Journal of Biomedical Informatics 54 (2015) 85–95

Contents lists available at ScienceDirect

Journal of Biomedical Informatics

journal homepage: www.elsevier.com/locate/yjbin

A multi-label approach using binary relevance and decision trees applied to functional genomics

Erica Akemi Tanaka [a], Sérgio Ricardo Nozawa [b,1], Alessandra Alaniz Macedo [a], José Augusto Baranauskas [a,*]

[a] Department of Computer Science and Mathematics, University of Sao Paulo (USP), Av. Bandeirantes, 3900, Ribeirão Preto, SP 14040-901, Brazil
[b] Dow AgroSciences (Seeds, Traits & Oils), Av. Antonio Diederichsen, 400, Ribeirão Preto, SP 14020-250, Brazil

ARTICLE INFO

Article history:
Received 14 August 2014
Accepted 18 December 2014
Available online 27 December 2014

Keywords:
Multi-label classification
Decision tree
Functional genomics

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 algorithm taking into account possible relations among labels, focusing on the interpretability of the model, not only on its performance. Experiments were conducted to compare the performance of our approach against others commonly found in the literature and applied to functional genomic datasets. The experimental results show that our proposal has a performance comparable to that of other methods and that, at 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 has become possible. The amount of biological data available has been increasing, which also increases the need for computational tools for knowledge extraction. Machine learning techniques are widely used to predict gene functions so that the best predictions can then be tested in the lab to validate the results [1]. However, predicting gene functions is a complex process because a single gene may have multiple functions. Consequently, multi-label classification seems to be appropriated.

There are several reasons to investigate and propose new multi-label classification techniques, especially in the bioinformatics or bio-related research fields. *Gene Ontology*[2] is an example of a multi-label problem, where genes and proteins may have more than one function or feature. Another example is the MIPS Functional Catalogue [2], in which genes and proteins may belong to more than one functional class. Therefore, it is very important to carry out research on computational techniques to classify multi-label problems using proteins, genes and other biological and medical data: with such knowledge it is possible to develop new drugs, treat diseases, and help in diagnostics.

Traditional algorithms are unable to handle a set of multi-label instances, since such algorithms were designed to predict a single label. A simple solution to this is to transform the original dataset into several sets of instances where each set contains all the attributes, but only one label to be predicted. This algorithm is known as *Binary Relevance* (BR). However, studies have shown that this approach is not a good solution [3,4], since each label is treated individually, generating one classifier for each label, and ignoring possible correlations among them. An algorithm that finds a classifier for more than one label can intuitively capture some correlations between them, and a simpler classifier may be found (one which uses a smaller number of rules, for example). Under these circumstances, it is important to research and develop techniques that use the *Binary Relevance* algorithm, extending it to capture possible relations among labels.

This study presents a new adaptation of the *Binary Relevance* algorithm 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, by human experts, about the knowledge extracted. For this reason, the algorithm proposed here was designed to capture relations between labels, a feature the original *Binary Relevance* algorithm does not take into account, and consequently upgrade its generalization ability. Furthermore, since the present study takes model interpretability into account (and not only performance), our approach reduces the number of induced trees for expert interpretation: in the best scenario, it builds only one model (tree) that classifies all labels.

This paper is organized as follows: Section 2 describes related studies in the literature; Section 3 presents the basic concepts of multi-label classification; Section 4 presents our multi-label learning algorithm. Section 5 describes the experimental methodology to

* Corresponding author. Fax: +55 16 3315 0407.
E-mail addresses: kemi@gmail.com (E.A. Tanaka), srnozawa@gmail.com (S.R. Nozawa), ale.alaniz@usp.br (A.A. Macedo), augusto@usp.br (J.A. Baranauskas).
[1] Fax: +55 16 3602 5696.
[2] http://www.geneontology.org/.

http://dx.doi.org/10.1016/j.jbi.2014.12.011
1532-0464/

evaluate our approach; Results and discussion are presented in Section 6. Finally, Section 7 presents the final remarks and future work.

2. Related work

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

BR + algorithm [5], an extension of the BR algorithm, considers the relationship between labels, and constructs binary classification problems, similarly to BR. Its main differences are its descriptor attributes, which merge all original attributes as well as all labels, except for the label to be predicted itself.

Another study using decision trees for hierarchical multi-label classification was used to analyze information about *Saccharomyces cerevisiae*, and tries to predict new gene functions [3]. Resampling strategies were developed, and a modified version of the algorithm C4.5 [6] was used.

The Mulam [7] tool was developed based on the Weka machine learning library [8], and contains several algorithms, such as BR (Binary Relevance) [9], LP (Label Powerset) [9], RaKel (RAndom k-labELsets) [10], and ML-kNN (Multi-Label k-Nearest Neighbours) [11]. In the Binary Relevance algorithm, the original dataset is divided into sets of instances, where each instance contains all the attributes but only the label to be predicted. Then, c classifiers are induced (where c represents the total number of labels), and each induced classifier is trained to distinguish one label against all the others involved. The Label Powerset algorithm is based on a combination of more than one label to create a new one, but this may result in a considerable increase in the number of labels, and some may end up with few instances. The RAkEL algorithm constructs an ensemble of LP classifiers, and each classifier is trained with a small subset of k random labels. Algorithm ML-KNN is based on algorithm kNN: for each test instance, its k nearest neighbors in the training set are identified. Then, according to statistical information from the label set of neighboring instances, the maximum a posteriori principle is applied to determine the label set for a particular test instance.

A tool called Clus [12] 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 constructing the PCT is similar to other induction algorithms of decision trees such as C4.5 [6] and CART [13]. Clus-HMC [14] 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 a decision tree for each label.

MHCAIS (Multi-label Hierarchical Classification with an Artificial Immune System) [15] is an adapted algorithm for multi-label and hierarchical classification. The first version of this algorithm builds a global classifier to predict all labels, while the second version builds a classifier for each label. In both versions, the classifier is expressed as a set of IF–THEN rules, which has the advantage of being knowledge understandable to specialists.

Other researchers developed a Network Hierarchical Multi-label Classification algorithm that exploits individual properties of proteins as well as protein–protein interactions (PPI) to predict gene/protein functions [16]. These researchers advocate that (i) the PPI network is exploited in the training phase and can thus make predictions for genes/proteins whose interactions are yet to be investigated; (ii) their method yields better performance than the others by using network and properties separately; and (iii) the use of network information improves the accuracy of gene function prediction not only for highly connected genes, but also for genes with only a few connections. Like Clus-HMC, NHMC also exploits the hierarchical organization of class labels (gene functions), which may have the form of a tree or of a direct acyclic graph (DAG).

The R3P-Loc is a multi-label ridge regression classifier that uses two databases for feature extraction, applying random projection to reduce its feature dimensions [17]. In terms of locating proteins within cellular contexts, R3P-Loc indicates a reduction in the number of dimensions of feature vectors as much as seven-folds, while it also improves the classification performance. Considering the multi-level classification of phylogenetic profiles, authors have proposed an algorithm to capture, at each level, the different aspects of affinity of a protein with another, in the same or in different species [18]. As a result, inter and intra-genome gene clusters are predicted. Aiming at facilitating biological interpretation, the same authors extract close gene associations from metabolic pathways through unsupervised clustering at a sequence level [19]. This level of association can be enhanced if the phylogenetic relationship of the corresponding genomes is taken under consideration.

3. Background: multi-label classification

Basically, the classification task aims to discover knowledge that can be used to predict the unknown class of an instance, based on the values of the attributes that describe such an instance. As a result, we can divide the classification tasks according to the number of labels to be predicted for each instance into two groups: (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 [20]. The basic principles of multi-label classification are similar to single-label classification, however the multi-label classification has two or more concept labels to be predicted. Considering symbolic models expressed as rules, a multi-label classification rule contains two or more conclusions, each one involving a different label.

Next, we formalize the notation used in the remaining text. 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 [21].

Table 1 shows the modified representation of *attribute–value* 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, row i refers to the i-th instance ($i = 1, 2, \ldots, N$); 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. The 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 the fact that z_i, x_i and y_i are vectors is implicit. Note each y_i is a member of the set $Y_1 \times Y_2 \times \ldots \times Y_c$; without loosing generality we will assume $Y_i \in \{0, 1\}$, i.e., each label will only assume binary values.

4. Proposal: The BR-DT algorithm

Next, before introducing our algorithm, we introduce some additional notations:

- D: the full dataset with all attributes and labels $\{X_1, \ldots, X_m, Y_1, \ldots, Y_c\}$;

Table 1
Set of instances in the attribute–value format 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}	$\cdots$	x_{Nm}	y_{N1}	y_{N2}	$\cdots$	y_{Nc}

- $x_i \equiv \vec{x}_i = (x_{i1}, x_{i2}, \ldots, x_{im})$: learning attributes for instance i;
- $y_i \equiv \vec{y}_i = (y_{i1}, y_{i2}, \ldots, y_{ic})$: learning labels for instance i;
- $z_i = (x_i, y_i) \equiv \vec{z}_i = (\vec{x}_i, \vec{y}_i)$: learning instance i;
- $h(x_i)$: the multi-label classifier model which outputs the predicted labels for instance i;
- D_l: the labels dataset $D_l \equiv D \setminus \{X_1, \ldots, X_m\}$;
- D_a: the attributes dataset $D_a \equiv D \setminus \{Y_1, \ldots, Y_c\}$;
- $D_l^i : D_l^i \equiv \{Y_1, \ldots, Y_{i-1}, Y_{i+1}, \ldots, Y_c\} \cup \{Y_i\}$ is a label dataset where $\{Y_1, \ldots, Y_{i-1}, Y_{i+1}, \ldots, Y_c\}$ are learning attributes and $\{Y_i\}$ is 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 $R_j^t \equiv B^t \rightarrow E^t$ represents the logic implication (if B^t then E^t).

The strategy to deal with multi-label problems proposed in this study is presented in Algorithm 1. It can be divided into three main steps.

Algorithm 1. Binary Relevance with Decision Trees – BR-DT

Require: multi-label dataset D containig m attributes $X_1, \ldots, X_m$ and c labels $Y_1, \ldots, Y_c$
Ensure: ExtendedTrees
1: $G \leftarrow \emptyset$
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 $R_j^{T_i'} = R_k^S$
17: $Rule^{T_i'} \leftarrow$ BuildRules(SR), in form $L_j^{T_i'} \rightarrow R_j^{T_i'} \wedge R_k^S$
18: $L(Rule^{T_i'}) \leftarrow$calculate laplace of $Rule^{T_i'}$
19: $\Omega \leftarrow$select the rule with the largest $L(Rule^{T_i'})$ 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

The first step (Lines 3–10), in Fig. 1, performs the induction of c decision trees, only taking labels into account (Line 4). In this situation, for each label Y_i ($i = 1, \ldots, c$), a decision tree A_i is induced, using the $c-1$ remaining labels ($Y_1, \ldots, Y_{i-1}, Y_{i+1}, \ldots, Y_c$) as attributes and label Y_i as the target label. After that, the c trees induced earlier are converted into an initially empty graph structure G. Let $C(G)$ be the number of connected components of graph G. For each A_i, an edge connecting labels Y_i, and Y_j is added to G iff labels Y_i and Y_j are connected in A_i (Line 7). Fig. 1 shows an example of how the graph is built from a set of trees $A_1, \ldots, A_c$.

This first step tries to find groups of related labels and is represented by a connected component in G. It does not consider the directionality of edges, i.e., an undirected graph is built. 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 label is related to one another, then G contains c connected components; and
3. $1 < C(G) < c$, there are relationships among some labels.

In the second step (Lines 11–29), in Fig. 2, the induction of tree T_i is carried out, now 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 so that each tree T_i can predict more than one label. The connected component of G containing node Y_i is considered, since tree T_i is related to label Y_i, whereas it is the label to be predicted by T_i. This results in new trees $T\prime_i$ that have a list of labels at each leaf node. If all labels are correlated (first situation above), then the tree will be extended to include all labels on its leaves. If there are two or more connected components (third situation), the tree is extended only for labels that are part of its component in G. If the second situation occurs, there are exactly c unextended trees, one for each label.

Still in the second step, tree A_i is initially selected to start the extension S for tree T_i, where $S \leftarrow A_i$ (Line 13). For each rule j from T_i, a rule is created up to the root level of the tree. For each rule j from T_i all k rules S are then selected, 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., the premise and conclusion of k-th rule of T_i are concatenated with the premise of rule S.

Then, the Laplace precision metric is computed (Line 18) for all rules $R_k^{T_i}$ to find the most accurate.[3] The best rule is selected with the largest Laplace value; this rule will be used to extend rule j from tree T_i (Line 19). The Laplace [22] is defined in (1), where $N(B \rightarrow E)$ is the number of instances satisfying both the premise and the 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 $\widehat{k} = 2$:

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

[3] To choose the best metric, other metrics were investigated apart from Laplace. Specificity, negative precision and precision consider the set of instances with false premises and conclusions. Consequently, it is not interesting to use them to select the best rule, which must consider the set of instances with positive premise and conclusion. Metrics composed by divisor as the set of instances are not also interesting because conclusion identifies rules to defined edges. As a result, the idea of dividing the total number of instances eliminates the use of conclusion. Positive precision and sensitivity have unwanted properties because they further rules composed by less instances (false negative and false positive) without considering true positive. A solution to this problem is Laplace.

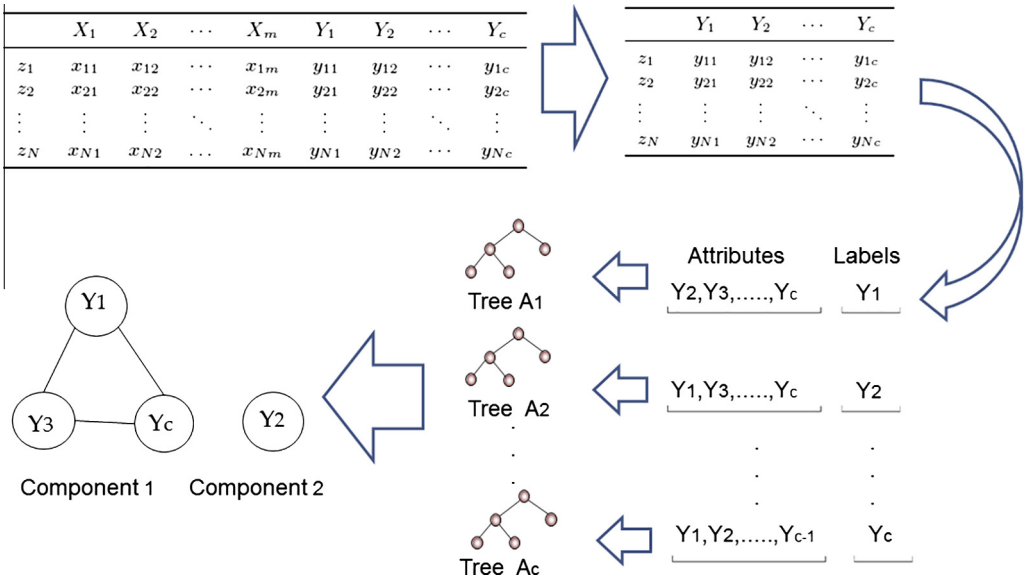

Fig. 1. BR-DT Algorithm – Step 1.

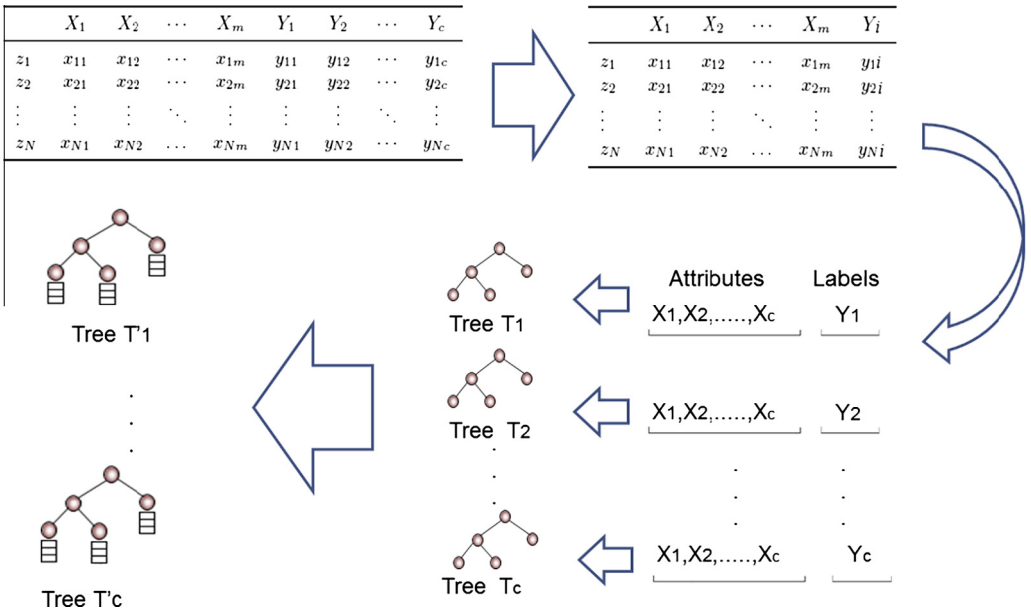

Fig. 2. BR-DT Algorithm – Step 2.

Fig. 3a shows the extension of trees, where the computation of Laplace values is carried out for each $R_k^{T_i}$, choosing the largest value, as mentioned earlier; Fig. 3b shows how to extend the first rule on tree T_1.

In case not all labels are extended from T_i components (Line 22), the process continues the extension on another tree. This tree is selected considering only trees in Extended, a subset from $\{A_1, \ldots, A_{i-1}, A_{i+1}, \ldots, A_c\}$. Only in case Y_i appears in the rule selected is A_i considered as a part of Extended (Line 24). Otherwise, if the extension of the tree T_i is finished (Line 26), the loop terminates.

Finally, the third step, (Lines 30–33), in Fig. 4, is when the selection of the tree with the lowest HammingLoss rate (see Section 5.4) per component occurs (Line 32). This step allows the selection of the best tree for each component, thus decreasing the number of trees to be analyzed. The algorithm outputs $C(G)$ trees, each one of which is the best tree by component.

5. Experimental methodology

In order to evaluate the BR-DT Algorithm, we performed the following experiments:

5.1. Functional genomic datasets

The datasets used in the experiments reported here are from the functional genomic field, made available by the Catholic University of Leuven,[4] and are related to *S. cerevisiae*, in which the labels are hierarchically structured according to the Funcat catalog [23] developed by MIPS.[5] This catalog provides descriptions of functional proteins, and is structured as a four-level-deep tree.

In the experiments, we used the following ten sets of instances:

[4] http://dtai.cs.kuleuven.be/clus/hmc-ens/.
[5] http://www.aber.ac.uk/~dcswww/Research/bio/dss/yeastpreds/yeast/classes.txt.

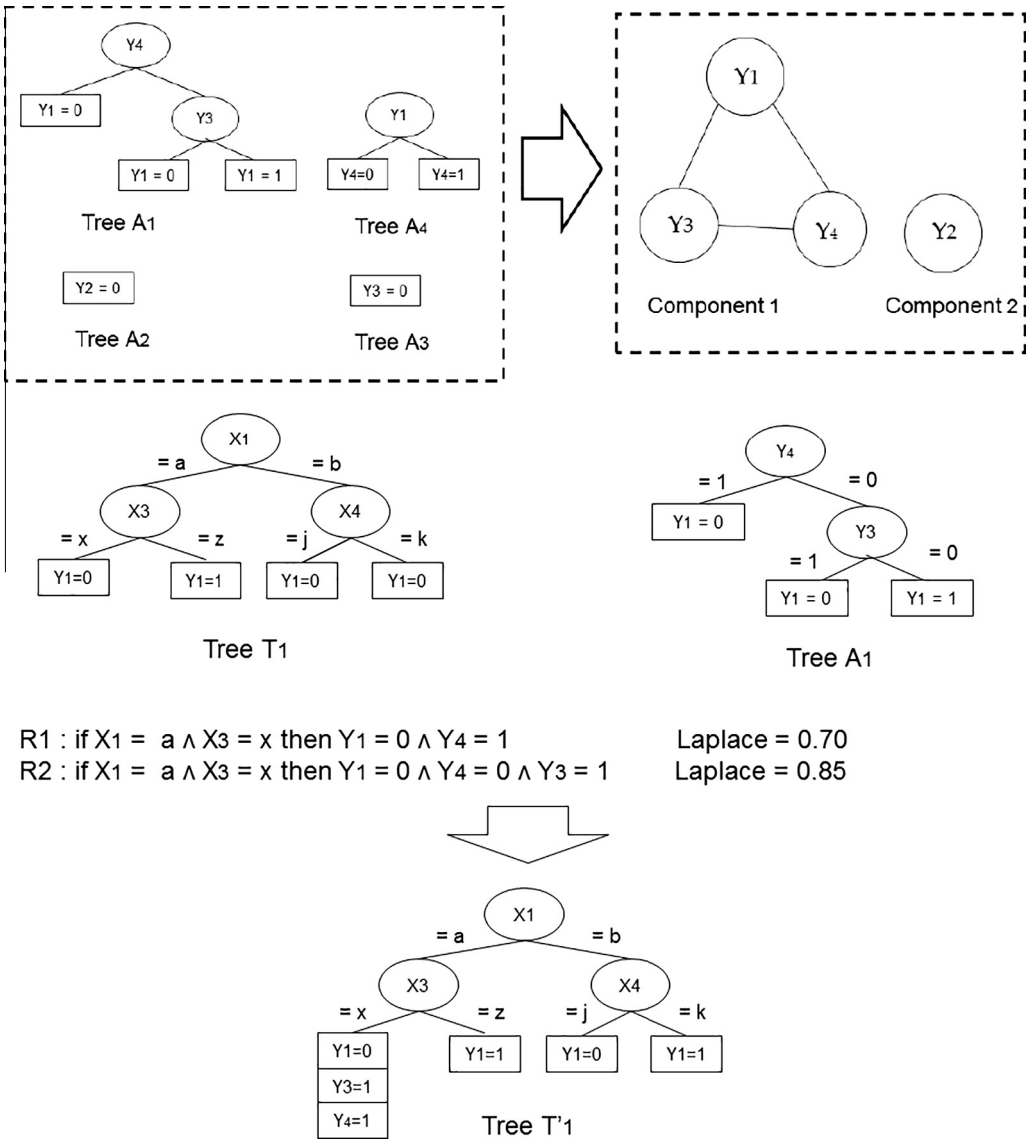

Fig. 3. (a) Transformation of trees A_i (left) into graph G (right); (b) extending tree T_1.

- **Seq**: contains sequence statistics, and is collected from various sources including ProtParam [24] and MIPS;
- **Pheno**: contains phenotype data, and is collected from various sources including TRIPLES [25], EUROFAN [26] and MIPS;
- **CellCycle, Church, Derisi, Expr, Eisen, Gasch1, Gasch2, SPO**: *microarray* data from [27–33], respectively;

A pre-processing of datasets was necessary to transform them into non-hierarchical data. Instead of a hierarchical attribute-class, a binary vector was created in which each position corresponded to a main category contained in the class of hierarchical dataset. Then, each instance was transformed from hierarchical class to non-hierarchical:

- Hierarchy Level 1: only considers the first hierarchy level (16 labels);
- Hierarchy Level 2: only considers the second level (102 labels);
- Hierarchy Level 3: only considers the third level (89 labels);
- Hierarchy Level 4: only considers the fourth level (42 labels);

5.2. Balancing classes

Normally, real world information is sampled in an irregular or unbalanced way. Unbalanced classes are a potential obstacle for classification algorithms because they may hinder the construction of models that are able to correctly discriminate the majority set from the minority one [34]. However, in general, the minority class is the most interesting and valuable in terms of representing possible new knowledge.

Several studies have reported that many base classifiers perform better if they are applied to balanced datasets [35,34,36,37]. Therefore a solution for learning models from unbalaced datasets is to make an adjustment to the dataset to equalize the distribution of instances. An example using sampling would be removing instances of the majority class (undersampling), or by adding instances of the minority class (oversampling) [38].

There are many methods in the literature for balancing classes applied to single-label problems. However, they do not apply to multi-label problems directly. To measure the balancing factor from a multi-label dataset D, we propose (2), where

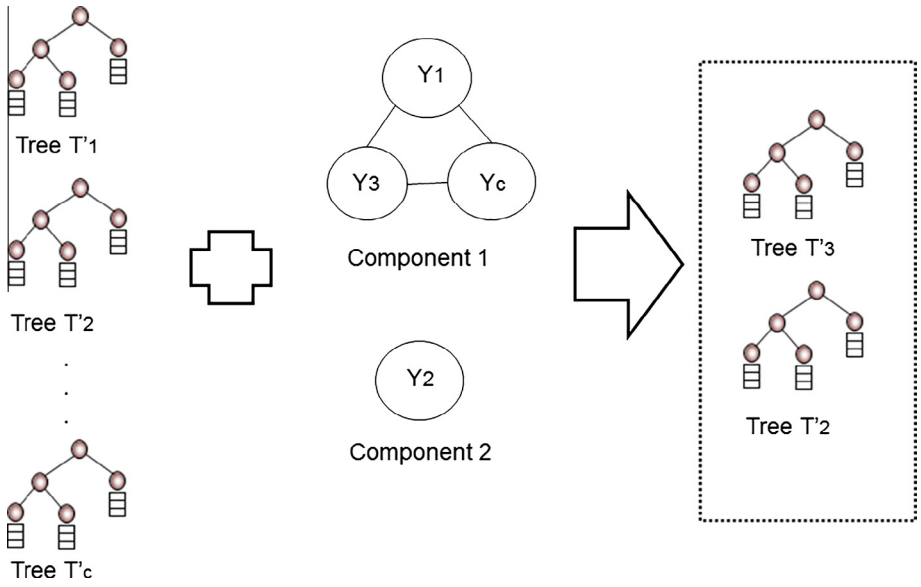

Fig. 4. BR-DT Algorithm – Step 3.

$n_0^k = n(Y_k = 0)$ is the number of instances labeled $Y_k = 0$, and $n_1^k = n(Y_k = 1)$ is the number of instances labeled $Y_k = 1$. This metric assumes values in the range $[0, 1]$, where higher values indicate more balanced class labels.

$$Balancing(D) = \frac{\sum_{k=1}^{c}(n_0^k \times n_1^k)}{N \times c} \tag{2}$$

Next, we describe some balancing strategies to solve multi-label problems. One way to solve the problem of unbalanced classes uses the algorithm-independent approach. In this case, a multi-label problem is transformed into a set of single-label problems, and thus it is possible to balance one label at a time.

In this study, we propose a balancing approach based on the Binary Relevance algorithm that uses both over- and undersampling as follows: first, the multi-label dataset D is transformed into a set of c single-label datasets D_k. Each D_k contains all attributes X, and only label Y_k ($k = 1, \ldots, c$). For binary classes, each dataset D_k has a majority class, and a minority class. In these cases, for each dataset D_k, undersampling is applied to instances belonging to the majority class, and oversampling is applied to instances belonging to the minority class. This strategy results in nearly balanced datasets.

5.3. Tree pruning

After building a decision tree, it is possible that the induced classifier is very specific for the training data. In this case, the classifier overfits the training data too well. As the training data is only a sample of all possible instances, it is possible to add branches to the tree that improve the performance on the training data while decreasing performance on other instances outside it. In this situation, the accuracy (or other measure) on an independent (unseen) dataset yields to a poor performance classifier [39]. In order to avoid overfitting the data, some inducers *prune* the tree after inducing it. This process reduces the number of internal test nodes thus reducing the tree complexity while giving a better performance than the original tree. There are several pruning methods such as error-complexity [13] and pessimistic error [6]. The latter was used in this study.

5.4. Evaluation metrics

The computation of multi-label evaluation metrics for model h on dataset D can be performed using two basic procedures called *macro-averaging*, and *micro-averaging* [9]. For a binary problem B, let $B(tp(Y_i), fp(Y_i), tn(Y_i), fn(Y_i))$ denote the number of true positives, false positives, true negatives, and false negatives for label Y_i, respectively. *Micro-averaging* values are computed globally on all labels, given by (3). On the other hand, *macro-averaging* metrics are calculated locally, according to (4).

$$\text{micro}(B) = B\left(\sum_{i=1}^{c} tp(Y_i), \sum_{i=1}^{c} fp(Y_i), \sum_{i=1}^{c} tn(Y_i), \sum_{i=1}^{c} fn(Y_i)\right) \tag{3}$$

$$\text{macro}(B) = \frac{1}{c}\sum_{i=1}^{c} B(tp(Y_i), fp(Y_i), tn(Y_i), fn(Y_i)) \tag{4}$$

As can be seen, *micro-averaging* considers all examples as having the same weight; *macro-averaging* considers all labels having the same weight, regardless of their frequency [40]. In the reported experiments, we selected the *micro-averaging* computation for the F-measure (7), defined in terms of precision (5) and recall (6).

$$\text{Precision}(h, D) = \frac{\sum_{i=1}^{c} tp(Y_i)}{\sum_{i=1}^{c} tp(Y_i) + \sum_{i=1}^{c} fp(Y_i)} \tag{5}$$

$$\text{Recall}(h, D) = \frac{\sum_{Y_i=1}^{c} tp(Y_i)}{\sum_{i=1}^{c} tp(Y_i) + \sum_{i=1}^{c} fn(Y_i)} \tag{6}$$

$$\text{F-measure}(h, D) = \frac{2}{\frac{1}{Precision(h,D)} + \frac{1}{Recall(h,D)}} \tag{7}$$

The *HammingLoss* (8) measures the average error for all predicted labels [41], where $A \Delta B$ represents the symmetric difference between sets A and B, and is equivalent to the exclusive or (XOR) logic operation. It ranges over $[0, 1]$, and small HammingLoss values indicate a better classification performance.

$$\text{HammingLoss}(h, D) = \frac{1}{N}\sum_{i=1}^{N} \frac{|y_i \Delta h(x_i)|}{c} \tag{8}$$

Table 2
F-measure for hierarchy level 1.

Dataset	BR-DT Pru	BR-DT Unpr-Pr(1)	BR	LP	RAkEL	MLkNN	Clus Pru	Clus Unpr
F-measure								
pheno	0.393	0.368	0.393	0.377	**0.397**	0.377	0.379	0.374
seq	0.457	0.234	0.457	0.413	**0.490**	0.435	0.405	0.408
church	**0.459**	0.318	0.390	0.397	0.390	0.386	0.387	0.374
cellcycle	**0.461**	0.101	0.426	0.371	0.426	0.403	0.388	0.376
derisi	0.400	0.390	**0.417**	0.370	0.385	0.405	0.399	0.375
eisen	0.501	0.313	0.498	0.468	**0.535**	0.515	0.474	0.479
gasch2	**0.461**	0.258	0.424	0.386	0.425	0.429	0.394	0.375
spo	**0.461**	0.362	0.379	0.369	0.396	0.407	0.386	0.375
gasch1	0.427	0.114	0.422	0.407	**0.457**	0.421	0.396	0.397
expr	0.380	0.122	0.429	0.389	**0.467**	0.422	0.397	0.400
Post-hoc test results								
BR-DT Pru	o	▲	△	▲	▽	△	▲	▲
BR-DT Unpr-Pr(1)	x	o	▼	▽	▼	▼	▼	▽
BR	x	x	o	▲	▽	△	△	▲
LP	x	x	x	o	▼	▽	▽	△
RAkEL	x	x	x	x	o	△	▲	▲
MLkNN	x	x	x	x	x	o	△	▲
Clus pru	x	x	x	x	x	x	o	△
Clus unpr	x	x	x	x	x	x	x	o
Average Rank	2.400	7.700	3.000	5.850	**2.300**	3.450	5.200	6.100

5.5. Experimental setup

The experiments were performed using the Weka library [8]. In the proposed BR-DT algorithm, decision trees were based on algorithm J48 [6] with *default* settings; pruned and unpruned trees were used to evaluate our proposal as explained in the following text. We evaluated our algorithm and five other algorithms: four from the Mulam library: Binary Relevance, Label Powerset [9], RAkEL (RAndom k-labELsets) [10] and MLkNN (Multi-Label k-Nearest Neighbours) [11]. Furthermore, the Clus library was also used for comparison.

The BR-DT algorithm was evaluated considering two situations (BR-DT Pru and BR-DT Unpr-Pr(1)), by performing two variations of pruning and balancing classes. In the first variation, BR-DT Pru, decision trees were induced unbalanced and trees were pruned in steps 1 and 3; in the second variation, BR-DT Unpr-Pr(1), decision trees were induced (from balanced samples) but not pruned in step 1, only in step 3. These variations were chosen to test the possibility of obtaining better results when generating trees in order to improve the connection between classes.

For BR, LP and RAkEL, the algorithm J48 was used with default settings, as was algorithm MLkNN. The algorithm Clus was used with reduced variance as heuristic, no binary split, and minimal weight equals 2. With these settings, two variations were used: Clus Pru (pruning method enable), and Clus Unpr (no pruning).

To analyze the performance, a 10-fold cross-validation was performed for each algorithm and each dataset, recording the metric F-measure described previously. To analyze significant results the Friedman test [42] was used, considering a significance level of 5%, and the Benjamini-Hochberg as the *post hoc* procedure [43].

6. Results and discussion

This section presents results concerning the F-Measure for each of the four levels of the hierarchy. We also show the average rank obtained by the Friedman. The best results for each dataset are shown in boldface and the best overall performance can be seen by analyzing the lower average rank. Furthermore, we also show the post hoc test results, where the symbol △ (▲) indicates that the variation of the specific line is better (significantly better) than

Table 3
F-measure for hierarchy level 2.

Datasets	BR-DT Pru	BR-DT Unpr-Pr(1)	BR	LP	RAkEL	MLkNN	Clus Pru	Clus Unpr
F-measure								
pheno	0.054	0.021	0.058	**0.094**	0.055	0.044	0.000	0.037
seq	0.070	0.056	0.221	0.160	**0.227**	0.085	0.002	0.157
church	0.049	0.032	0.095	0.098	**0.101**	0.010	0.000	0.108
cellcycle	0.060	0.020	**0.164**	0.131	0.160	0.077	0.000	0.103
derisi	0.052	0.021	0.102	**0.126**	0.114	0.070	0.000	0.109
eisen	0.068	0.034	0.267	0.210	**0.275**	0.179	0.000	0.200
gasch2	0.058	0.026	0.150	**0.155**	0.154	0.095	0.000	0.138
spo	0.056	0.047	**0.116**	0.112	0.115	0.076	0.000	0.102
gasch1	0.058	0.043	0.223	0.171	**0.230**	0.139	0.000	0.151
expr	0.071	0.045	0.221	0.164	**0.231**	0.112	0.000	0.152
Post-hoc test results								
BR-DT Pru	o	△	▼	▼	▼	▽	△	▽
BR-DT Unpr-Pr(1)	x	o	▼	▼	▼	▽	△	▼
BR	x	x	o	△	▽	▲	▲	△
LP	x	x	x	o	▽	▲	▲	△
RAkEL	x	x	x	x	o	▲	▲	△
MLkNN	x	x	x	x	x	o	▲	▽
Clus pru	x	x	x	x	x	x	o	▼
Clus unpr	x	x	x	x	x	x	x	o
Average Rank	5.700	6.900	2.300	2.400	**1.700**	5.200	8.000	3.800

Table 4
F-measure for hierarchy level 3.

Datasets	BR-DT Pru	BR-DT Unpr-Pr(1)	BR	LP	RAkEL	MLkNN	Clus Pru	Clus Unpr
F-measure								
pheno	**0.044**	0.002	0.013	0.002	0.012	0.004	0.000	0.000
seq	0.032	0.000	**0.136**	0.080	0.108	0.025	0.000	0.081
church	0.027	0.000	0.006	0.013	0.005	0.002	0.000	**0.033**
cellcycle	0.040	0.000	0.080	0.058	**0.073**	0.007	0.000	0.042
derisi	0.030	0.002	0.016	0.049	0.014	0.002	0.000	**0.045**
eisen	0.055	0.000	**0.132**	0.087	0.113	0.040	0.000	0.066
gasch2	0.042	0.000	0.067	**0.070**	0.052	0.064	0.000	0.042
spo	0.035	0.000	0.058	**0.064**	0.048	0.010	0.000	0.062
gasch1	0.036	0.000	**0.120**	0.090	0.092	0.037	0.000	0.054
expr	0.048	0.013	**0.133**	0.086	0.109	0.021	0.000	0.058
Post-hoc test results								
BR-DT Pru	o	▲	▽	▽	▽	△	▲	▽
BR-DT Unpr-Pr(1)	x	o	▼	▼	▼	▽	△	▼
BR	x	x	o	△	△	▲	▲	△
LP	x	x	x	o	△	▲	▲	△
RAkEL	x	x	x	x	o	△	▲	△
MLkNN	x	x	x	x	x	o	△	▽
Clus Pru	x	x	x	x	x	x	o	▼
Clus Unpr	x	x	x	x	x	x	x	o
Average Rank	4.250	7.150	**2.000**	2.750	3.100	5.450	7.600	3.700

the variation of the corresponding column, while the symbol ▽ (▼) indicates that the variation of the specific line is worse (significantly worse) than the variation of the corresponding column.

6.1. First hierarchy level (16 labels)

From Table 2, it is possible to note that the BR-DT Pru algorithm had the second best average rank; the RAkEL algorithm achieved the best performance. Furthermore, the post hoc test does not show any significant difference in performance between BR-DT Pru and RAkEL. It can also be observed that the BR-DT Pru algorithm has got significantly better performance than algorithms BR-DT Unpr-Pr(1), LP, Clus Pru and Clus Unpr. Considering the BR-DT Unpr-Pr(1) algorithm, it presented the worst performance being worse than any other algorithm, and significantly worse than algorithms BR, RAkEL, MLkNN and Clus Pru.

6.2. Second hierarchy level (102 labels)

From Table 3, one can note that the BR-DT Pru and the BR-DT Unpr-Pr(1) algorithms obtained the fifth and sixth best average ranks, respectively. However, the post hoc test shows that the BR-DT Pru algorithm is better than the Clus Pru algorithm. It is noteworthy that only BR-DT and Clus algorithms can produce models that can be interpreted by human experts. At this level, the Clus Pru algorithm only produced tree leaves classifying all labels as negative; therefore the F-measure for all datasets was zero, except for dataset 'Seq'.

6.3. Third hierarchy level (89 labels)

Table 4 shows that the BR-DT Pru and the BR-DT Unpr-Pr(1) algorithms had the fourth and seventh best average ranks, respectively. It is also possible to note that in the post hoc test BR-DT Pru

Table 5
F-measure for hierarchy level 4.

Datasets	BR-DT Pru	BR-DT Unpr-Pr(1)	BR	LP	RAkEL	MLkNN	Clus Pru	Clus Unpr
F-measure								
pheno	**0.037**	0.002	0.000	0.000	0.000	0.000	0.000	0.000
seq	0.029	0.000	0.007	0.082	0.083	0.000	0.000	**0.084**
church	0.020	0.000	0.002	0.000	0.000	0.000	0.000	**0.039**
cellcycle	0.044	0.000	**0.075**	0.044	0.044	0.002	0.000	0.051
derisi	0.026	0.002	0.000	0.045	0.000	0.000	0.000	**0.062**
eisen	0.049	0.000	0.098	**0.091**	0.002	0.002	0.000	0.069
gasch2	0.018	0.000	0.000	**0.056**	0.002	0.000	0.000	0.023
spo	0.027	0.024	0.002	0.054	0.002	0.002	0.000	**0.067**
gasch1	0.037	0.004	**0.077**	0.060	0.049	0.000	0.000	0.042
expr	0.037	0.000	0.098	**0.069**	0.065	0.000	0.000	0.056
Post-hoc test results								
BR-DT Pru	o	△	△	▽	△	▲	▲	▽
BR-DT Unpr-Pr(1)	x	o	▽	▼	▽	△	△	▼
BR	x	x	o	▽	△	▲	▲	▽
LP	x	x	x	o	△	▲	▲	▽
RAkEL	x	x	x	x	o	△	△	▽
MLkNN	x	x	x	x	x	o	△	▼
Clus Pru	x	x	x	x	x	x	o	▼
Clus Unpr	x	x	x	x	x	x	x	o
Average Rank	3.400	5.750	3.650	2.950	4.550	6.350	6.900	**2.450**

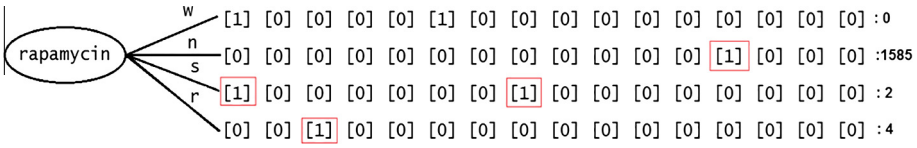

Fig. 5. Dataset 'Pheno' BR-DT Extended Tree.

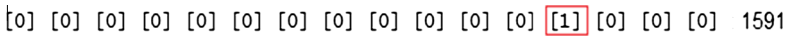

Fig. 6. Dataset 'Pheno' Clus Pru Extended Tree.

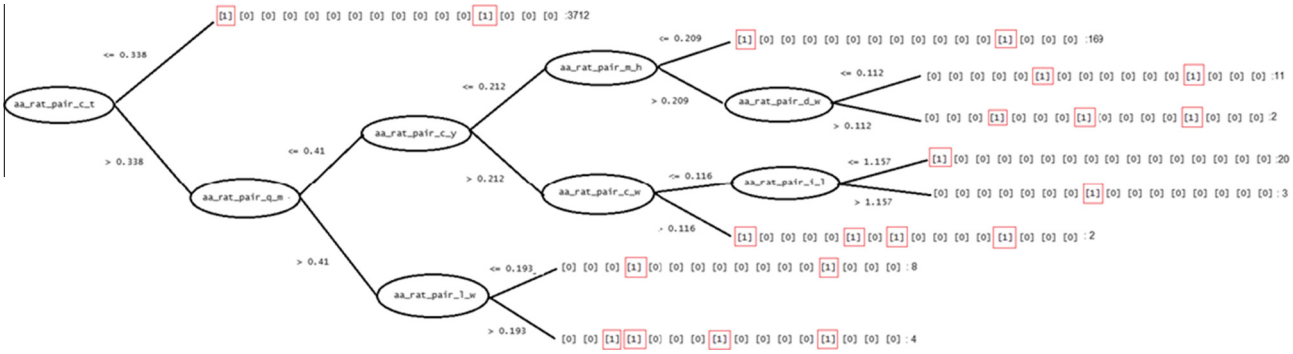

Fig. 7. Dataset 'Seq' BR-DT Extended Tree.

algorithm showed a significantly better performance than both the BR-DT Unpr-Pr(1) and the Clus Pru algorithms. Considering the BR-DT Unpr-Pr(1) algorithm, we can see that it presented the second worst performance, after Clus Pru. However, one can also observe in the post hoc test that the BR-DT Pru algorithm was significantly better than the Clus Pru algorithm, and that it was not significantly worse than the other algorithms. At this hierarchy level, the Clus Pru algorithm also produced tree leaves classifying all labels as negative; again the F-measure for all datasets was zero.

6.4. Fourth hierarchy level (42 labels)

As shown in Table 5, the BR-DT Pru and the BR-DT Unpr-Pr(1) algorithms obtained the third and fifth best average ranks, respectively. It can also be seen in the post hoc test that the BR-DT Pru algorithm had a better performance than the BR-DT Unpr-Pr(1), the RAkEL, the MLkNN, and the Clus Pru algorithms. The BR-DT Unpr-Pr(1) algorithm had a better performance than the MLkNN and the Clus Pru algorithms. At this hierarchy level, the Clus Pru

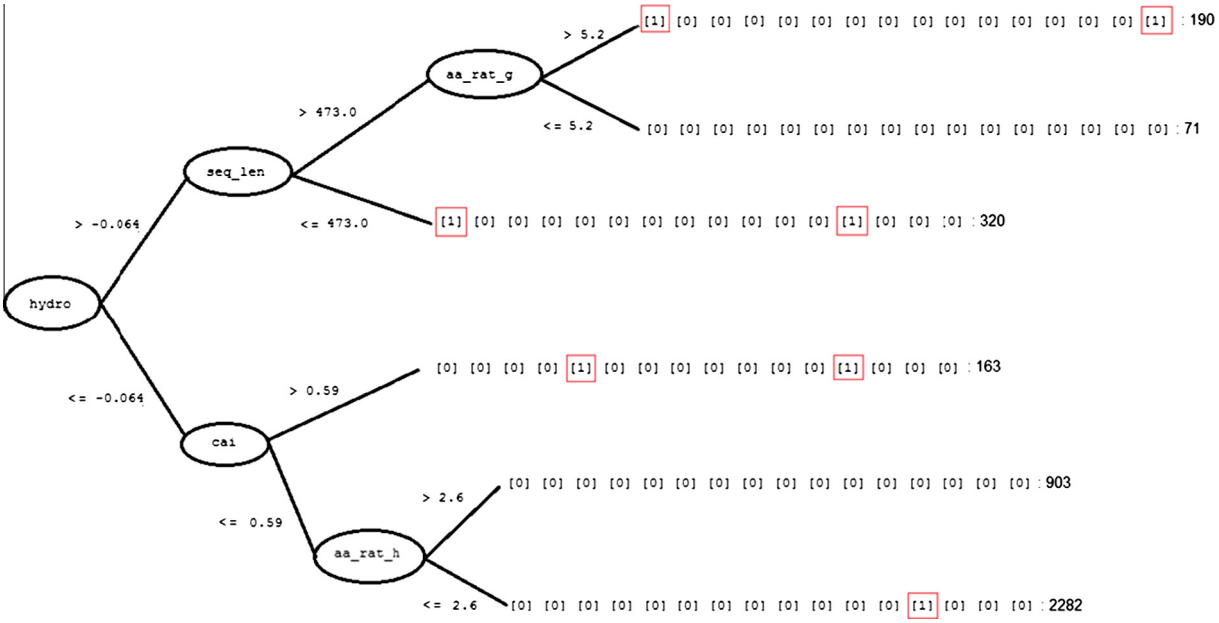

Fig. 8. Dataset 'Seq' Clus Pru Extended Tree.

algorithm also produced tree leaves, classifying all labels as negative; again the F-measure for all datasets was zero.

6.4.1. Biological results

This section presents models obtained from the datasets 'Pheno' and 'Seq', considering only the first level of the hierarchy, since models generated from other datasets were tree leaves, except for the data set 'SPO', where a very large model was obtained.

Analyzing the tree illustrated in Fig. 5 and generated by the BR-DT algorithm from dataset 'Pheno', we can observe the presence of only one attribute ('rapacymin'), which is a drug that inhibits the target of rapamycin (TOR) responsible for regulating growth, metabolism and aging. The model generated by the BR-DT algorithm contains four branches, and the branch that contains the greatest number of examples (99%) in the leaf is the one with value 'n = no data'. The branch with value 'w = wild type' contains no examples on its leaf. Therefore, it is interesting to analyze the last two branches: the branch with value 's = sensitive' predicts as 1 (present) the 'Metabolism' and the 'Cellular Communications' labels, and the branch with 'r = resistence' predicts to 1 (present) the label 'Cell Cycle'.

Observing the model obtained by the Clus Pru algorithm, illustrated in Fig. 6, a tree leaf only predicts the 'Subcellular localization' label as 1 (present). Therefore, any example predicted by this model for the 'Subcellular Localization' label is classified as 1 (present) and the other labels are classified as 0 (absent).

Fig. 7 shows the model generated by the BR-DT algorithm using the 'Seq' dataset in which it is possible to observe that the first two branches of the first tree are the most significant with respect to the number of examples (98.7%) in their leaves and they classify the 'Metabolism' and the 'Subcellular Localization' labels. Another observation is that all attributes appearing in the tree are aa_rat_pair_X_Y, meaning that the percentage of X and Y amino acid pairs appear consecutively, which is important to predict the CT (Cysteine and Threonine), GM (Glycine and Methionine), CY (Cysteine and Tyrosine), and MH (Methionine and Histidine) pairs.

Fig. 8 shows the model generated by the Clus Pru algorithm using the 'Seq' dataset. The last branch is the most significant with respect to the number of examples (58.0%) in its leaf, and it predicts the 'Subcellular Localization' label as 1 (present).

7. Conclusions

This paper presents a study on multi-label classification problem. In this scenario 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 proposed an adaptation for the Binary Relevance algorithm in order to overcome the disadvantage of the BR algorithm: we considered possible relationships among labels. This may improve the generalization ability of the model, and may possibly decrease the number of classifiers to be analyzed by human experts. When all the labels are related, our approach finds a single classifier (decision tree) that is able to classify them. Only when all labels are unrelated is the output model of our approach equal to BR (one decision tree for each label).

Experiments were conducted to compare the performance of our approach against others commonly found in the literature. The results lead us to conclude that our proposal has a performance comparable to that of other algorithms, as it obtained good results using the F-measure.

Based on these results we can observe that the variation in the BR-DT Pru algorithm in comparison with the Clus Pru algorithm at all levels had better results, although it did not obtain good results compared to the other algorithms in the second and third levels. One explanation for this result may be the high number of labels and few instances (per label) in the second and third levels.

Nowadays BR-DT only handles binary values. As future work, we intend to augment our proposal aiming to manipulate labels composed by more than two values. The current BR-DT algorithm only assigns a label to a class during classification and during the extension of the tree. After some iterations, this assignment should be rechecked to avoid error propagation. Another future work will be carried out in terms of improvement of execution time since BR-DT may be computationally expensive depending on the dataset.

Software Availability: Tool and thesis are available for downloading at http://dcm.ffclrp.usp.br/?pagina=dcm-eventos-pt&cod=168.

Acknowledgements

This research 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 National Institute of Science and Technology Program, INCT ADAPTA Project (Centre for Studies of Adaptations of Aquatic Biota of the Amazon).

Appendix A. Supplementary material

Supplementary data associated with this article can be found, in the online version, at http://dx.doi.org/10.1016/j.jbi.2014.12.011.

References

[1] Schietgat L, Vens C, Struyf J, Blockeel H, Kocev D, Dzeroski S. Predicting gene function using hierarchical multi-label decision tree ensembles. BMC Bioinf 2010;11(1):2+.
[2] Ruepp A, Zollner A, Maier D, Albermann K, Hani J, Mokrejs M, et al. The funcat, a functional annotation scheme for systematic classification of proteins from whole genomes. Nucl Acids Res 2004;32(18):5539–45.
[3] Clare A, King RD. Knowledge discovery in multi-label phenotype data. Lect Notes Comp Sci 2001:42–53.
[4] Suzuki E, Gotoh M, Choki Y. Bloomy decision tree for multi-objective classification. Princ Data Min Knowl Discov 2001:436–47. http://dx.doi.org/10.1007/3-540-44794-6_36.
[5] Cherman EA, Metz J, Monard MC. Métodos multirrótulo independentes de algoritmo: um estudo de caso. In: Anais da XXXVI Conferencia Latinoamericana de Informática (CLEI). Asuncion, Paraguay; 2010. p. 1–14.
[6] Quinlan JR. C4.5: programs for machine learning. San Francisco (CA): Morgan Kaufman; 1993.
[7] Tsoumakas G, Spyromitros-Xioufis E, Vilcek J, Vlahavas I. Mulan: a java library for multi-label learning. J Mach Learn Res 2011;12:2411–4.
[8] Witten IH, Frank E. Data mining: practical machine learning tools and techniques. In: Morgan Kaufmann series in data management systems. San Francisco (CA): Morgan Kaufman; 2005. <http://www.worldcat.org/isbn/0120884070>.
[9] Tsoumakas G, Katakis I, Vlahavas I. Mining multi-label data. Springer; 2010 [Ch. 34].
[10] Tsoumakas G, Vlahavas I. Random k-labelsets: an ensemble method for multilabel classification. Mach Learn: ECML 2007;2007:406–17.
[11] Zhang ML, Zhou ZH. Ml-knn: a lazy learning approach to multi-label learning. Pattern Recogn 2007;40(7):2038–48.
[12] Blockeel H, Raedt LD, Ramon J. Top-down induction of clustering trees. In: Proceedings of the 15th international conference on machine learning, ICML '98; 1998. p. 55–63.
[13] Breiman L, Friedman J, Stone CJ, Olshen R. Classification and regression trees. Pacific Grove (CA): Wadsworth & Books; 1984.
[14] Blockeel H, Schietgat L, Struyf J, Clare A, Dzeroski S. Hierarchical multilabel classification trees for gene function prediction. In: Probabilistic modeling and machine learning in structural and systems biology. Tuusula, Finland; 2006. p. 1–6.
[15] Alves RT, Delgado MR, Freitas AA. Multi-label hierarchical classification of protein functions with artificial immune systems. Adv Bioinf Comput Biol 2008:1–12.
[16] Stojanova D, Ceci M, Malerba D, Dzeroski S. Using ppi network autocorrelation in hierarchical multi-label classification trees for gene function prediction. BMC Bioinf 2013;14(1):285. http://dx.doi.org/10.1186/1471-2105-14-285. <http://www.biomedcentral.com/1471-2105/14/285>.

[17] Wan S, Mak M-W, Kung S-Y. R3p-loc: a compact multi-label predictor using ridge regression and random projection for protein subcellular localization. J Theoret Biol 2014;360(0):34–45. http://dx.doi.org/10.1016/j.jtbi.2014.06.031. <http://www.sciencedirect.com/science/article/pii/S0022519314003749>.
[18] Psomopoulos F, Mitkas P. Multi level clustering of phylogenetic profiles. In: IEEE international conference on bioinformatics and bioengineering (BIBE), 2010; 2010. p. 308–9.
[19] Vitsios D, Psomopoulos F, Mitkas P, Ouzounis C. Multi-genome core pathway identification through gene clustering. In: IFIP international federation for information processing; 2012. p. 545–55.
[20] Mitchell TM. Machine learning. USA: McGraw–Hill; 1997.
[21] Shen X, Boutell M, Luo J, Brown C. Multi-label Machine learning and its application to semantic scene classification. In: Storage and retrieval methods and applications for multimedia; 2004. p. 18–199.
[22] Clark P, Niblett T. The cn2 induction algorithm. In: Machine learning, vol. 3; 1989. p. 261–83.
[23] Mewes HW, Frishman D, Mayer KFX, Münsterkötter M, Noubibou O, Rattei T, et al. Mips: analysis and annotation of proteins from whole genomes. Nucl Acids Res 2004;32:41–4.
[24] Gasteiger E, Hoogland C, Gattiker A, Duvaud S, Wilkins M, Appel R, et al. Protein identification and analysis tools on the expasy server. In: The proteomics protocols handbook. Humana Press; 2005. p. 571–607.
[25] Kumar A, Cheung K, Ross-Macdonald P, Coelho P, Miller P, Snyder M. Triples: a database of gene function in saccharomyces cerevisiae. Nucl Acids Res 2000;28(1):81–4.
[26] Oliver S et al. A network approach to the systematic analysis of yeast gene function. Trends Genet: TIG 1996;12(7):241.
[27] Spellman P, Sherlock G, Zhang M, Iyer V, Anders K, Eisen M, et al. Comprehensive identification of cell cycle–regulated genes of the yeast saccharomyces cerevisiae by microarray hybridization. Mol Biol Cell 1998;9(12):3273–97.
[28] Roth F, Hughes J, Estep P, Church G. Finding dna regulatory motifs within unaligned noncoding sequences clustered by whole-genome mrna quantitation. Nat Biotechnol 1998;16:939.
[29] Clare A. Machine learning and data mining for yeast functional genomics, Ph.D. thesis. The University of Wales; 2003.
[30] Eisen M, Spellman P, Brown P, Botstein D. Cluster analysis and display of genome-wide expression patterns. Proc Nat Acad Sci 1998;95(25):14863.
[31] Gasch A, Spellman P, Kao C, Carmel-Harel O, Eisen M, Storz G, et al. Genomic expression programs in the response of yeast cells to environmental changes. Mol Biol Cell 2000;11(12):4241–57.
[32] Gasch A, Huang M, Metzner S, Botstein D, Elledge S, Brown P. Genomic expression responses to dna-damaging agents and the regulatory role of the yeast atr homolog mec1p. Mol Biol Cell 2001;12(10):2987–3003.
[33] Chu S, DeRisi J, Eisen M, Mulholland J, Botstein D, Brown P, et al. The transcriptional program of sporulation in budding yeast. Science 1998;282(5389):699.
[34] Tahir MA, Kittler J, Mikolajczyk K, Yan F. A multiple expert approach to the class imbalance problem using inverse random under sampling. In: Proceedings of the 8th international workshop on multiple classifier systems, MCS '09; 2009. p. 82–91.
[35] Laurikkala J. Improving identification of difficult small classes by balancing class distribution. In: Proceedings of the 8th conference on AI in medicine in Europe: artificial intelligence medicine, AIME '01; 2001. p. 63–6.
[36] Estabrooks A, Jo T, Japkowicz N. A multiple resampling method for learning from imbalanced data sets. Comput Intell 2004;20(1):18–36.
[37] Orriols A, Bernadó-Mansilla E. The class imbalance problem in learning classifier systems: a preliminary study. In: Proceedings of the 2005 workshops on genetic and evolutionary computation, GECCO '05. Washington (DC, USA): ACM; 2005. p. 74–8.
[38] Garcia V, S'anchez J, Mollineda R, Alejo R, Sotoca J. The class imbalance problem in pattern classification and learning. Patt Anal Learn Group 2007:283–91.
[39] Weiss SM, Indurkhya N. Predictive data mining: a practical guide. San Francisco (CA): Morgan Kaufman; 1998.
[40] Özgür A, Özgür L, Güngör T. Text categorization with class-based and corpus-based keyword selection. In: Proceedings of the 20th international conference on computer and information sciences, ISCIS'05; 2005. p. 606–15.
[41] Schapire RE, Singer Y. Boostexter: a boosting-based system for text categorization. Mach Learn 2000;39(2/3):135–68.
[42] Friedman M. A comparison of alternative tests of significance for the problem of m rankings. Ann Math Stat 1940;11(1):86–92.
[43] Benjamini Y, Hochberg Y. Controlling the false discovery rate: a practical and powerful approach to multiple testing. J Royal Stat Soc Ser B 1995;57:289–300.