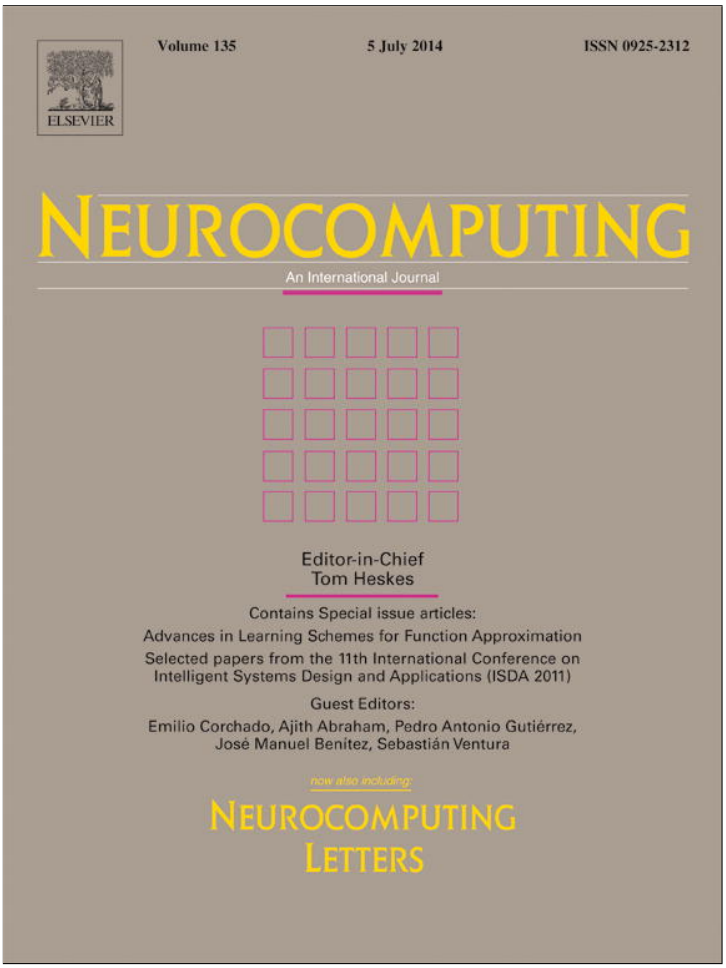

Neurocomputing 135 (2014) 3–12

Contents lists available at ScienceDirect

Neurocomputing

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

A framework for bottom-up induction of oblique decision trees

Rodrigo C. Barros [a,*,1], Pablo A. Jaskowiak [b,1], Ricardo Cerri [b], Andre C.P.L.F. de Carvalho [b]

[a] Faculdade de Informática, Pontifícia Universidade Católica do Rio Grande do Sul - Av. Ipiranga, 6681, 90619-900, Porto Alegre-RS, Brazil
[b] Instituto de Ciências Matemáticas e de Computação - ICMC, Universidade de São Paulo — Campus de São Carlos, Caixa Postal 668, 13560-970 São Carlos-SP, Brazil

ARTICLE INFO

Article history:
Received 17 October 2012
Received in revised form
14 January 2013
Accepted 19 January 2013
Available online 4 January 2014

Keywords:
Oblique decision trees
Bottom-up induction
Clustering

ABSTRACT

Decision-tree induction algorithms are widely used in knowledge discovery and data mining, specially in scenarios where model comprehensibility is desired. A variation of the traditional univariate approach is the so-called oblique decision tree, which allows multivariate tests in its non-terminal nodes. Oblique decision trees can model decision boundaries that are oblique to the attribute axes, whereas univariate trees can only perform axis-parallel splits. The vast majority of the oblique and univariate decision-tree induction algorithms employ a top-down strategy for growing the tree, relying on an impurity-based measure for splitting nodes. In this paper, we propose *BUTIF*—a novel *B*ottom-*U*p Oblique Decision-*T*ree *I*nduction *F*ramework. BUTIF does not rely on an impurity-measure for dividing nodes, since the data resulting from each split is known *a priori*. For generating the initial leaves of the tree and the splitting hyperplanes in its internal nodes, BUTIF allows the adoption of distinct clustering algorithms and binary classifiers, respectively. It is also capable of performing embedded feature selection, which may reduce the number of features in each hyperplane, thus improving model comprehension. Different from virtually every top-down decision-tree induction algorithm, BUTIF does not require the further execution of a pruning procedure in order to avoid overfitting, due to its bottom-up nature that does not overgrow the tree. We compare distinct instances of BUTIF to traditional univariate and oblique decision-tree induction algorithms. Empirical results show the effectiveness of the proposed framework.

1. Introduction

Decision trees are efficient nonparametric methods that can be applied to both classification and regression tasks. They are hierarchical data structures for supervised learning whereby the input space is split into local regions in order to predict the dependent variable (class) [1]. Decision trees are widely used in data mining and machine learning applications, mostly due to their intuitive hierarchical knowledge representation, which is easily assimilated by humans. Another reason for their popularity is their good predictive accuracy in several application domains, such as medical diagnosis and credit risk assessment [2].

A major issue in the induction of decision trees from data is selecting the attribute(s) for splitting a node into subsets. For the case of *axis-parallel* decision trees—also known as *univariate*—the tasks are to choose the attribute that better discriminates the input data. A decision rule based on the given attribute is thus generated, and the input data is filtered according to the outcomes of this rule. For *oblique* decision trees—commonly regarded as *multivariate*—the goal is to find a combination of attributes with good discriminatory power. Either way, both strategies (univariate and multivariate) are concerned with ranking attributes quantitatively.

Oblique decision trees are not as popular as the univariate trees, mainly because they are harder to interpret. Nevertheless, researchers reckon that multivariate splits can improve the performance of the tree in several data sets, while generating smaller trees [4–6]. Clearly, there is a tradeoff to consider in allowing multivariate tests: simple tests may result in large trees that are hard to understand, yet multivariate tests may result in small trees with tests that are hard to understand [7]. Fig. 1 presents a univariate decision tree that needs 5 splits to separate data from two classes, which could be easily separated by a single oblique split.

A decision tree with multivariate splits is able to produce polygonal (polyhedral) partitions of the attribute space (hyperplanes at an oblique orientation to the attribute axes), whereas univariate trees can only produce hyper-rectangles parallel to the attribute axes. The tests at each internal node of the tree have the following form:

$$w_0 + \sum_{i=1}^{m} w_i x_i \leq 0 \quad (1)$$

* Corresponding author.
E-mail address: rcbarros@gmail.com (R.C. Barros).
[1] Both authors contributed equally to this work.

0925-2312/$ - see front matter © 2014 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.neucom.2013.01.067

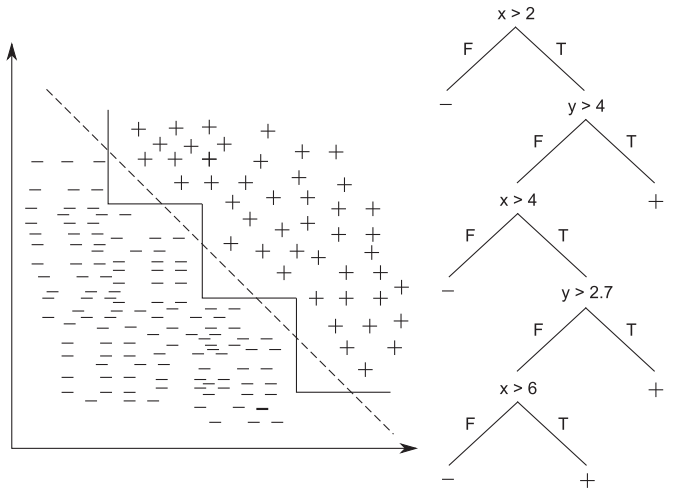

Fig. 1. A bivariate input space composed by two attributes (x and y) in a binary-class problem (left) and its corresponding univariate decision tree (right). The single oblique split that separates the two classes is $x+y \leq 8$ (left—dashed line). Adapted from [3].

In Eq. (1), w_i is a real-valued coefficient associated to the ith attribute x_i, and w_0 is the disturbance coefficient of the performed test.

Several approaches have been proposed in the last few decades for the induction of oblique and axis-parallel decision trees, such as hybrid induction [8], evolutionary induction [9–17], and ensemble of trees [18]. Notwithstanding, no strategy has been more successful in generating accurate decision trees with low computational effort than the greedy top-down induction strategy. The most well-known algorithms for decision tree induction indeed implement this strategy, *e.g.*, CART [19], C4.5 [20], and OC1 [21]. These algorithms make use of impurity-based criteria to decide which attribute(s) will split the data into purer data subsets—a subset is said to be *pure* when its instances belong to a single class. Since these algorithms are top-down, it is not possible to know *a priori* which instances will go to each subset of a partition. Therefore, trees are usually grown until every leaf node is pure, and a pruning method is later employed to avoid overfitting.

Studies that implement a bottom-up strategy are very rare in the literature. The key ideas behind bottom-up induction were first presented by Landeweerd et al. [22], in which a decision tree was growth from the leaves to the root, assuming that each class is represented by a leaf node, and that the closest nodes according to the Mahalanobis distance are recursively merged into a parent node. This approach presents several deficiencies, *e.g.*, it requires a single leaf per class, which means that that binary-class problems will always be modeled by a 3-node tree. This is quite problematic since there are complex binary-class problems in which a 3-node decision tree cannot accurately model the attribute space. This deficiency is probably one of the reasons for demotivating researchers to further investigate bottom-up induction of decision trees. Another shortcoming may be the extra computational cost required to compute the Mahalanobis distance.

In this paper, we propose a novel *framework* called BUTIF (*B*ottom-*U*p Oblique Decision-*T*ree *I*nduction *F*ramework) for solving the deficiencies of the pioneering bottom-up approach from Landeweerd et al. [22]. Note that this paper is a thorough extension of a previous conference paper [15], in which a particular instance of BUTIF was presented. We now focus our contribution in a framework instead of a particular bottom-up tree induction algorithm, mainly because:

- Any clustering algorithm may be employed to allow each class to be partitioned and represented by more than one leaf node.
- Any binary classifier that generates hyperplanes may be employed for creating the rule at each internal node of the oblique decision tree.
- Any feature selection method may be used within each non-terminal node for generating simpler rules.

In order to assess the performance of our framework, we instantiate distinct algorithms under its definition, by considering different clustering algorithms, binary classifiers and feature selection procedures. These algorithms are, in turn, evaluated in 35 real-world gene expression datasets. Empirical results provide good evidence of the framework applicability and effectiveness.

The remainder of the paper is organized as follows. In Section 2 we present studies related to our approach. In Section 3 we detail BUTIF, the proposed framework for generating oblique decision trees. In Section 4 we conduct a comparison among different BUTIF instances and traditional top-down decision-tree induction algorithms, namely, C4.5 [20], CART [19] and OC1 [21]. Additionally, we compare BUTIF to the recent Functional Trees [23]. Finally, in Section 5 we present the main conclusions and possibilities for future work.

2. Related work

There are many studies that propose top-down induction of oblique decision trees in the literature, and we briefly review some of them as follows.

The Classification and Regression Trees algorithm, also known as CART [19], is one of the first algorithms that allowed multivariate splits. CART employs a hill-climbing strategy with a backward attribute elimination to find good (albeit suboptimal) linear combinations of attributes in non-terminal nodes. It is a fully deterministic algorithm with no built-in mechanisms to escape local-optima.

Simulated Annealing of Decision Trees (SADT) [4] is a tree induction method that employs simulated annealing (SA) for finding good coefficient values for attributes in non-terminal nodes of decision trees. First, it places a hyperplane in a canonical location, and then iteratively perturbs the coefficients in small random amounts guided by the SA algorithm. Although SADT can eventually escape from local-optima, its efficiency is compromised, since it may consider tens of thousands of hyperplanes in a single node during annealing.

Oblique Classifier 1 (OC1) [21] is yet another top-down oblique decision-tree method. It is a thorough extension of CART's strategy, which presents the advantage of being more efficient than the previously described methods. It searches for the best univariate split as well as the best oblique split, and it only employs the oblique split when there is an improvement over the univariate split. OC1 uses both a deterministic heuristic search (as in CART) for finding local-optima and a non-deterministic search (as employed in SADT—though not SA) for escaping local-optima. Ittner [24] proposes using OC1 over an augmented attribute space, generating *non-linear decision trees*. The key idea is to "build" new attributes by considering all possible pairwise products and squares of the original set of m attributes.

A more recent approach for top-down oblique trees that employs SVM for generating hyperplanes is SVM-ODT [25]. It is an extension to the C4.5 algorithm, allowing the use of either an univariate split or a SVM-generated oblique split, according to the impurity-measure

values. Several methods are applied to avoid model overfitting, such as reduced-error pruning and MDL computation, resulting in an overly complex algorithm. Model overfitting is directly related to the own nature of the top-down strategy.

Another recently explored strategy for inducing oblique decision trees is the use of evolutionary algorithms (EAs). The reader can refer to [13] for studies that employ EAs for generating the hyperplanes in top-down oblique tree inducers.

Bottom-up induction of decision trees is practically not explored in the machine learning research community. The first work to present the concepts of bottom-up induction is Landeweerd et al. [22], where the authors propose growing a decision tree from the leaves to the root, assuming that each class is represented by a leaf node, and that the most similar nodes are recursively united into a parent node. This approach is too simplistic, because it allows only a single leaf per class, which means that binary-class problems will always be modeled by a 3-node tree. We believe that this deficiency is the main reason that has demotivated researchers to further investigate the bottom-up induction of decision trees in the past years.

The bottom-up strategy has only been employed for pruning decision trees [26], or for evaluating Alternating Decision Trees (ADTree) [27], issues that are not investigated in this paper. Our approach, presented in Section 3, intends to expand the work of Landeweerd et al. [22] in such a way that it can be effectively applied to distinct problem domains. Particularly, we provide an alternative to solve the one-leaf-per-class problem by performing clustering in each class of the data, and we deal with the hyperplane generation task by employing a binary classifier in each internal node.

3. BUTIF

In this work, we propose a novel bottom-up oblique decision-tree framework, called BUTIF (*B*ottom-*U*p Oblique Decision-*T*ree *I*nduction *F*ramework). Our motivation for building trees in a bottom-up fashion is twofold, as follows.

In a bottom-up approach, we have *a priori* information on which set of training instances belongs to a given node of the tree. It means that we know the result of each split even before generating the separating hyperplane. Indeed, BUTIF makes use of these *a priori* information for generating separating hyperplanes between instances from different classes. Hence, there is no need of relying on an impurity-measure to evaluate the goodness of a split, *i.e.*, we eliminate the need of another heuristic procedure.

The top-down induction strategy usually overgrows the decision tree until every leaf node is pure, and then a pruning procedure is responsible for simplifying the tree in order to avoid data overfitting. In bottom-up induction, a pruning step is not necessary because we are not overgrowing the tree. Since we start growing the tree from the leaves to the root, our approach reduces significantly the chances of overfitting by initially clustering data points.

3.1. Framework steps

Given a space of instances $\mathbf{X} = \{\mathbf{x}_1 \ldots \mathbf{x}_n\}$, $\mathbf{x}_i \in \Re^m$, and a set of classes $C = \{c_1 \ldots c_l\}$, BUTIF works according to the following steps (see Algorithm 1):

(1) Divide the training data into pure subsets, *i.e.*, instances belonging to the same subset have the same class label. Formally, for an l-class problem, generate l subsets $S_i = \{\mathbf{x} | \mathcal{C}(\mathbf{x}) = c_i\}$, where $\mathcal{C}(\mathbf{x}_j)$ returns the class label of instance $\mathbf{x}_j$ (procedure *butif*, line 4).

(2) Apply a clustering algorithm $\mathcal{G}$ over each subset S_i, $1 \le i \le l$. BUTIF allows any clustering algorithm to be employed. Preference should be given to clustering algorithms that are capable of automatically estimating the number of clusters from the data, otherwise a critical parameter is introduced to the end-user (procedure *butif*, line 5).

(3) Compute the centroid (cluster mean vector) of each cluster generated in step 2. Each of the previously generated clusters becomes a leaf node of the tree (procedure *butif*, lines 6–11).

(4) Generate a new internal node by merging the two most similar nodes, *i.e.*, those whose centroids are closest according to the Euclidean distance (procedure *merging*, lines 7–17). Once a node is created, its instances (that came up from its children) have their class label replaced by a new unique meta-class (procedure *merging*, line 18). During the merging procedure, two constraints are imposed: (i) only nodes whose instances are from different classes are allowed to be merged; and (ii) nodes can be merged only once.

(5) Perform feature selection with algorithm $\mathcal{S}$ for rule simplification (procedure *merging*, line 19). This step is optional and aims at reducing the feature space of those instances that will be used for generating the node rule (hyperplane), eventually leading to smaller (and simpler) decision rules. Note that, even though algorithm $\mathcal{S}$ is used for reducing the instances' feature space, the original (complete) feature space is kept intact when sent to the parent nodes.

(6) Compute the new node's centroid and generate the separating hyperplane through a binary classifier $\mathcal{F}$ (procedure *merging*, lines 20 and 21). The resulting hyperplane (rule) is responsible for separating the training instances that arrive at this node. Recall that the node's children come from two different classes (for leaf nodes) or meta-classes (for internal nodes). Thence, the classifier needs to deal exclusively with binary problems, regardless of the number of classes in the original dataset, which means the application of state-of-the-art classifiers, such as Support Vector Machines (SVMs) is straightforward and does not rely on a majority voting scheme between pairwise class subsets. This is particularly interesting to preserve the interpretability of the generated hyperplane as a rule.

Steps 4–6 are repeated recursively until reaching the tree root.

Algorithm 1. BUTIF framework.

```
procedure butif(X, C, G, F, S)
input:  Dataset X
        Set of classes C
        Clustering algorithm G
        Binary classifier F
        Feature selection method S (optional)
output: The oblique decision tree
1   begin
2     Leaves ← ∅;
3     foreach class c_i ∈ C do
4       S_i ← {x | C(x) = c_i}
5       Partition ← result of clustering S_i with G
6       foreach cluster ∈ Partition do
7         create new leaf node n
8         n.instances ← instances from cluster
9         n.centroid ← mean vector of n.instances
10        n.class ← c_i
11        Leaves ← Leaves ∪ {n}
12    Tree ← merging(Leaves, F, S)
13    return Tree
    procedure merging(L)
    input: A list L of nodes
```

output: The oblique decision tree root node
begin
  if $|L| = 1$ **then**
    return *unique node in L*
  else
    $smallestDistance \leftarrow \infty$
    $distance \leftarrow 0$
    foreach $i, j \in L$ *with* $i \neq j$ **do**
      if $i.class \neq j.class$ **then**
        $distance \leftarrow$ distance between i and j
        if $distance < smallestDistance$ **then**
          $smallestDistance \leftarrow distance$
          $s1 \leftarrow i$
          $s2 \leftarrow j$
    create new internal node t
    $t.leftChild \leftarrow s1$
    $t.rightChild \leftarrow s2$
    $t.instances \leftarrow s1.instances \cup s2.instances$
    $t.class \leftarrow$ new meta−class
    $t.instances \leftarrow \mathcal{S}(t.instances)$(optional)
    $t.centroid \leftarrow$ mean vector of $t.instances$
    $t.hyperplane \leftarrow \mathcal{F}(t.instances)$
    $L \leftarrow L \cup \{t\}$
    $L \leftarrow L \backslash \{s1\}$
    $L \leftarrow L \backslash \{s2\}$
    return *merging*(L)

Fig. 2 depicts the execution steps of BUTIF. We believe that this approach presents the following advantages over the top-down methods:

- It can handle imbalanced-classes regardless of any explicitly provided cost matrix. Unlike top-down approaches, BUTIF always generates at least one leaf node per class, which guarantees that rare classes are represented in the predictive model.
- It can model problems in which one class is described by more than one probability distribution. By clustering each pure subset, we can identify possibly distinct probability distributions, and thus generate separating hyperplanes for the closest inter-class boundaries. Note that this is not trivially achieved by other methods, specially those whose design is based on impurity measures.

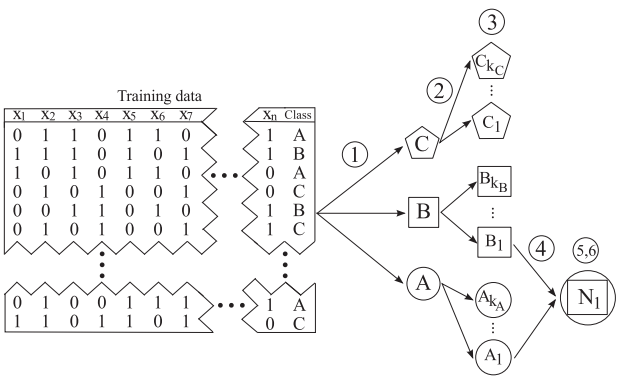

Fig. 2. Diagram of BUTIF's execution steps.

In Fig. 3, we detail how BUTIF would perform in a synthetic binary-class dataset with $m=2$. The original data is presented in (1), and the clustering step is performed in (2) within each class. Class (x) was clustered into two groups, as it was class (o), resulting in 4 leaf nodes. Next, the centroids for each node are calculated, and the closest nodes, according to the Euclidean distance, are merged. Recall that nodes can only be merged when their (meta-)classes are distinct. The new merged node has its centroid computed, its instances are associated to a meta-class, and the corresponding hyperplane is generated (3). Next, the algorithm merges the next two closest centroids, repeating the same procedure of creating a new node, assigning a new meta-class and generating a new hyperplane (4) and (5). The algorithm terminates when all nodes (apart from one, the root node) are merged, resulting in the oblique partitions presented in (6). In this particular example, we present hyperplane margins (dashed lines), since we are simulating a BUTIF instance in which a SVM classifier is being applied to generate the hyperplanes. In the following sections, we discuss and provide detail regarding alternatives that may be employed in each main step of BUTIF.

3.2. Generating leaves with clustering

As previously mentioned, the first and second steps of our framework consist in (i) stratifying the dataset, *i.e.*, dividing it according to the predefined class labels and (ii) applying a clustering algorithm $\mathcal{G}$ to each class separately in order to generate the leaves of the tree. Although there is only one way to carry out step (i), many different algorithms may take the place of $\mathcal{G}$ in step (ii). Well-known clustering algorithms such as k-means (KM) [28], hierarchical methods (HM) [29], and Expectation Maximization (EM) [30] may be employed, just to name a few. Additionally, one may employ different proximity measures among objects during the clustering process.

The choice of a particular clustering algorithm and proximity measure is usually related to the application at hand. For diverse application domains, comparative studies have been performed to indicate which proximity measure or clustering algorithm is preferable (*e.g.*, [31–33]), though we are aware that the flexibility given by the choice of $\mathcal{G}$ comes with the price of extensive experimentation and previous acquired knowledge. When *a priori* information is not available, employing classical clustering algorithms, such as KM and EM, is a reasonable choice, since these algorithms have been shown to be effective in diverse application areas.

After choosing the algorithm $\mathcal{G}$, the next issue to be addressed is the number of clusters to be generated for each class. Choosing the suitable number of clusters for a given dataset is an active research area in the clustering literature, and many different procedures may be employed for such a task [29]. Regarding EM, one may choose the number of clusters for each class by performing a 10-fold cross-validation procedure, evaluating the partition's stability according to the *loglikelihood* measure. For KM, one may use procedures like *Ordered Multiple Runs* (OMRk), in which the best partition is selected according to a relative clustering validity criterion, *e.g.*, the Silhouette Width Criterion [34] (see Algorithm 2). For a review of these criteria, we refer the reader to [35]. We would like to observe that the performance of these criteria is usually domain-related [35].

If the user has previously acquired domain knowledge, this knowledge can be used in order to define the leaves of the oblique decision tree. As an example, the user may possess a previously defined partition for a particular class. This partition can be simply provided as input for the BUTIF framework (overriding the algorithmic generation of per-class leaves). Nevertheless, this kind of information may be available only for a few classes, while the remainder should be normally provided as input for $\mathcal{G}$. As a practical example, consider the microarray dataset from Golub et al. [36], for which there are two classes: *ALL*, Acute Lymphoblastic Leukemia

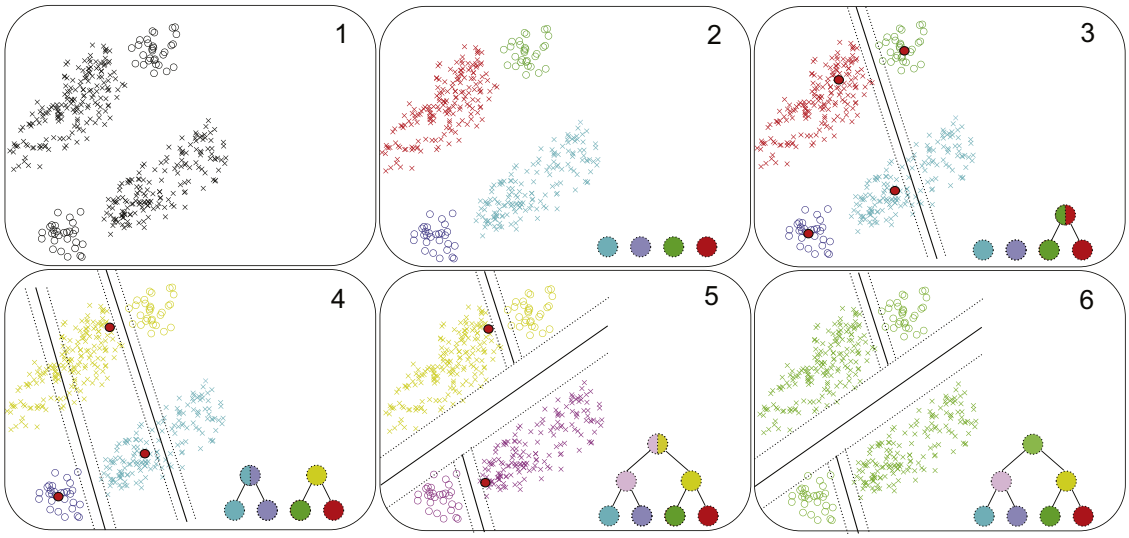

Fig. 3. BUTIF's execution in a synthetic 2-d binary-class dataset.

patients, and *AML*, patients with Acute Myeloid Leukemia. Biologically, *ALL* tumors can be further distinguished between B-Cell and T-Cell. If the user has such information at hand, this partition may be provided as input without the need of applying $\mathcal{G}$ over the class data, whereas the *AML* class can go through the whole process defined by BUTIF.

Algorithm 2. OMR*k* algorithm.

input: Training dataset **X**, number of executions p, maximum number of clusters k^{max}
output: The optimal number of clusters k^* and the resulting partition P

```
begin
    V* ← −∞
    for k = 2 to k^max do
        for i = 1 to p do
            Generate a random partition with k clusters
            Run k−means until convergence
            Compute V for the resulting partition
            if V > V* then
                V* ← V
                k* ← k
                Hold the resulting partition as P*
    return k*, P*
```

In brief, with respect to the leaf generation process, we have two distinct scenarios. On the one hand, a user without knowledge about a particular application domain may employ our framework with "default" choices, employing *classical* algorithms and standard parameters. On the other hand, a field practitioner with higher domain knowledge may carefully select a clustering algorithm, proximity measure, and even some of the initial clusters needed to build the bottom-up oblique decision tree. In this paper, we select two classical clustering algorithms to evaluate the performance of our framework —KM and EM. Results regarding these two instances of our framework are discussed in Section 4.

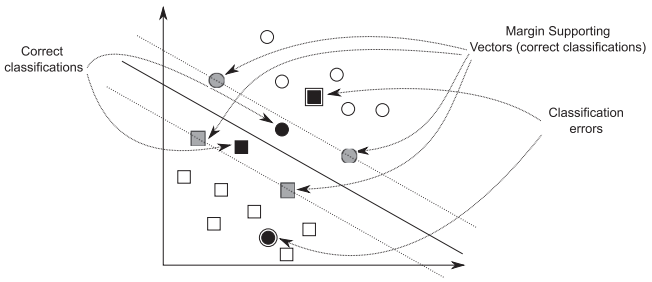

Fig. 4. A support vector machine.

3.3. Generating hyperplanes with binary classifiers

As previously discussed, each internal node of the oblique decision tree consists of a rule generated by a binary classifier regarding two distinct meta-classes (left and right paths). The user has to choose among classifiers that generate decision boundaries defined by hyperplanes. Within the internal nodes of the tree, a hyperplane is thus responsible for separating examples that should go to either the left or the right path of the tree. This process is performed until a leaf node is reached.

Support Vector Machines (SVMs) [37] are suitable classifiers to perform this task, considering that they are formulated to obtain the largest-possible linear margin that separates data belonging to two classes. SVMs with soft margins are particularly interesting because their constraints allow few examples to stay along the margins of the hyperplane built upon support vectors from two different classes. This makes the classifier more robust to outliers and noise. It is illustrated in Fig. 4.

Once again, different choices are available regarding the classifier that is going to be employed for the hyperplane generation. One may employ classifiers such as the already mentioned Support Vector Machines (SVMs) [37], Logistic Regression (LR) [38], and Fisher Linear Discriminant Analysis (FLDA) [39], just to mention a few. In this paper, we select two classifiers to demonstrate the effectiveness of BUTIF: SVMs and LR. Experimental results regarding these two instances of our framework (one per classifier) are presented and discussed in Section 4.

3.4. Performing embedded feature selection

After selecting a clustering algorithm and classification algorithm, the bottom-up tree induction process can take place. Although it is not mandatory, one may also employ different feature selection algorithms [40] during the tree induction process. The reasons to employ feature selection strategies are manifold. For instance, feature selection may be employed to reduce the final number of features needed by the tree, or to obtain simpler rules in each of its internal nodes. Feature selection may also reduce overfitting, thus improving model generalization and consequently its performance in unseen data. Note that when feature selection is performed, it is carried out internally in each node, before the hyperplane of the respective node is generated.

To demonstrate the effects of feature selection and how it can improve the results obtained by a BUTIF instance, we verify the effectiveness of a well-established feature selection method—Correlation-based Feature Selection (CFS) [41]—within distinct instances of BUTIF. Once more, we would like to keep in mind that different options may be implemented regarding the feature selection step of BUTIF.

4. Experimental analysis

4.1. Datasets

In order to evaluate our novel framework, we use a set of 35 publicly available datasets from microarray gene expression data, described in [42]. In brief, microarray technology enables expression level measurement for thousands of genes in parallel, given a biological tissue of interest. Once combined, results from a set of microarray experiments produce a gene expression dataset. The datasets employed here are related to different types or subtypes of cancer, *e.g.*, patients with prostate, lung, skin, and other types of cancer. These datasets comprehend the two flavors in which microarray technology is generally available: single channel (21 datasets) and double channel (14 datasets) [43]. Hereafter, we refer to single channel microarrays as Affymetrix (Affy) and double channel microarrays as cDNA, since the data were collected using either of these technologies [42]. Our task is to classify different examples (instances) according to their gene (attribute) expression levels. The main characteristics of the 35 datasets are summarized in Table 1.

4.2. Algorithms

We compared BUTIF to four different classification algorithms: Oblique Classifier 1 (OC1) [21] (oblique trees), Functional Trees (FT) [23] (logistic regression trees), Classification and Regression Trees (CART) [19] (univariate trees), and C4.5 [20] (univariate trees). For the experiments, we used the following implementations:

- OC1—publicly available code at the authors′ website: ⟨http://cbcb.umd.edu/~salzberg/announce-oc1.html⟩;
- FT, CART, and C4.5—publicly available codes implemented in java and found within the Weka toolkit [44].

We used the default values for the free parameters. Each classifier was evaluated regarding its predictive generalization

Table 1
Summary of the gene expression datasets.

Dataset ID	Dataset name	Type	Number of instances	Number of classes	Class distribution	Number of genes
1	alizadeh-v1	cDNA	42	2	21(2)	1095
2	alizadeh-v2	cDNA	62	3	42–11–9	2093
3	alizadeh-v3	cDNA	62	4	21(2)–11–9	2093
4	armstrong-v1	Affy	72	2	48–24	1081
5	armstrong-v2	Affy	72	3	28–24–20	2194
6	bhattacharjee	Affy	203	5	139–21–20–17–6	1543
7	bittner	cDNA	38	2	19(2)	2201
8	bredel	cDNA	50	3	31–14–5	1739
9	chen	cDNA	180	2	104–75	85
10	chowdary	Affy	104	2	62–42	182
11	dyrskjot	Affy	40	3	20–11–9	1203
12	garber	cDNA	66	4	40–17–5–4	4553
13	golub-v1	Affy	72	2	47–25	1877
14	golub-v2	Affy	72	3	38–25–9	1877
15	gordon	Affy	181	2	150–31	1626
16	khan	cDNA	83	4	29–25–18–11	1069
17	laiho	Affy	37	2	29–8	2202
18	lapointe-v1	cDNA	69	3	39–19–11	1625
19	lapointe-v2	cDNA	110	4	41–39–19–11	2496
20	liang	cDNA	37	3	28–6–3	1411
21	nutt-v1	Affy	50	4	15–14(2)–7	1377
22	nutt-v2	Affy	28	2	14(2)	1070
23	nutt-v3	Affy	22	2	15–7	1152
24	pomeroy-v1	Affy	34	2	25–9	857
25	pomeroy-v2	Affy	42	5	10(3)–8–4	1379
26	ramaswamy	Affy	190	14	30–22–20–11(8)–10(3)	1363
27	risinger	cDNA	42	4	19–13–7–3	1771
28	shipp	Affy	77	2	58–19	798
29	singh	Affy	102	2	52–50	339
30	su	Affy	174	10	28–27–26(2)–23–12–11–8–7–6	1571
31	tomlins-v1	cDNA	104	5	32–27–20–13–12	2315
32	tomlins-v2	cDNA	92	4	32–27–20–13	1288
33	west	Affy	49	2	25–24	1198
34	yeoh-v1	Affy	248	2	205–43	2526
35	yeoh-v2	Affy	248	6	79–64–43–27–20–15	2526

capability (accuracy and F-Measure), which was estimated with the use of a 10-fold cross-validation procedure.

Since we are introducing a *Framework*, we present and evaluate distinct instances of BUTIF by instantiating Algorithm 1:

- $\mathcal{G}$—we evaluated two clustering algorithms: Expectation Maximization (EM) and *k*-means (KM).
- $\mathcal{F}$—we evaluated two different classifiers: Support Vector Machines (SVM) and Logistic Regression (LR).
- $\mathcal{S}$—we evaluated the feature selection method called Correlation-based Feature Selection (CFS), and also the option of not employing feature selection.

Hence, we have $2 \times 2 \times 2 = 8$ BUTIF instances, as we further detail in Table 2.

Table 2
The eight different BUTIF instances.

Instance name	Clustering	Classifier	Feature selection
EM-LR	EM	LR	–
EM-LR$_s$	EM	LR	CFS
EM-SVM	EM	SVM	–
EM-SVM$_s$	EM	SVM	CFS
KM-LR	KM	LR	–
KM-LR$_s$	KM	LR	CFS
KM-SVM	KM	SVM	–
KM-SVM$_s$	KM	SVM	CFS

4.3. Statistical analysis

To evaluate the statistical significance of the experimental results, we present the results of statistical tests carried out according to the approach proposed by Demšar [45]. In brief, this approach seeks to compare multiple algorithms on multiple datasets, and it is based on the use of the Friedman test with a corresponding post hoc test. The Friedman test is a non-parametric counterpart of the well-known ANOVA. If the null hypothesis—which states that the classifiers under study present similar performances—is rejected, then we proceed with the Nemenyi test for pairwise comparisons.

4.4. Results

We present the accuracy values obtained by each one of the 12 classifiers (8 BUTIF instances and 4 baselines) in Table 3. Besides accuracy, we present the F-Measure values. By evaluating both accuracy and F-Measure, we believe that a more fair comparison among the algorithms is possible, since some of the datasets employed in our evaluation have unbalanced classes. F-Measure values are presented in Table 4. In both tables, we present the average values for each evaluation measure, which were obtained following the 10-fold cross-validation procedure, alongside with their respective standard deviation values. Additionally, in the bottom of each table, we show the average ranks of the methods for all datasets. A smaller average rank indicates a better classifier.

The experimental results from both analyses were very consistent overall. The best results regarding either evaluation measure were achieved by two of the BUTIF instances, namely KM-LR$_s$ and KM-SVM$_s$. These results not only provide good evidence in favor of

Table 3
Results regarding accuracy for the 35 datasets. Table shows average values ($\pm$ s.d.) obtained with 10-fold cross validation.

ID	EM-LR	EM-LR$_s$	KM-LR	KM-LR$_s$	EM-SVM	EM-SVM$_s$	KM-SVM	KM-SVM$_s$	FT	CART	J48	OC1
1	0.98 ± 0.02	0.93 ± 0.05	0.94 ± 0.04	1.00 ± 0.00	0.95 ± 0.03	0.97 ± 0.03	0.94 ± 0.04	1.00 ± 0.00	0.84 ± 0.19	0.71 ± 0.16	0.68 ± 0.20	0.72 ± 0.19
2	0.95 ± 0.04	0.96 ± 0.05	0.94 ± 0.09	0.97 ± 0.06	0.98 ± 0.03	0.98 ± 0.02	0.94 ± 0.10	0.97 ± 0.06	0.97 ± 0.07	0.92 ± 0.11	0.86 ± 0.15	0.87 ± 0.12
3	0.86 ± 0.04	0.95 ± 0.06	0.82 ± 0.08	0.97 ± 0.07	0.93 ± 0.05	0.95 ± 0.04	0.87 ± 0.13	0.97 ± 0.06	0.87 ± 0.12	0.69 ± 0.15	0.71 ± 0.15	0.66 ± 0.20
4	0.98 ± 0.01	0.97 ± 0.02	0.95 ± 0.02	1.00 ± 0.00	0.94 ± 0.03	0.93 ± 0.03	0.92 ± 0.03	1.00 ± 0.00	0.97 ± 0.06	0.90 ± 0.07	0.89 ± 0.06	0.56 ± 0.23
5	0.87 ± 0.06	0.93 ± 0.06	0.88 ± 0.02	1.00 ± 0.00	0.85 ± 0.08	0.89 ± 0.05	0.88 ± 0.01	1.00 ± 0.00	0.92 ± 0.07	0.85 ± 0.05	0.83 ± 0.05	0.76 ± 0.08
6	0.92 ± 0.09	0.89 ± 0.08	0.91 ± 0.09	1.00 ± 0.00	0.92 ± 0.09	0.91 ± 0.09	0.94 ± 0.08	1.00 ± 0.00	0.94 ± 0.05	0.89 ± 0.10	0.89 ± 0.08	0.82 ± 0.07
7	0.84 ± 0.04	0.74 ± 0.03	0.79 ± 0.06	0.96 ± 0.03	0.74 ± 0.03	0.63 ± 0.07	0.77 ± 0.02	1.00 ± 0.00	0.88 ± 0.18	0.53 ± 0.18	0.49 ± 0.16	0.74 ± 0.13
8	0.74 ± 0.06	0.64 ± 0.17	0.71 ± 0.04	1.00 ± 0.01	0.84 ± 0.09	0.78 ± 0.14	0.84 ± 0.09	1.00 ± 0.01	0.82 ± 0.20	0.76 ± 0.18	0.82 ± 0.11	0.79 ± 0.14
9	0.75 ± 0.10	0.69 ± 0.10	0.93 ± 0.04	1.00 ± 0.00	0.78 ± 0.11	0.71 ± 0.11	0.97 ± 0.01	0.99 ± 0.00	0.95 ± 0.05	0.85 ± 0.07	0.81 ± 0.07	0.80 ± 0.07
10	0.93 ± 0.05	0.93 ± 0.02	0.95 ± 0.03	0.97 ± 0.03	0.97 ± 0.02	0.95 ± 0.03	0.96 ± 0.02	0.97 ± 0.02	0.96 ± 0.05	0.97 ± 0.05	0.95 ± 0.05	0.66 ± 0.19
11	0.85 ± 0.09	0.77 ± 0.08	0.59 ± 0.04	1.00 ± 0.00	0.89 ± 0.05	0.89 ± 0.08	0.65 ± 0.07	0.99 ± 0.02	0.78 ± 0.14	0.75 ± 0.12	0.75 ± 0.24	0.68 ± 0.28
12	0.78 ± 0.03	0.87 ± 0.06	0.76 ± 0.03	0.94 ± 0.04	0.82 ± 0.02	0.88 ± 0.02	0.81 ± 0.03	0.96 ± 0.02	0.83 ± 0.14	0.76 ± 0.15	0.77 ± 0.22	0.57 ± 0.18
13	0.89 ± 0.01	0.91 ± 0.04	0.93 ± 0.03	1.00 ± 0.00	0.93 ± 0.05	0.95 ± 0.03	0.97 ± 0.02	1.00 ± 0.00	0.92 ± 0.10	0.86 ± 0.07	0.88 ± 0.08	0.61 ± 0.11
14	0.89 ± 0.04	0.87 ± 0.05	0.83 ± 0.08	1.00 ± 0.00	0.90 ± 0.05	0.97 ± 0.02	0.93 ± 0.03	1.00 ± 0.00	0.93 ± 0.09	0.88 ± 0.12	0.96 ± 0.09	0.53 ± 0.22
15	0.66 ± 0.10	0.79 ± 0.10	0.85 ± 0.08	1.00 ± 0.00	0.78 ± 0.09	0.74 ± 0.08	0.87 ± 0.07	1.00 ± 0.00	0.98 ± 0.03	0.96 ± 0.04	0.94 ± 0.05	0.71 ± 0.38
16	0.87 ± 0.08	0.99 ± 0.02	0.67 ± 0.09	0.95 ± 0.04	0.97 ± 0.02	0.99 ± 0.01	0.74 ± 0.08	0.95 ± 0.04	1.00 ± 0.00	0.85 ± 0.11	0.88 ± 0.12	0.74 ± 0.11
17	0.61 ± 0.21	0.47 ± 0.17	0.70 ± 0.08	1.00 ± 0.00	0.80 ± 0.14	0.53 ± 0.21	0.70 ± 0.10	1.00 ± 0.00	0.81 ± 0.24	0.82 ± 0.13	0.89 ± 0.14	0.92 ± 0.06
18	0.76 ± 0.09	0.62 ± 0.11	0.67 ± 0.05	1.00 ± 0.00	0.79 ± 0.09	0.69 ± 0.09	0.76 ± 0.13	0.96 ± 0.04	0.81 ± 0.17	0.78 ± 0.18	0.71 ± 0.14	0.39 ± 0.51
19	0.68 ± 0.08	0.75 ± 0.10	0.71 ± 0.02	0.97 ± 0.01	0.79 ± 0.10	0.74 ± 0.10	0.81 ± 0.07	0.97 ± 0.01	0.81 ± 0.16	0.68 ± 0.19	0.57 ± 0.11	0.68 ± 0.17
20	0.87 ± 0.07	0.86 ± 0.06	0.70 ± 0.12	0.99 ± 0.02	0.98 ± 0.03	0.98 ± 0.03	0.84 ± 0.12	1.00 ± 0.00	0.98 ± 0.08	0.71 ± 0.14	0.76 ± 0.19	0.83 ± 0.13
21	0.47 ± 0.10	0.59 ± 0.08	0.50 ± 0.08	1.00 ± 0.00	0.58 ± 0.08	0.53 ± 0.06	0.47 ± 0.11	0.98 ± 0.02	0.72 ± 0.14	0.54 ± 0.19	0.50 ± 0.17	0.81 ± 0.09
22	0.63 ± 0.12	0.77 ± 0.07	0.96 ± 0.05	0.82 ± 0.19	0.82 ± 0.09	0.64 ± 0.10	0.96 ± 0.04	0.82 ± 0.19	0.37 ± 0.07	0.77 ± 0.21	0.87 ± 0.17	0.96 ± 0.02
23	0.67 ± 0.11	0.67 ± 0.12	0.62 ± 0.26	0.62 ± 0.26	1.00 ± 0.00	0.71 ± 0.17	0.62 ± 0.26	0.62 ± 0.26	0.68 ± 0.23	0.68 ± 0.23	0.52 ± 0.41	0.83 ± 0.22
24	0.77 ± 0.03	0.71 ± 0.06	0.75 ± 0.13	1.00 ± 0.00	0.78 ± 0.04	0.75 ± 0.04	0.93 ± 0.08	1.00 ± 0.01	0.86 ± 0.15	0.84 ± 0.17	0.88 ± 0.16	0.24 ± 0.31
25	0.74 ± 0.07	0.52 ± 0.07	0.77 ± 0.07	1.00 ± 0.00	0.74 ± 0.07	0.63 ± 0.08	0.76 ± 0.10	1.00 ± 0.00	0.81 ± 0.22	0.57 ± 0.10	0.62 ± 0.17	0.80 ± 0.22
26	0.28 ± 0.07	0.29 ± 0.11	0.27 ± 0.11	0.97 ± 0.01	0.29 ± 0.09	0.30 ± 0.09	0.28 ± 0.08	0.87 ± 0.03	0.58 ± 0.11	0.60 ± 0.09	0.61 ± 0.05	0.73 ± 0.24
27	0.72 ± 0.06	0.68 ± 0.06	0.64 ± 0.03	1.00 ± 0.00	0.67 ± 0.06	0.61 ± 0.05	0.63 ± 0.07	1.00 ± 0.00	0.71 ± 0.19	0.52 ± 0.15	0.53 ± 0.19	0.57 ± 0.50
28	0.97 ± 0.04	0.85 ± 0.06	0.93 ± 0.09	1.00 ± 0.00	0.96 ± 0.03	0.95 ± 0.04	0.96 ± 0.04	0.99 ± 0.01	0.84 ± 0.10	0.77 ± 0.12	0.84 ± 0.16	0.63 ± 0.18
29	0.78 ± 0.09	0.72 ± 0.10	0.78 ± 0.06	1.00 ± 0.00	0.75 ± 0.10	0.77 ± 0.09	0.78 ± 0.07	0.98 ± 0.01	0.91 ± 0.10	0.74 ± 0.14	0.78 ± 0.12	0.36 ± 0.27
30	0.66 ± 0.07	0.60 ± 0.05	0.60 ± 0.05	0.99 ± 0.01	0.69 ± 0.07	0.62 ± 0.06	0.67 ± 0.06	0.99 ± 0.01	0.90 ± 0.04	0.74 ± 0.09	0.79 ± 0.09	0.71 ± 0.10
31	0.56 ± 0.06	0.68 ± 0.05	0.54 ± 0.10	0.99 ± 0.01	0.86 ± 0.09	0.81 ± 0.07	0.86 ± 0.10	0.99 ± 0.01	0.84 ± 0.05	0.58 ± 0.20	0.58 ± 0.18	0.76 ± 0.11
32	0.58 ± 0.11	0.62 ± 0.05	0.71 ± 0.09	1.00 ± 0.00	0.73 ± 0.12	0.70 ± 0.12	0.69 ± 0.09	0.98 ± 0.02	0.85 ± 0.13	0.59 ± 0.15	0.58 ± 0.17	0.65 ± 0.15
33	0.84 ± 0.03	0.76 ± 0.06	0.82 ± 0.08	1.00 ± 0.00	0.78 ± 0.07	0.75 ± 0.06	0.78 ± 0.07	1.00 ± 0.00	0.79 ± 0.16	0.89 ± 0.11	0.84 ± 0.13	0.68 ± 0.36
34	0.92 ± 0.03	0.95 ± 0.04	0.83 ± 0.07	1.00 ± 0.00	0.94 ± 0.06	0.96 ± 0.05	0.98 ± 0.02	1.00 ± 0.00	1.00 ± 0.01	0.99 ± 0.02	0.99 ± 0.02	0.99 ± 0.03
35	0.33 ± 0.17	0.61 ± 0.16	0.31 ± 0.11	1.00 ± 0.00	0.43 ± 0.16	0.59 ± 0.13	0.43 ± 0.15	0.91 ± 0.09	0.84 ± 0.06	0.75 ± 0.06	0.71 ± 0.11	0.75 ± 0.10
Rank	7.89	8.09	8.43	2.0	5.99	6.91	6.63	2.17	4.79	8.14	8.17	8.8

BUTIF, but also suggest that employing a reasonable feature selection method during the bottom-up induction process can also largely improve the final results. An interestingly observed result is that the embedded feature selection did not provide the same improvements when the EM clustering algorithm was employed. By further inspection, we verified that this happened mainly because KM generated more clusters (leaf nodes) than EM. Therefore, the feature selection procedure (CFS), when combined with KM, had to deal with more specialized nodes (nodes with less instances and thus with more chances of model overfitting). By reducing the number of features in these nodes, CFS was able to avoid this particular case of overfitting. At the same time, the decision tree benefited from the additional number of leaves generated by KM, providing evidence that a more robust tree hierarchy with simpler rules was preferable for the gene expression application domain.

Although we did not observe the same improvements with feature selection for EM, it is important to note that the average ranks of EM-LR$_s$ and EM-SVM$_s$ are better than three out of the four baseline algorithms. In fact, the only tree induction algorithm that presented better results than these two particular instances of BUTIF was FT. Regarding accuracy, seven out of eight BUTIF instances outperformed the traditional CART, J48, and OC1

Table 4
Results regarding F-Measure for the 35 datasets. Table shows average values (± s.d.) obtained with 10-fold cross validation.

ID	EM-LR	EM-LR$_s$	KM-LR	KM-LR$_s$	EM-SVM	EM-SVM$_s$	KM-SVM	KM-SVM$_s$	FT	CART	J48	OC1
1	0.98 ± 0.02	0.93 ± 0.04	0.94 ± 0.04	1.00 ± 0.00	0.95 ± 0.03	0.97 ± 0.03	0.94 ± 0.04	1.00 ± 0.00	0.82 ± 0.23	0.67 ± 0.21	0.63 ± 0.26	0.70 ± 0.20
2	0.96 ± 0.04	0.97 ± 0.05	0.94 ± 0.11	0.96 ± 0.09	0.98 ± 0.03	0.98 ± 0.02	0.92 ± 0.14	0.96 ± 0.09	0.96 ± 0.09	0.92 ± 0.10	0.84 ± 0.17	0.86 ± 0.13
3	0.86 ± 0.04	0.95 ± 0.05	0.82 ± 0.09	0.95 ± 0.09	0.93 ± 0.05	0.95 ± 0.04	0.85 ± 0.15	0.96 ± 0.09	0.86 ± 0.14	0.65 ± 0.18	0.68 ± 0.17	0.62 ± 0.23
4	0.98 ± 0.01	0.97 ± 0.02	0.96 ± 0.02	1.00 ± 0.00	0.96 ± 0.01	0.94 ± 0.02	0.94 ± 0.02	1.00 ± 0.00	0.97 ± 0.07	0.90 ± 0.07	0.88 ± 0.06	0.88 ± 0.08
5	0.89 ± 0.07	0.94 ± 0.06	0.91 ± 0.02	1.00 ± 0.00	0.86 ± 0.10	0.91 ± 0.06	0.90 ± 0.02	1.00 ± 0.00	0.91 ± 0.08	0.84 ± 0.05	0.83 ± 0.05	0.68 ± 0.13
6	0.92 ± 0.11	0.90 ± 0.10	0.92 ± 0.11	1.00 ± 0.00	0.91 ± 0.12	0.91 ± 0.12	0.93 ± 0.11	1.00 ± 0.00	0.93 ± 0.06	0.87 ± 0.12	0.89 ± 0.08	0.90 ± 0.07
7	0.87 ± 0.03	0.78 ± 0.05	0.82 ± 0.08	0.97 ± 0.02	0.78 ± 0.06	0.67 ± 0.11	0.80 ± 0.04	1.00 ± 0.00	0.87 ± 0.18	0.49 ± 0.21	0.45 ± 0.19	0.58 ± 0.16
8	0.78 ± 0.06	0.68 ± 0.16	0.71 ± 0.03	1.00 ± 0.01	0.82 ± 0.09	0.80 ± 0.13	0.79 ± 0.11	1.00 ± 0.01	0.80 ± 0.20	0.73 ± 0.19	0.81 ± 0.12	0.70 ± 0.12
9	0.72 ± 0.15	0.66 ± 0.15	0.93 ± 0.04	1.00 ± 0.00	0.76 ± 0.16	0.69 ± 0.15	0.97 ± 0.01	0.99 ± 0.00	0.95 ± 0.05	0.85 ± 0.07	0.81 ± 0.07	0.82 ± 0.07
10	0.95 ± 0.02	0.95 ± 0.02	0.96 ± 0.03	0.98 ± 0.01	0.98 ± 0.01	0.97 ± 0.01	0.97 ± 0.02	0.98 ± 0.01	0.96 ± 0.05	0.97 ± 0.05	0.95 ± 0.05	0.98 ± 0.04
11	0.86 ± 0.10	0.79 ± 0.10	0.61 ± 0.05	1.00 ± 0.00	0.91 ± 0.04	0.87 ± 0.10	0.69 ± 0.05	0.99 ± 0.02	0.71 ± 0.17	0.70 ± 0.14	0.72 ± 0.27	0.61 ± 0.12
12	0.78 ± 0.04	0.87 ± 0.06	0.72 ± 0.04	0.94 ± 0.04	0.82 ± 0.03	0.88 ± 0.02	0.81 ± 0.03	0.96 ± 0.03	0.80 ± 0.16	0.69 ± 0.17	0.75 ± 0.23	0.70 ± 0.10
13	0.91 ± 0.02	0.92 ± 0.04	0.94 ± 0.03	1.00 ± 0.00	0.93 ± 0.05	0.96 ± 0.02	0.97 ± 0.02	1.00 ± 0.00	0.91 ± 0.11	0.86 ± 0.07	0.87 ± 0.08	0.83 ± 0.13
14	0.90 ± 0.05	0.89 ± 0.05	0.84 ± 0.07	1.00 ± 0.00	0.90 ± 0.05	0.97 ± 0.02	0.94 ± 0.03	1.00 ± 0.00	0.93 ± 0.10	0.86 ± 0.14	0.96 ± 0.09	0.77 ± 0.10
15	0.69 ± 0.06	0.81 ± 0.10	0.87 ± 0.07	1.00 ± 0.00	0.80 ± 0.09	0.77 ± 0.07	0.89 ± 0.07	1.00 ± 0.00	0.98 ± 0.03	0.95 ± 0.05	0.95 ± 0.04	0.95 ± 0.02
16	0.87 ± 0.07	0.99 ± 0.02	0.72 ± 0.07	0.94 ± 0.05	0.98 ± 0.02	0.99 ± 0.01	0.78 ± 0.06	0.94 ± 0.05	1.00 ± 0.00	0.84 ± 0.10	0.88 ± 0.12	0.76 ± 0.16
17	0.60 ± 0.14	0.46 ± 0.14	0.73 ± 0.04	1.00 ± 0.00	0.83 ± 0.08	0.51 ± 0.16	0.74 ± 0.06	1.00 ± 0.00	0.78 ± 0.25	0.75 ± 0.18	0.85 ± 0.20	0.81 ± 0.23
18	0.76 ± 0.10	0.62 ± 0.11	0.69 ± 0.04	1.00 ± 0.00	0.79 ± 0.10	0.68 ± 0.10	0.74 ± 0.15	0.96 ± 0.04	0.78 ± 0.20	0.73 ± 0.20	0.68 ± 0.15	0.75 ± 0.08
19	0.65 ± 0.12	0.74 ± 0.11	0.70 ± 0.03	0.97 ± 0.01	0.76 ± 0.14	0.73 ± 0.11	0.80 ± 0.09	0.97 ± 0.01	0.78 ± 0.17	0.66 ± 0.20	0.58 ± 0.10	0.62 ± 0.20
20	0.86 ± 0.07	0.85 ± 0.07	0.71 ± 0.11	0.99 ± 0.02	0.97 ± 0.04	0.97 ± 0.04	0.85 ± 0.11	1.00 ± 0.00	0.96 ± 0.11	0.67 ± 0.21	0.77 ± 0.22	0.68 ± 0.26
21	0.52 ± 0.10	0.64 ± 0.07	0.55 ± 0.09	1.00 ± 0.00	0.64 ± 0.05	0.60 ± 0.05	0.52 ± 0.10	0.98 ± 0.02	0.69 ± 0.17	0.48 ± 0.19	0.46 ± 0.16	0.49 ± 0.02
22	0.70 ± 0.08	0.82 ± 0.06	0.96 ± 0.05	0.77 ± 0.25	0.87 ± 0.04	0.70 ± 0.12	0.96 ± 0.04	0.77 ± 0.25	0.20 ± 0.07	0.73 ± 0.26	0.87 ± 0.17	0.77 ± 0.27
23	0.71 ± 0.08	0.71 ± 0.08	0.50 ± 0.33	0.50 ± 0.33	1.00 ± 0.00	0.74 ± 0.11	0.50 ± 0.33	0.50 ± 0.33	0.57 ± 0.31	0.61 ± 0.29	0.50 ± 0.42	0.67 ± 0.30
24	0.85 ± 0.04	0.79 ± 0.10	0.82 ± 0.13	1.00 ± 0.00	0.85 ± 0.07	0.83 ± 0.08	0.94 ± 0.09	1.00 ± 0.01	0.83 ± 0.19	0.79 ± 0.22	0.85 ± 0.20	0.92 ± 0.19
25	0.79 ± 0.08	0.60 ± 0.09	0.80 ± 0.11	1.00 ± 0.00	0.79 ± 0.11	0.70 ± 0.11	0.79 ± 0.13	1.00 ± 0.00	0.75 ± 0.28	0.46 ± 0.12	0.52 ± 0.19	0.54 ± 0.19
26	0.30 ± 0.06	0.31 ± 0.10	0.30 ± 0.11	0.97 ± 0.01	0.33 ± 0.08	0.33 ± 0.08	0.32 ± 0.07	0.88 ± 0.02	0.57 ± 0.10	0.56 ± 0.10	0.56 ± 0.07	0.46 ± 0.14
27	0.75 ± 0.07	0.72 ± 0.08	0.67 ± 0.08	1.00 ± 0.00	0.70 ± 0.07	0.66 ± 0.07	0.66 ± 0.08	1.00 ± 0.00	0.69 ± 0.20	0.48 ± 0.19	0.53 ± 0.21	0.50 ± 0.20
28	0.97 ± 0.04	0.90 ± 0.07	0.92 ± 0.11	1.00 ± 0.00	0.97 ± 0.03	0.96 ± 0.04	0.96 ± 0.04	0.99 ± 0.01	0.83 ± 0.12	0.75 ± 0.13	0.83 ± 0.17	0.71 ± 0.11
29	0.83 ± 0.05	0.78 ± 0.07	0.84 ± 0.03	1.00 ± 0.00	0.82 ± 0.04	0.82 ± 0.05	0.83 ± 0.03	0.99 ± 0.01	0.91 ± 0.10	0.72 ± 0.18	0.77 ± 0.13	0.76 ± 0.11
30	0.60 ± 0.08	0.55 ± 0.07	0.56 ± 0.06	0.99 ± 0.01	0.63 ± 0.07	0.57 ± 0.08	0.62 ± 0.07	0.99 ± 0.01	0.88 ± 0.05	0.71 ± 0.11	0.78 ± 0.09	0.59 ± 0.16
31	0.65 ± 0.05	0.74 ± 0.09	0.61 ± 0.12	0.99 ± 0.01	0.88 ± 0.10	0.84 ± 0.09	0.87 ± 0.11	0.99 ± 0.01	0.82 ± 0.07	0.56 ± 0.20	0.56 ± 0.20	0.58 ± 0.13
32	0.67 ± 0.12	0.71 ± 0.08	0.74 ± 0.12	1.00 ± 0.00	0.78 ± 0.12	0.76 ± 0.12	0.75 ± 0.10	0.98 ± 0.02	0.83 ± 0.15	0.55 ± 0.13	0.57 ± 0.17	0.51 ± 0.15
33	0.84 ± 0.03	0.76 ± 0.05	0.80 ± 0.13	1.00 ± 0.00	0.76 ± 0.11	0.74 ± 0.10	0.76 ± 0.11	1.00 ± 0.00	0.77 ± 0.19	0.89 ± 0.12	0.81 ± 0.17	0.74 ± 0.29
34	0.94 ± 0.03	0.95 ± 0.05	0.85 ± 0.09	1.00 ± 0.00	0.93 ± 0.08	0.95 ± 0.05	0.98 ± 0.02	1.00 ± 0.00	1.00 ± 0.01	0.99 ± 0.02	0.99 ± 0.02	0.99 ± 0.02
35	0.34 ± 0.14	0.62 ± 0.14	0.35 ± 0.08	1.00 ± 0.00	0.43 ± 0.13	0.58 ± 0.10	0.43 ± 0.12	0.91 ± 0.06	0.83 ± 0.06	0.72 ± 0.06	0.70 ± 0.11	0.72 ± 0.11
Rank	7.2	7.56	7.99	2.13	5.53	6.64	6.59	2.24	5.46	9.04	8.49	9.14

Table 5
Statistical test summary for accuracy and F-Measure.

	EM-LR	EM-LR$_s$	KM-LR	KM-LR$_s$	EM-SVM	EM-SVM$_s$	KM-SVM	KM-SVM$_s$	FT	CART	J48	OC1
EM-LR				⊛				⊛	∗			
EM-LR$_s$		⊛				⊛	∗					
KM-LR				⊛				⊛	∗			
KM-LR$_s$												
EM-SVM				⊛				⊛				
EM-SVM$_s$				⊛				⊛				
KM-SVM				⊛				⊛				
KM-SVM$_s$												
FT				□				□				
CART				⊛	□			⊛	⊛			
J48				⊛	□			⊛	⊛			
OC1				⊛	□			⊛	⊛			

The symbol in each cell denotes that the column algorithm outperformed the one in the row w.r.t.: ∗ accuracy, □ F-Measure, ⊛ both.

algorithms. With respect to the F-Measure, all eight instances of our framework provided better results than CART, J48, and OC1. Note that, although FT obtained very good results, two of the BUTIF outperformed FT regarding the average ranks, namely KM-LR_s and KM-SVM_s.

In order to provide some reassurance about the validity of these results, we applied the Friedman and Nemenyi statistical tests as suggested in [45]. The tests were applied *separately* for the accuracy and F-Measure results. The Friedman test indicated the rejection of the null-hypothesis, *i.e.*, there is a statistically significant difference among the algorithms under comparison, for both accuracy (p-value$=2.60 \times 10^{-39}$) and F-Measure (p-value$=1.56 \times 10^{-38}$). Hence, we executed the Nemenyi post-hoc test for pairwise comparison. Results from this comparison can be seen in Table 5.

From Table 5, it is possible to see that two BUTIF instances, namely KM-LR_s and KM-SVM_s outperformed most baseline algorithms with statistical significance considering both evaluation measures. Regarding FT, these two BUTIF instances provided statistically significant improvements only w.r.t F-Measure. This is not unexpected, considering that accuracy may be a misleading criterion within unbalanced-class scenarios. What is actually happening is that FT is generating models that benefit the majority class(es) rather than generating a model capable of representing the rare classes. This, in turn, leads to high accuracy values, preventing the BUTIF instances to outperform FT in an accuracy-based comparison. However, a predictive model that neglects rare classes is seldom of practical use.

Note that none of the baseline classifiers outperformed instances from our framework with statistical significance, except for FT. However, it must be stressed again that FT outperformed two BUTIF instances only regarding the accuracy criterion, which is not a reliable evaluation measure in unbalanced scenarios, as previously discussed. We believe that these results provide solid evidence that BUTIF is a competitive alternative to traditional, well-established, and widely used decision-tree induction algorithms. BUTIF instances that employ the well-known k-means algorithm seem to be particularly promising, considering the large improvements in performance when compared to the baseline algorithms.

5. Conclusions and future work

In this work we have presented a novel *B*ottom-*U*p Oblique Decision-*T*ree *I*nduction *F*ramework (*BUTIF*). Our framework allows the user to choose different methods for clustering, feature selection, and binary classification, which are employed in each of its steps. Due to its bottom-up strategy for building the tree, BUTIF presents some interesting advantages over other top-down algorithms, such as robustness to imbalanced data and to certain types of data overfitting. Indeed, BUTIF does not require the further execution of a pruning procedure, which is the case of virtually every top-down decision-tree algorithm in the literature.

We have tested eight different BUTIF instances in 35 gene expression benchmarking datasets from [42]. Our experimental results indicated that BUTIF is able to outperform traditional and widely used algorithms, such as C4.5 [20], CART [19], OC1 [21], and FT [23]. Moreover, the difference in performance was shown to be statistically significant in favor of BUTIF, regarding accuracy and/or F-Measure. In addition, BUTIF offers much more flexibility than the baseline algorithms, since different methods can be internally combined, suiting the needs of the most diverse range of application domains.

We believe that this work has opened several venues for future research. Our framework provided competitive results to well-established algorithms for tree induction that have at least two decades of research, such as C4.5 and CART. We believe, however, that further exploration and research may still bring substantial improvements to it. For instance, our framework embraces only *oblique* decision trees. It would be interesting to consider simpler rules in internal nodes of the tree (univariate tests) and see if the results are comparable with those obtained with other univariate models. It would also be interesting to investigate the performance of BUTIF in more complex problems, such as hierarchical multi-label classification [46–48]. Finally, we hope that our results will instigate and promote more research on the topic of bottom-up tree induction, which has experienced a long time hiatus.

Acknowledgments

We would like to thank the Brazilian research agency FAPESP (Fundação de Amparo à Pesquisa do Estado de São Paulo) for funding this study.

References

[1] E. Alpaydin, Introduction to Machine Learning, 2nd ed., The MIT Press, Cambridge, Massachusetts, 2010.
[2] P.-N. Tan, M. Steinbach, V. Kumar, Introduction to Data Mining, Addison-Wesley, 2005.
[3] C.E. Brodley, P.E. Utgoff, Multivariate decision trees, Mach. Learn. 19 (1995) 45–77.
[4] D. Heath, S. Kasif, S. Salzberg, Induction of oblique decision trees, J. Artif. Intell. Res. 2 (1993) 1–32.
[5] S.K. Murthy, S. Kasif, S.S. Salzberg, A system for induction of oblique decision trees, J. Artif. Intell. Res. 2 (1994) 1–32.
[6] L. Rokach, O. Maimon, Top-down induction of decision trees classifiers–a survey, IEEE Trans. Syst. Man Cybern. C 35 (2005) 476–487 .
[7] P. Utgoff, C. Brodley, An incremental method for finding multivariate splits for decision trees, in: 7th International Conference on Machine Learning, pp. 58–65.
[8] B. Kim, D. Landgrebe, Hierarchical classifier design in high-dimensional numerous class cases, IEEE Trans. Geosci. Remote 29 (1991) 518–528.
[9] M. Basgalupp, R.C. Barros, A. de Carvalho, A. Freitas, D. Ruiz, Legal-tree: a lexicographic multi-objective genetic algorithm for decision tree induction, in: Proceedings of the 24th Annual ACM Symposium on Applied Computing, pp. 1085–1090.
[10] M.P. Basgalupp, A.C.P.L.F. de Carvalho, R.C. Barros, D.D. Ruiz, A.A. Freitas, Lexicographic multi-objective evolutionary induction of decision trees, Int. J. Bio-inspired Comput. 1 (2009) 105–117.
[11] R.C. Barros, M.P. Basgalupp, D.D. Ruiz, A.C.P.L.F.D. Carvalho, A.A. Freitas, Evolutionary model tree induction, in: Proceedings of the 25th Annual ACM Symposium on Applied Computing, pp. 1131–1137.
[12] R.C. Barros, D.D. Ruiz, M.P. Basgalupp, Evolutionary model trees for handling continuous classes in machine learning, Inf. Sci. 181 (2011) 954–971.
[13] R.C. Barros, M.P. Basgalupp, A.C.P.L.F. Carvalho, A.A. Freitas, A survey of evolutionary algorithms for decision-tree induction, IEEE Trans. Syst. Man Cybern.–Part C: Appl. Rev. 42 (2012) 291–312.
[14] R.C. Barros, M.P. Basgalupp, A.C. de Carvalho, A.A. Freitas, Towards the automatic design of decision tree induction algorithms, in: Proceedings of the 13th Annual Conference Companion on Genetic and Evolutionary computation, GECCO '11, ACM, New York, NY, USA, 2011, pp. 567–574.
[15] R.C. Barros, R. Cerri, P.A. Jaskowiak, A.C.P.L.F. de Carvalho, A bottom-up oblique decision tree induction algorithm, in: 11th International Conference on Intelligent Systems Design and Applications, pp. 450 –456.
[16] R.C. Barros, M.P. Basgalupp, A.C.P.L.F. de Carvalho, A.A. Freitas, A hyper-heuristic evolutionary algorithm for automatically designing decision-tree algorithms, in: Proceedings of the International Genetic and Evolutionary Computation Conference (ACM GECCO 2012), pp. 1237–1244.
[17] R.C. Barros, A.T. Winck, K.S. Machado, M.P. Basgalupp, A. de Carvalho, D.D. Ruiz, O. Norberto de Souza, Automatic design of decision-tree induction algorithms tailored to flexible-receptor docking data, BMC Bioinformat. 13 (2012) 310.
[18] L. Breiman, Random forests, Mach. Learn. 45 (2001) 5–32.
[19] L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Stone, Classification and Regression Trees, Wadsworth, Monterey, CA, 1984.
[20] J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann, San Francisco, CA, USA, 1993.
[21] S.K. Murthy, S. Kasif, S. Salzberg, R. Beigel, OC1: a randomized induction of oblique decision trees, in: AAAI, pp. 322–327.
[22] G. Landeweerd, T. Timmers, E. Gelsema, M. Bins, M. Halie, Binary tree versus single level tree classification of white blood cells, Pattern Recogn. 16 (1983) 571–577.
[23] J. Gama, Functional trees, Mach. Learn. 55 (2004) 219–250.
[24] A. Ittner, Non-linear decision trees-NDT, in: 13th International Conference on Machine Learning, pp. 1–6.

[25] V. Menkovski, I. Christou, S. Efremidis, Oblique decision trees using embedded support vector machines in classifier ensembles, in: 7th IEEE International Conference on Cybernetic Intelligent Systems, pp. 1–6.
[26] F. Esposito, D. Malerba, G. Semeraro, A comparative analysis of methods for pruning decision trees, IEEE Trans. Pattern Anal. 19 (1997) 476–491.
[27] B. Yang, T. Wang, D. Yang, L. Chang, BOAI: fast alternating decision tree induction based on bottom-up evaluation, in: Lecture Notes in Computer Science, Springer, 2008, pp. 405–416.
[28] J.B. MacQueen, Some methods for classification and analysis of multivariate observations, in: L.M.L. Cam, J. Neyman (Eds.), Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, vol. 1, University of California Press, 1967, pp. 281–297.
[29] R. Xu, D. Wunsch, Clustering, Wiley-IEEE Press, Hoboken, NJ, 2009.
[30] A.P. Dempster, N.M. Laird, D.B. Rubin, Maximum likelihood from incomplete data via the EM algorithm, J. R. Stat. Soc. 39 (1977) 1–38.
[31] P.A. Jaskowiak, R.J.G.B. Campello, T.F. Covoes, E.R. Hruschka, A comparative study on the use of correlation coefficients for redundant feature elimination, in: Brazilian Symposium on Neural Networks, pp. 13–18.
[32] K.W. Boyack, D. Newman, R.J. Duhon, R. Klavans, M. Patek, J.R. Biberstine, B. Schijvenaars, A. Skupin, N. Ma, K. Börner, Clustering more than two million biomedical publications: comparing the accuracies of nine text-based similarity approaches, PLoS ONE 6 (2011) e18029.
[33] P.A. Jaskowiak, R.J.G.B. Campello, I.G. Costa, Evaluating correlation coefficients for clustering gene expression profiles of cancer, in: Brazilian Symposium on Bioinformatics, Lecture Notes in Computer Science, vol. 7409, Springer, Berlin/Heidelberg, 2012, pp. 120–131.
[34] L. Kaufman, P.J. Rousseeuw, Finding Groups in Data: An Introduction to Cluster Analysis, Wiley, 1990.
[35] L. Vendramin, R.J.G.B. Campello, E.R. Hruschka, Relative clustering validity criteria: a comparative overview, Stat. Anal. Data Min. 3 (2010) 209–235.
[36] T.R. Golub, D.K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J.P. Mesirov, H. Coller, M.L. Loh, J.R. Downing, M.A. Caligiuri, C.D. Bloomfield, E.S. Lander, Molecular classification of cancer: class discovery and class prediction by gene expression monitoring, Science 286 (1999) 531–537.
[37] V.N. Vapnik, The Nature of Statistical Learning Theory, Springer-Verlag New York, Inc., New York, NY, USA, 1995.
[38] D.W. Hosmer, S. Lemeshow, Applied logistic regression (Wiley Series in Probability and Statistics), 2nd ed., Wiley-Interscience Publication, 2000.
[39] R.A. Fisher, The use of multiple measurements in taxonomic problems, Ann. Eugen. 7 (1936) 179–188.
[40] H. Liu, H. Motoda, R. Setiono, Z. Zhao, Feature selection: an ever evolving frontier in data mining, J. Mach. Learn. Res. Proc. Track 10 (2010) 4–13.
[41] M.A. Hall, Correlation-based feature selection for discrete and numeric class machine learning, in: ICML '00: Proceedings of the Seventeenth International Conference on Machine Learning, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2000, pp. 359–366.
[42] M. Souto, I. Costa, D. de Araujo, T. Ludermir, A. Schliep, Clustering cancer gene expression data: a comparative study, BMC Bioinformat. 9 (2008) 497.
[43] A.L. Tarca, R. Romero, S. Draghici, Analysis of microarray experiments of gene expression profiling, Am. J. Obstet. Gynecol. 195 (2006) 373–388.
[44] I.H. Witten, E. Frank, Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations, Morgan Kaufmann, 1999.
[45] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[46] R. Cerri, A. de Carvalho, A. Freitas, Adapting non-hierarchical multilabel classification methods for hierarchical multilabel classification, Intell. Data Anal. 15 (2011) 861–887.
[47] R. Cerri, R.C. Barros, A. de Carvalho, Hierarchical multi-label classification for protein function prediction: a local approach based on neural networks, in: 11th International Conference on Intelligent Systems Design and Applications (ISDA 2011), 2011, pp. 337 –343.
[48] R. Cerri, R.C. Barros, A.C.P.L.F. de Carvalho, A genetic algorithm for hierarchical multi-label classification, in: Proceedings of the 27th Annual ACM Symposium on Applied Computing, SAC '12, ACM, New York, NY, USA, 2012, pp. 250–255.

Rodrigo Coelho Barros received the B.Sc. degree from Universidade Federal de Pelotas, Pelotas, Brazil, the M. Sc. degree from Pontifícia Universidade Católica do Rio Grande do Sul, Porto Alegre, Brazil, and the Ph.D. degree from Universidade de São Paulo, São Carlos, Brazil, all in computer science, in 2007, 2009, and 2013, respectively. He is currently an assistant professor in the Faculty of Informatics, Pontifícia Universidade Católica do Rio Grande do Sul, where he works with machine learning and data mining topics. He has published several papers in peer-reviewed journals and conferences. His current research interests are machine learning, data mining, knowledge discovery, and biologically inspired computational intelligence algorithms.

Pablo Andretta Jaskowiak received his B.Sc. degree in Informatics from Western Paraná State University (UNIOESTE) and his M.Sc. from University of São Paulo (USP) in the years of 2008 and 2011, respectively. Since 2011, he has been working toward the Ph.D. degree in Computer Science in the Institute of Mathematics and Computer Science at University of São Paulo (USP). His research interests are in the areas of machine learning, data mining, and bioinformatics.

Ricardo Cerri received his B.Sc. degree in Computer Science from São Paulo State University, Brazil, in 2007, and his M.Sc. degree in Computer Science from University of São Paulo, Brazil, in 2010. Currently he is a Ph. D. candidate in the Department of Computer Science at University of São Paulo, São Carlos, Brazil. His main research interests include Machine Learning and Bioinformatics, specially in hierarchical and multi-label classification.

André Carlos Ponce de Leon Ferreira de Carvalho is a Full Professor in the Department of Computer Science, University of São Paulo, Brazil, where he was head of Department from 2008 to 2010. He received his B.Sc. and M.Sc. degrees in Computer Science from the Universidade Federal de Pernambuco, Brazil. He received his Ph.D. degree in Electronic Engineering from the University of Kent, UK. He has published around 90 Journals and 200 Conference refereed papers. He has participated in the organization of several conferences and journal special issues. His main interests are Machine Learning, Data Mining and Hybrid Intelligent Systems. He is in the editorial board of several journals and was a member of the Brazilian Computing Society, SBC, Council. He was until July 2011 the editor of the SBC/Elsevier textbook series. He has a CNPq productivity in research grant level 1C.