thus this may improve the generalization of the model induced, and possibly may decrease the number of classifiers to be analyzed. When all labels are correlated, our proposal finds a single classifier (decision tree) that can classify all labels. When labels as uncorrelated the output model of our proposal will be equal to BR (one decision tree for each label).

Experiments have been conducted to compare the performance of our proposal against other approaches found in the literature. Results allow us to conclude our proposal has performance comparable to other methods, obtaining nice results using F-measure and average results using HammingLoss. Since the main concern of our research group embraces human interpretability in the induced models, then regardless of the method Clus always get better performance than the proposal, it generates trees of large size in comparison with the proposed method.

Acknowledgements

This work was partially funded by a joint grant between the Coordination for the Improvement of Higher Level (CAPES), and the Amazon State Research Foundation (FAPEAM) through the Program National Institutes of Science and Technology, INCT ADAPTA Project (Centre for Studies of Adaptations of Aquatic Biota of the Amazon).

References

Alves, R. T., Delgado, M. R., and Freitas, A. A. (2008). Multi-label hierarchical classification of protein functions with artificial immune systems. *Advances in Bioinformatics and Computational Biology*, pages 1–12.

Benjamini, Y. and Hochberg, Y. (1995). Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society Series B*, 57:289–300.

Blockeel, H., Raedt, L. D., and Ramon, J. (1998). Top-down induction of clustering trees. In *Proceedings of the 15th International Conference on Machine Learning*, ICML '98, pages 55–63.

Blockeel, H., Schietgat, L., Struyf, J., Clare, A., and Dzeroski, S. (2006). Hierarchical multilabel classification trees for gene function prediction.

Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. (1984). *Classification and Regression Trees*. Chapman and Hall/CRC, 1 edition.

Cherman, E. A., Metz, J., and Monard, M. C. (2010). Métodos multirrótulo independentes de algoritmo: um estudo de caso. In *Anais da XXXVI Conferencia Latinoamericana de Informática (CLEI)*, pages 1–14, Asuncion, Paraguay. Publicado em CD-ROM.

Clare, A. and King, R. D. (2001). Knowledge discovery in multi-label phenotype data. *Lecture Notes in Computer Science*, pages 42–53.

Clark, P. and Niblett, T. (1989). The cn2 induction algorithm. In *MACHINE LEARNING*, volume 3, pages 261–283.

Friedman, M. (1940). A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics*, 11(1):86–92.

Mewes, H. W., Frishman, D., Mayer, K. F. X., Münsterkötter, M., Noubibou, O., Rattei, T., Oesterheld, M., and Stümpflen, V. (2004). Mips: Analysis and annotation of proteins from whole genomes. *Nucleic Acids Res*, 32:41–44.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann. San Francisco, CA.

Ruepp, A., Zollner, A., Maier, D., Albermann, K., Hani, J., Mokrejs, M., Tetko, I., Güldener, U., Mannhaupt, G., Münsterkötter, M., and Mewes, H. W. (2004). The funcat, a functional annotation scheme for systematic classification of proteins from whole genomes. *Nucleic Acids Research*, 32(18):5539–5545.

Schietgat, L., Vens, C., Struyf, J., Blockeel, H., Kocev, D., and Dzeroski, S. (2010). Predicting gene function using hierarchical multi-label decision tree ensembles. *BMC Bioinformatics*, 11(1):2+.

Shen, X., Boutell, M., Luo, J., and Brown, C. (2004). Multi-label Machine Learning and Its Application to Semantic Scene Classification. In *Storage and Retrieval Methods and Applications for Multimedia*, pages 18–199.

Suzuki, E., Gotoh, M., and Choki, Y. (2001). Bloomy decision tree for multi-objective classification. pages 436–447.

Tsoumakas, G., Katakis, I., and Vlahavas, I. (2010). Mining multi-label data. pages 667–685. Springer.

Tsoumakas, G. and Vlahavas, I. (2007). Random k-labelsets: An ensemble method for multilabel classification. *Machine Learning: ECML 2007*, 4701:406–417.

Witten, I. H. and Frank, E. (1999). *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*, volume 1. Morgan Kaufmann.

Zhang, M. L. (2006). Multilabel Neural Networks with Applications to Functional Genomics and Text Categorization. *IEEE Transactions on Knowledge and Data Engineering*, 18(10):1338–1351.

Zhang, M. L. and Zhou, Z. H. (2007). Ml-knn: A lazy learning approach to multi-label learning. *Pattern Recogn.*, 40(7):2038–2048.