

# Clus-DTI: improving decision-tree classification with a clustering-based decision-tree induction algorithm

Rodrigo C. Barros · Márcio P. Basgalupp ·  
André C.P.L.F. de Carvalho · Marcos G. Quiles

Received: 10 October 2011 / Accepted: 10 April 2012 / Published online: 27 April 2012  
© The Brazilian Computer Society 2012

**Abstract** Decision-tree induction is a well-known technique for assigning objects to categories in a white-box fashion. Most decision-tree induction algorithms rely on a sub-optimal greedy top-down recursive strategy for growing the tree. Even though such a strategy has been quite successful in many problems, it presents several deficiencies. For instance, there are cases in which the hyper-rectangular surfaces generated by these algorithms can only fit the input space after several sequential partitions, which results in a large and incomprehensible tree. In this paper, we propose a new decision-tree induction algorithm based on clustering named Clus-DTI. Our intention is to investigate how clustering data as a part of the induction process affects the accuracy and complexity of the generated models. Our performance analysis is not based solely on the straightforward comparison of our proposed algorithm to baseline classifiers. We also perform a data-dependency analysis in order to identify scenarios in which Clus-DTI is a more suitable option for inducing decision trees.

**Keywords** Decision trees · Clustering · Hybrid intelligent systems · Data-dependency analysis · Machine learning

## 1 Introduction

Classification is the problem of identifying the category to which observations belong, given that the identity of the category is unknown a priori. A classification algorithm is required to place these observations (objects) into groups based on quantitative or qualitative information collected from characteristics (attributes) of each object. A great deal of algorithms have been proposed by researchers in machine learning and statistics in the last decades for classifying data in a variety of domains.

Decision-tree induction is a classification method that represents the induced knowledge through a hierarchical tree. Such a representation is intuitive and easy to assimilate by humans, which has motivated a large number of studies that make use of this strategy. Some well-known algorithms for decision-tree induction are C4.5 [1] and CART [2].

Whilst the most successful decision-tree induction algorithms make use of a top-down recursive strategy for growing the trees [3], more recent studies have looked for more efficient strategies, e.g., creating an ensemble of inconsistent trees. Ensembles are created by inducing different decision trees from training samples and the ultimate classification is generally performed through a voting scheme. Part of the research community avoid the ensemble solution with the argument that the comprehensibility component of analyzing a single decision tree is lost. Indeed, the inconsistent trees are necessary to achieve diversity in the ensemble, which in turn is necessary to increase its predictive accuracy [4]. Since comprehensibility may be crucial in several applications, ensembles are usually not a good option for these cases.

---

A previous version of this paper appeared at ENIA 2011, the Brazilian Meeting on Artificial Intelligence.

---

R.C. Barros (✉) · A.C.P.L.F. de Carvalho  
University of São Paulo—ICMC-USP, São Paulo, Brazil  
e-mail: [rbarros@icmc.usp.br](mailto:rbarros@icmc.usp.br)

A.C.P.L.F. de Carvalho  
e-mail: [andre@icmc.usp.br](mailto:andre@icmc.usp.br)

M.P. Basgalupp · M.G. Quiles  
Universidade Federal de São Paulo, São Paulo, Brazil

M.P. Basgalupp  
e-mail: [basgalupp@unifesp.br](mailto:basgalupp@unifesp.br)

M.G. Quiles  
e-mail: [quiles@unifesp.br](mailto:quiles@unifesp.br)

Another strategy that has been increasingly employed is the induction of decision trees through evolutionary algorithms (EAs) [5, 6]. The main benefit of evolving decision trees is due to the EAs' capability of escaping local optima. EAs perform a robust global search in the space of candidate solutions, and thus they are less susceptible to local optima convergence. In addition, as a result of this global search, EAs tend to cope better with attribute interactions than greedy methods [7]. The main disadvantages of evolutionary induction of decision trees are related to time and space constraints. EAs are a solid but computationally expensive heuristic. Another disadvantage lies on the large number of parameters that need to be set (and tuned) in order to optimally run the EA for decision-tree induction [8].

There are advantages and disadvantages in using one strategy or another for decision-tree induction, and the same can be said for every machine learning algorithm developed to date. Algorithms that excel in a certain dataset may perform poorly in others (i.e., the no free-lunch theorem [9]). Whereas there are many studies that propose new classification algorithms every year, most of them neglect the fact that the performance of the induced classifiers is heavily data dependent. Moreover, these studies usually fail to point out the cases in which the proposed classifier may present significant gain over well-established classifiers.

In this paper, we attempt to improve the accuracy of typical decision-tree induction algorithms, and at the same time we try to preserve the comprehensibility of the model generated. For that, we propose a new classification algorithm based on clustering, named Clus-DTI (Clustering for improving Decision-Tree Induction). The key assumption that is exploited in this paper is that a difficult problem can be decomposed into simpler sub-problems. Our main hypothesis is that we can make use of clustering for improving classification with the premise that, instead of directly solving a difficult problem, solving its sub-problems may lead to improved performance.

We investigate the performance of Clus-DTI regarding the data it is being applied to. Our contributions are twofold: (i) we propose a new classification algorithm whose objective is to improve decision tree classification through the use of clustering, and we empirically analyze its performance in public datasets; (ii) we investigate scenarios in which the proposed algorithm is a better option than a traditional decision-tree induction algorithm. This investigation is based on data-dependency analyses. For such, it takes into account how the underlying data structure affects the performance of Clus-DTI. Note that this work is an extended version of a previous conference paper [10], in which we significantly increase the number of experiments, and also largely improve on the original Clus-DTI methodological aspects.

In Sect. 2 we detail the proposed algorithm, which combines clustering and decision trees. In Sect. 3 we conduct

a straightforward comparison among Clus-DTI and a well-known decision-tree induction algorithm, C4.5 [1] (more specifically, its Java version, J48). Section 4 presents a data-dependency analysis, where we suggest some scenarios in which Clus-DTI can be a better option than J48. Section 5 presents related work, and we discuss our main conclusions in Sect. 6.

## 2 Clus-DTI

We have developed a new algorithm, named Clus-DTI (Clustering for improving decision-tree induction), whose goal is to cluster the classification data aiming at improving decision-tree induction. It works based on the following premise: some classification problems are quite difficult to solve; thus, instead of growing a single tree to solve a potentially difficult classification problem, clustering the dataset into smaller sub-problems and growing several trees may ease the complexity of solving the original problem. Formally speaking, our algorithm approximates the target function in specific regions of the input space, instead of making use of the full input space. These specific regions are not randomly chosen, but defined through a well-known clustering strategy. It is reasonable to assume that objects drawn from the same class distribution can be grouped together in order to make the classification problem simpler. We do not assume, however, that we can achieve a perfect cluster-to-class mapping (and we do not mean to).

Our hypothesis is that solving sub-problems independently from each other, instead of directly trying to solve the larger problem, may provide better classification results overall. This assumption is not new and has motivated several strategies in computer science (e.g., the divide-and-conquer strategy).

Clus-DTI works as follows. Given a classification training set  $\mathbf{X}$ , Clus-DTI clusters  $\mathbf{X}$  in  $k$  non-overlapping subsets of  $\mathbf{X}$ , i.e.,  $P = \{C_1, \dots, C_k\}$ , such that  $C_j \cap C_i = \emptyset$  for  $j \neq i$  and  $C_1 \cup \dots \cup C_k = \mathbf{X}$ . In this step, Clus-DTI uses either  $k$ -means [11] or Expectation Maximization (EM) [12], two very well-known clustering algorithms. The choice of clustering algorithm is specified by the user, according to its own knowledge of the data. Note that Clus-DTI ignores the target attribute when clustering the training set. In addition, note that Clus-DTI scales all numeric values to the interval  $[0, 1]$ , and that nominal values with  $w$  categories are transformed in  $w$  binary attributes.

Once all objects are assigned to a cluster, Clus-DTI builds a distinct decision tree to each cluster according to its respective data. It uses the well-known C4.5 decision-tree induction algorithm [1]. Therefore, partition  $P$  will have a set of  $k$  distinct decision trees,  $D = \{Dt_1, \dots, Dt_k\}$ , trained according to data belonging to each one of the clusters.

In order to decide the “optimal” value of  $k$ , Clus-DTI employs different strategies according to the clustering algorithm employed. If the user has selected  $k$ -means as the clustering algorithm, Clus-DTI employs the *Ordered Multiple Runs of  $k$ -means* (OMRk) algorithm [13] for deciding the “optimal” value of  $k$ . OMRk executes  $k$ -means repeatedly for an increasing number of clusters. For each value of  $k$ , a number of partitions obtained by  $k$ -Means (from different initializations) are assessed by using a relative validity index [14], in order to determine the best estimated value for  $k$ . Algorithm 1 depicts this rationale, where  $k^{\max}$  is the largest value of  $k$  tested,  $k^*$  is the number of clusters estimated by the method,  $V$  is the value of the validity criterion after a single run of  $k$ -means,  $V^*$  is the value of the validity criterion for the best partition found and  $p$  is the number of different partitions generated for each number  $k \leq k^{\max}$  of clusters. We assume that  $V$  should be maximized.

**Algorithm 1:** OMRk algorithm

```

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

begin
     $V^* \leftarrow -\infty$ 
    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
                 $V^* \leftarrow V$ 
                 $k^* \leftarrow k$ 
                Hold the resulting partition as  $P^*$ 
        end for
    return  $k^*, P^*$ 
end
    
```

We define the required parameters of OMRk as follows. The value of  $k^{\max}$  is set to  $\sqrt{|\mathbf{X}|}$ , where  $|\mathbf{X}|$  is the number of objects to be clustered. This value is a commonly used rule of thumb [15]. The number of executions,  $p$ , is set to 10, to generate different partitions of  $k$ -means by varying the seed of its random initialization. Recalling that the estimated time complexity of  $k$ -means is of the order of  $O(k \cdot N \cdot n)$ , where  $N$  is the number of objects,  $n$  the number of attributes and  $k$  the number of clusters, then the estimated time complexity of OMRk can be calculated as

$$\begin{aligned}
 &O(p \cdot N \cdot (2 + 3 + \dots + k^{\max}) \cdot n) \\
 &\rightarrow O(p \cdot N \cdot (k^{\max})^2 \cdot n)
 \end{aligned} \tag{1}$$

Since we have defined  $p$  as a small constant, and that usually  $N \gg n$ , then  $N$  and  $k^{\max}$  are the potentially critical variables of this problem.

Regarding the validity criterion to evaluate the partitions, we have chosen the *Simplified Silhouette Width Criterion* (SSWC) [14]. It is a simplified version of the well-known silhouette width criterion for cluster validity [16]. In order to define this criterion, let us consider that the  $i$ th object of the dataset,  $x_i$ , belongs to a given cluster  $t \in \{1, \dots, k\}$ . Let the average distance of this object to the centroid of cluster  $t$  be denoted by  $a_{t,i}$ . In addition, let the average distance of this object to the centroid of another cluster  $q, q \neq t$ , be called  $d_{q,i}$ . Finally, let  $b_{t,i}$  be the minimum  $d_{q,i}$  computed over  $q \in \{1, \dots, k\}, q \neq t$ , which represents the distance of object  $x_i$  to its closest neighboring cluster. Then, the simplified silhouette of the individual object  $x_i$  is defined as

$$S_{x_i} = \frac{b_{t,i} - a_{t,i}}{\max[a_{t,i}, b_{t,i}]} \tag{2}$$

where the denominator is a normalization term. The higher the value of  $S_{x_i}$ , the better the assignment of  $x_i$  to cluster  $t$ . The SSWC is given by the average of  $S_{x_i}, 1 \leq i \leq N$ , i.e.:

$$\text{SSWC} = \frac{1}{N} \sum_{i=1}^N S_{x_i} \tag{3}$$

and the higher the SSWC value, the better the performance of the clustering algorithm, since maximizing SSWC means minimizing the intra-cluster distance  $a_{t,i}$  while maximizing the inter-cluster distance  $b_{t_i}$ .

We have chosen SSWC since it has scored among the best validity criteria in a recent comparative study [14]. Furthermore, its time complexity is estimated as  $O(n \cdot N)$ , which is considerably better than the original silhouette width criterion— $O(n \cdot N^2)$ .

The other strategy for selecting the “optimal” value of  $k$ , i.e., if the user has chosen the EM algorithm for clustering in Clus-DTI, is based on a 10-fold cross-validation (CV) procedure, as follows:

1. The number of clusters is set to 1.
2. The training set is split randomly into 10 folds.
3. EM is performed 10 times using the 10 folds the usual CV way.
4. The resulting loglikelihood is averaged over all 10 results.
5. If the loglikelihood has increased, the number of clusters is increased by 1 and the program continues at step 2. Else, the estimated number of clusters  $k^*$  is returned, and the optimal partition  $P^*$  is generated with the full (training) dataset.

After partition  $P^*$  has been chosen either by OMRk or by the 10-fold cross-validation procedure, and after one decision tree per generated cluster has been induced, each object belonging to the test set  $\mathbf{Z}$  is assigned to one of the  $k^*$  clusters, and the corresponding decision tree  $\mathbf{Dt}_i, 1 \leq i \leq k^*$  is used to classify the test object. This procedure is repeated

for every object in test set  $Z$ . The assignment of a test instance to a cluster is performed according to the algorithm being employed. If the algorithm of choice is  $k$ -means, the test instance is assigned to the closest cluster according to the Euclidean distance between test instance and clusters' centroids. Similarly, if the EM algorithm is being employed, the joint densities are calculated so the test instance is assigned to its most probable cluster.

Algorithm 2 depicts the previously mentioned steps of Clus-DTI. Note that we do not necessarily expect a full adherence of clusters to classes. While clustering the training data may sometimes gather examples from the same class in their own cluster, we do not believe that would be the case for most real-world classification problems. Our intention on designing Clus-DTI was to provide subsets of data in which the classification problem is easier than the one of the original set. In other words, we want to identify situations in which the sum of the performance of decision trees in subsets results in better overall performance than the one of a single decision tree in the original set.

Even though this concept may be similar to the one of an ensemble of trees, we do not make use of any voting scheme for combining predictions. In that sense, Clus-DTI could be seen as a particular case of the well-known *mixture of experts* strategy [17], in which we have multiple experts (in our case, decision trees) and a “gating network” that stochastically decides which expert to use on each test instance. In the particular case of Clus-DTI, the “gating network” role is performed by the cluster analysis step. The absence of a voting scheme results in an important advantage of Clus-DTI over traditional ensembles: it can track which decision tree was used to achieve the prediction of a particular object, instead of relying on averaged results provided by inconsistent models.

---

#### Algorithm 2: Clus-DTI

---

```

input : Training dataset  $X$ , test dataset  $Z$ 
output: Set of decision trees  $D$ , test accuracy  $acc$ 
begin
  Execute OMRk or 10-fold cross-validation to discover
  optimal partition  $P^*$ 
  foreach cluster  $C_i \in P^*$  do
    Generate decision tree  $DT_i$  for data  $\in C_i$ 
    Add  $DT_i$  to the set  $D$  of decision trees
   $numCorrect \leftarrow 0$ 
  foreach object  $obj \in Z$  do
    Assign  $obj$  to closest/most probable cluster  $C^*$ 
    Classify  $obj$  with  $DT^*$  built on  $C^*$ 
    if  $obj$  was correctly classified then
       $numCorrect \leftarrow numCorrect + 1$ 
   $acc \leftarrow numCorrect \div |Z|$ 
  return  $D, acc$ 
end

```

---

### 3 Baseline comparison

We tested the performance of Clus-DTI into 27 public UCI datasets [18] and compared it to the performance of J48 (java implementation of C4.5, available at the Weka Toolkit [19]). The datasets selected are: anneal, audiology, autos, balance-scale, breast-cancer, breast-w, colic, credit-a, credit-g, diabetes, glass, heart-c, heart-h, heart-statlog, hepatitis, ionosphere, iris, kr-vs-kp, labor, lymph, mushroom, primary-tumor, segment, sick, sonar, soybean and waveform-5000 (see Table 1 for details). Those datasets with  $k$  classes,  $k > 2$ , were transformed into  $k$  datasets with two classes (one class versus all other classes), so every classification problem in this paper is binary. The restriction of dealing only with binary classification problems is due to the fact that the geometrical complexity measures [20, 21] used in this study for data-dependency analyses can deal with only two classes in most cases. With the transformations, the 27 datasets became 129 datasets, which were divided in 70 % for training and 30 % for test.

We tested both versions of Clus-DTI, i.e., by using either  $k$ -means or EM to cluster objects. Henceforth, we refer to Clus $k$  as the Clus-DTI version with  $k$ -means and to ClusEM as the Clus-DTI version with EM. Table 2 presents the results obtained in this comparison among ClusEM, Clus $k$  and J48. # Wins indicates the number of datasets in which each method has outperformed the other two regarding either test accuracy (Acc) or tree size (TS). Note that ties were omitted.

We noticed that the three methods had a similar number of wins considering test accuracy, with a small advantage to J48. Regarding tree size, both Clus-DTI versions provided smaller trees (tree size is averaged over all trees generated for a single dataset), which was expected since we are generating trees for smaller training sets (clusters). One can argue that interpreting several small trees is as difficult as interpreting a big tree. However, we highlight the fact that, for each test instance, only one tree is selected to perform the instance classification, and thus the user has to interpret only a single tree at a time.

We executed the non-parametric Wilcoxon Signed Rank Test [22] for assessing the statistical significance of both test accuracy and tree size. The results are presented in Tables 3 and 4, indicating that both Clus-DTI methods outperform J48 with statistical significance regarding tree size. Moreover, ClusEM provides significantly smaller trees than Clus $k$ . The pairwise comparisons amongst the versions of Clus-DTI and J48 did not reveal any significant difference in terms of test accuracy. All differences were measured with a 95 % confidence level.

The statistical analysis suggest that ClusEM is the best option among the methods, since it generates smaller trees than J48 with no significant loss in accuracy. However, bearing in mind that Clus-DTI is computationally more expensive than J48, it would be unwise to suggest its application

**Table 1** Data sets specification. Columns “min-class” and “max-class” refer to, respectively, the number of instances of the less-frequent class and the number of instances of the most-frequent class

Dataset	Instances	Numeric attributes	Nominal attributes	% Missing values	min-class	max-class	Classes
anneal	898	6	32	0 %	0	684	6
audiology	226	0	69	2 %	1	57	24
autos	205	15	10	1.15 %	0	67	7
balance-scale	625	4	0	0 %	49	288	3
breast-cancer	286	0	9	0.34 %	85	201	2
breast-w	699	9	0	0.25 %	241	458	2
colic	368	7	15	23 %	136	232	2
credit-a	690	6	9	0.64 %	307	383	2
credit-g	1000	7	13	0 %	300	700	2
diabetes	768	8	0	0 %	268	500	2
heart-c	303	6	7	0.17 %	0	165	5
heart-h	294	6	7	20 %	0	188	5
heart-statlog	270	13	0	0 %	120	150	2
hepatitis	155	6	13	5.67 %	32	123	2
hypothyroid	3772	7	22	5.54 %	2	3481	4
ionosphere	351	34	0	0 %	126	225	2
iris	150	4	0	0 %	50	50	3
kr-vs-kp	3196	0	36	0 %	1527	1669	2
labor	57	8	8	35.74 %	20	37	2
lymph	148	3	15	0 %	2	81	4
mushroom	8124	0	22	1.38 %	3916	4208	2
primary-tumor	339	0	17	3.90 %	0	84	22
segment	2310	19	0	0 %	330	330	7
sick	3772	6	22	2.17 %	231	3541	2
sonar	208	60	0	0 %	97	111	2
soybean	683	0	35	9.77 %	8	92	19
waveform	5000	40	0	0 %	1653	1692	3

**Table 2** Comparison among the methods J48, Clusk and ClusEM

	Wins Acc	Wins TS	Margin Acc Wins (vs. J48)	Margin TS Wins (vs. J48)	Margin Acc Loss (vs. J48)	Margin TS Loss (vs. J48)
J48	14	2	–	–	–	–
Clusk	13	63	4.40 %	23.4	4.30 %	1.75
ClusEM	10	18	4.20 %	28.2	4.40 %	0.91

Wins Acc—number of times each method has outperformed the other two regarding accuracy

Wins TS—number of times each method has outperformed the other two regarding tree size

Margin Acc Wins (vs. J48)—average difference in accuracy when a Clus-DTI method outperforms J48

Margin TS Wins (vs. J48)—average difference in tree size when a Clus-DTI method outperforms J48

Margin Acc Loss (vs. J48)—average difference in accuracy when a Clus-DTI method is outperformed by J48

Margin TS Loss (vs. J48)—average difference in tree size when a Clus-DTI method is outperformed by J48



**Table 3** Wilcoxon two-sided signed rank test for test accuracy

Accuracy	J48	Clusk	ClusEM
J48	–		
Clusk		–	
ClusEM			–

▲ The algorithm in the column outperforms the one in the row with statistical significance at a 95 % confidence level

**Table 4** Wilcoxon two-sided signed rank test for tree size

Tree Size	J48	Clusk	ClusEM
J48	–	▲	▲
Clusk		–	▲
ClusEM			–

▲ The algorithm in the column outperforms the one in the row with statistical significance at a 95 % confidence level

for any future problems. Hence, our goal is to investigate in which situations each algorithm is preferred and try to relate these scenarios to meta-data and measures of geometrical complexity of datasets. This discussion is presented in the next section.

#### 4 Data-dependency analysis

It is often said that the performance of a classifier is data dependent [23]. Notwithstanding, work that proposes new classifiers usually neglects data-dependency when analyzing their performance. The strategy usually employed is to provide a few cases in which the proposed classifier outperforms baseline classifiers according to some performance measure. Similarly, theoretical studies that analyze the behavior of classifiers also tend to neglect data-dependency. They end up evaluating the performance of a classifier in a wide range of problems, resulting in weak performance bounds.

Recent efforts have tried to link data characteristics to the performance of different classifiers in order to build recommendation systems [23]. Meta-learning is an attempt to understand data a priori of executing a learning algorithm. Data that describe the characteristics of datasets and learning algorithms are called meta-data. A learning algorithm is employed to interpret these meta-data and suggest a particular learner (or ranking a few learners) in order to better solve the problem at hand.

Meta-learners for algorithm selection usually rely on data measures limited to statistical or information-theoretic descriptions. Whereas these descriptions can be sufficient for recommending algorithms, they do not explain the geometrical characteristics of the class distributions, i.e., the manner

in which classes are separated or interleaved, a critical factor for determining classification accuracy. Hence, geometrical measures are proposed in [20, 21] for characterizing the geometrical complexity of classification problems. The study of these measures is a first effort to better understand classifiers' data-dependency. Moreover, by establishing the difficulty of a classification problem quantitatively, several studies in classification can be carried out, such as algorithm recommendation, guided data pre-processing and design of problem-aware classification algorithms.

Next, we present a summary of the meta-data and geometrical measures we use to assess classification difficulty. For the latter, the reader should notice that they actually measure the apparent geometrical complexity of datasets, since the amount of training data is limited and the true probability distribution of each class is unknown.

#### 4.1 Meta-data

The first attempt to characterize datasets for evaluating the performance of learning algorithms was made by Rendell et al. [24]. Their approach intended to predict the execution time of classification algorithms through very simple meta-attributes such as number of attributes and number of examples.

A significant improvement of such an approach was project STATLOG [25] which investigated the performance of several learning algorithms over more than twenty datasets. Approaches that followed deepened the analysis of the same set of meta-attributes for data characterization [26, 27]. This set of meta-attributes was divided in three categories: (i) simple; (ii) statistical; and (iii) information theory-based. An improved set of meta-attributes is further discussed in [28], and we make use of the following measures presented in there:

1. Number of examples ( $N$ )
2. Number of attributes ( $n$ )
3. Number of continuous attributes ( $con$ )
4. Number of nominal attributes ( $nom$ )
5. Number of binary attributes ( $bin$ )
6. Number of classes ( $cl$ )
7. Percentage of missing values ( $\%mv$ )
8. Class entropy ( $H(Y)$ )
9. Mean attribute entropy ( $MAE$ )
10. Mean attribute Gini ( $MAG$ )
11. Mean mutual information of class and attributes ( $MMI$ )
12. Uncertainty coefficient ( $UC$ )

We have chosen these measures because they are widely used, presenting interesting results in the meta-learning literature. At the same time, they are computationally efficient and simple to be implemented.

Measures 1–7 can be extracted in a straightforward way from data. Measures 8–12 are information-theory based. Class entropy,  $H(Y)$  is calculated as follows:

$$H(Y) = - \sum_j^{cl} p(Y = y_j) \log_2 p(Y = y_j) \tag{4}$$

where  $p(Y = y_j)$  is the probability the class attribute  $Y$  will take value  $y_j$ . Entropy is a measure of randomness or dispersion of a given discrete attribute. Thus, the class entropy indicates the dispersion of the class attribute. The more uniform the distribution of the class attribute, the higher the value of entropy; the less uniform, the lower the value.

Mean attribute entropy is the average entropy of all discrete (nominal) attributes. It is given by

$$MAE = \frac{\sum_{i=1}^{nom} H(X_i)}{nom} \tag{5}$$

Similarly, the mean attribute Gini is the average Gini index of all nominal attributes. The Gini index is given by

$$Gini(X) = 1 - \sum_j [p(X = x_j)]^2 \tag{6}$$

$$MAG = \frac{\sum_{i=1}^{nom} Gini(X_i)}{nom} \tag{7}$$

The mean mutual information of class and attributes measures the average of the information each attribute  $X$  conveys about class attribute  $Y$ . The mutual information  $MI(Y, X)$  (also known as information gain in the machine learning community) describes the reduction in the uncertainty of  $Y$  due to the knowledge of  $X$ , and it is defined as

$$MI(Y, X) = H(Y) - H(Y|X) \tag{8}$$

$$H(Y|X) = \sum_i p(X = x_i) H(Y|X = x_i) \tag{9}$$

for attribute  $X$  with  $i$  categories and class attribute  $Y$  with  $j$  categories. Note that  $H(Y|X = x_i)$  is the entropy of class attribute  $Y$  considering only those examples in which  $X = x_i$ . Since  $MI(Y, X)$  is defined, the mean mutual information of class and attributes is given by

$$MMI = \frac{\sum_{i=1}^{nom} MI(Y, X_i)}{nom} \tag{10}$$

The last measure to be defined is the uncertainty coefficient, which is the  $MI$  normalized by the entropy of the class attribute,  $MI(Y, X)/H(Y)$ , which is analogous to the well-known gain ratio measure, though the gain ratio normalizes the  $MI$  by the entropy of the predictive attribute and not the class attribute.

In addition to the measures presented in [28], we have also make use of the ratio of the number of examples of the less-frequent class to the most-frequent class. For binary classification problems, such a measure indicates the balancing level of the dataset. Higher values (closer to one)

indicate a balanced dataset whereas lower values indicate an imbalanced-class problem.

Next we present measures that seek to explain how the data is structured geometrically in order to assess the difficulty of a classification problem.

#### 4.2 Geometrical complexity measures

In [20] a set of measures is presented to characterize datasets with regard to their geometrical structure. These measures can highlight the manner in which classes are separated or interleaved, which is a critical factor for classification accuracy. Indeed, the geometry of classes is crucial for determining the difficulty of classifying a dataset [21].

These measures are divided in three categories: (i) measures of overlaps in the attribute space; (ii) measures of class separability; and (iii) measures of geometry, topology, and density of manifolds.

##### 4.2.1 Measures of overlaps in the attribute space

The following measures estimate different complexities related to the discriminative power of the attributes.

*The maximum Fisher discriminant ratio (F1)* This measure computes the 2-class Fisher criterion, given by

$$F1 = \frac{[\mathbf{d}^T (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)]^2}{\mathbf{d}^T \bar{\Sigma} \mathbf{d}} \tag{11}$$

where  $\mathbf{d} = \bar{\Sigma}^{-1} \Delta$  is the directional vector on which data are projected,  $\Delta = \boldsymbol{\mu}_1 - \boldsymbol{\mu}_2$ ,  $\bar{\Sigma}^{-1}$  is the pseudo-inverse of  $\bar{\Sigma}$ ,  $\boldsymbol{\mu}_i$  is the mean vector for class  $i$ ,  $\bar{\Sigma} = a \Sigma_1 + (1 - a) \Sigma_2$ ,  $0 \leq a \leq 1$ ,  $\Sigma_i$  is the scatter matrix of instances for class  $c_i$ .

A high value of the Fisher discriminant ratio indicates that there exists a vector that can separate examples belonging to different classes after these instances are projected on it.

*The overlap of the per-class bounding boxes (F2)* This measure computes the overlap of the tails of distributions defined by the instances of each class. For each attribute, it computes the ratio of the width of the overlap interval (i.e., the interval that has instances of both classes) to the width of the entire interval. Then, the measure returns the product of the ratios calculated for each attribute:

$$F2 = \prod_{i=1}^n \frac{MINMAX_i - MAXMIN_i}{MAXMAX_i - MINMIN_i} \tag{12}$$

where  $MINMAX_i = \min(\max(f_i, c_1), \max(f_i, c_2))$ ,  $MAXMIN_i = \max(\min(f_i, c_1), \min(f_i, c_2))$ ,  $MAXMAX_i = \max(\max(f_i, c_1), \max(f_i, c_2))$ ,  $MINMIN_i = \min(\min(f_i, c_1), \min(f_i, c_2))$ , and  $n$  is the total number of attributes,  $f_i$  is the  $i$ th attribute,  $c_1$  and  $c_2$  refer to the two

classes, and  $\max(f_i, c_i)$  and  $\min(f_i, c_i)$  are, respectively, the maximum and minimum values of the attribute  $f_i$  for class  $c_i$ . Nominal values are mapped to integer values to compute this measure. A low value of this metric means that the attributes can discriminate the instances of different classes.

*The maximum (individual) attribute efficiency (F3)* This measure computes the discriminative power of individual attributes and returns the value of the attribute that can discriminate the largest number of training instances. For this purpose, the following heuristic is employed. For each attribute, we consider the overlapping region (i.e., the region where there are instances of both classes) and return the ratio of the number of instances that are not in this overlapping region to the total number of instances. Then, the maximum discriminative ratio is taken as measure F3. Note that a problem is easy if there exists one attribute for which the ranges of the values spanned by each class do not overlap (in this case, this would be a linearly separable problem).

*The collective attribute efficiency (F4)* This measure follows the same idea presented by F3, but now it considers the discriminative power of all the attributes (therefore, the collective attribute efficiency). To compute the collective discriminative power, we apply the following procedure. First, we select the most discriminative attribute, that is, the attribute that can separate a major number of instances of one class. Then, all the instances that can be discriminated are removed from the dataset, and the following most discriminative attribute (regarding the remaining examples) is selected. This procedure is repeated until all the examples are discriminated or all the attributes in the attribute space are analyzed. Finally, the measure returns the proportion of instances that have been discriminated. Thus, it gives us an idea of the fraction of instances whose class could be correctly predicted by building separating hyperplanes that are parallel to one of the axis in the attribute space. Note that the measure described herein slightly differs from the maximum attribute efficiency. F3 only considers the number of examples discriminated by the most discriminative attribute, instead of all the attributes. Thence, F4 provides more information by taking into account all the attributes since we want to highlight the collective discriminative power of all the attributes.

#### 4.2.2 Measures of class separability

In this section, we describe five measures that examine the shape of the class boundary to estimate the complexity of separating instances of different classes.

*The fraction of points on the class boundary (N1)* This measure provides an estimate of the length of the class boundary. For this purpose, it builds a minimum spanning tree over the entire dataset and returns the ratio of the number nodes of the spanning tree that are connected and belong to different classes to the total number of examples in the dataset. If a node  $n_i$  is connected with nodes of different classes,  $n_i$  is counted only one time. High values of this measure indicate that the majority of the points lay closely to the class boundary, and therefore, that it may be more difficult for the learner to define this class boundary accurately.

*The ratio of average intra/inter-class nearest neighbor distance (N2)* This measure compares the within-class spread with the distances to the nearest neighbors of other classes. That is, for each input instance  $x_i$ , we calculate the distance to its nearest neighbor within the class ( $\text{intraDist}(x_i)$ ) and the distance to its nearest neighbor of any other class ( $\text{interDist}(x_i)$ ). Then, the result is the ratio of the sum of the intra-class distances to the sum of the inter-class distances for each input example:

$$N2 = \frac{\sum_{i=1}^N \text{intraDist}(x_i)}{\sum_{i=1}^N \text{interDist}(x_i)} \quad (13)$$

where  $N$  is the total number of instances in the dataset.

Low values of this measure suggest that the examples of the same class lay closely in the attribute space. High values indicate that the examples of the same class are disperse.

*The leave-one-out error rate of the one-nearest neighbor classifier (N3)* The measure denotes how close the examples of different classes are. It returns the leave-one-out error rate of the one-nearest neighbor (the kNN classifier with  $k = 1$ ) learner. Low values of this metric indicate that there is a large gap in the class boundary.

*The minimized sum of the error distance of a linear classifier (L1)* This measure evaluates to what extent the training data is linearly separable. For this purpose, it returns the sum of the difference between the prediction of a linear classifier and the actual class value. We use a support vector machine (SVM) [29] with a linear kernel, which is trained with the sequential minimal optimization (SMO) algorithm to build the linear classifier. The SMO algorithm provides an efficient training method, and the result is a linear classifier that separates the instances of two classes by means of a hyperplane. A zero value of this metric indicates that the problem is linearly separable.

*The training error of a linear classifier (L2)* This measure provides information about to what extent the training data is linearly separable. It builds the linear classifier as explained above and returns its training error.



### 4.2.3 Measures of geometry, topology, and density of manifolds

The following four metrics indirectly characterize the class separability by assuming that a class is made up of single and multiple manifolds that form the support of the distribution of the class.

*The nonlinearity of a linear classifier (L3)* This metric implements a measure of nonlinearity proposed in [30]. Given the training dataset, the method creates a test set by linear interpolation with random coefficients between pairs of randomly selected instances of the same class. Then, the measure returns the test error rate of the linear classifier (the support vector machine with linear kernel) trained with the original training set. The metric is sensitive to the smoothness of the classifier boundary and the overlap on the convex hull of the classes. This metric is implemented only for 2-class datasets.

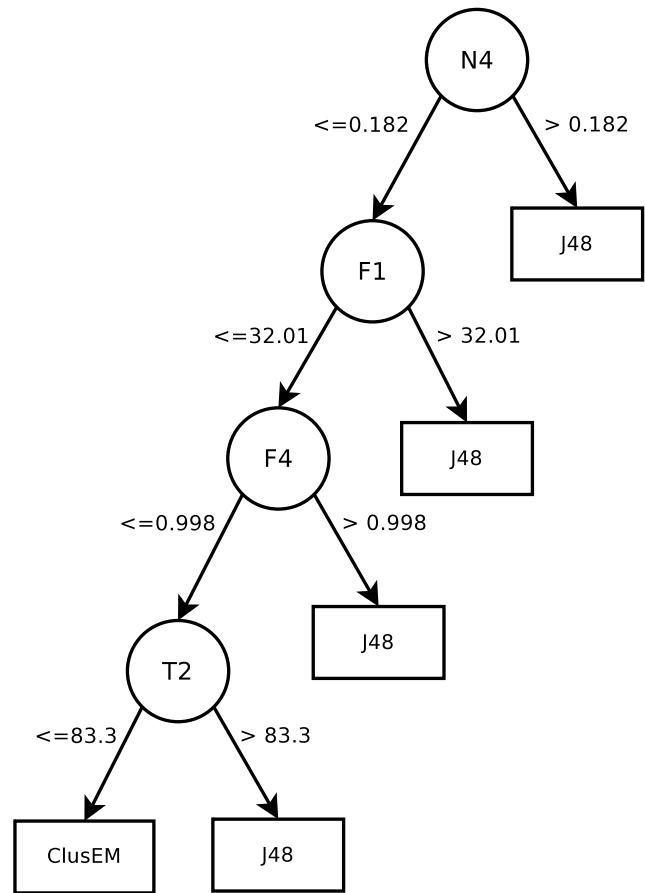
*The nonlinearity of the one-nearest neighbor classifier (N4)* This measure creates a test set as proposed by L3 and returns the test error of the 1NN classifier.

*The fraction of maximum covering spheres (T1)* This measure was originated in the work of Lebourgeois and Empyroz [31], which described the shapes of class manifolds with the notion of adherence subset. In summan, an adherence subset is a sphere centered on an example of the dataset which is grown as much as possible before touching any example of another class. Therefore, an adherence subset contains a set of examples of the same class and cannot grow more without including examples of other classes. The metric considers only the biggest adherence subsets or spheres, removing all those that are included in others. Then, the metric returns the number of spheres normalized by the total number of points.

*The average number of points per dimension (T2)* This measure returns the ratio of the number of examples in the dataset to the number of attributes. It is a rough indicator of sparseness of the dataset.

### 4.3 Results of the data-dependency analysis

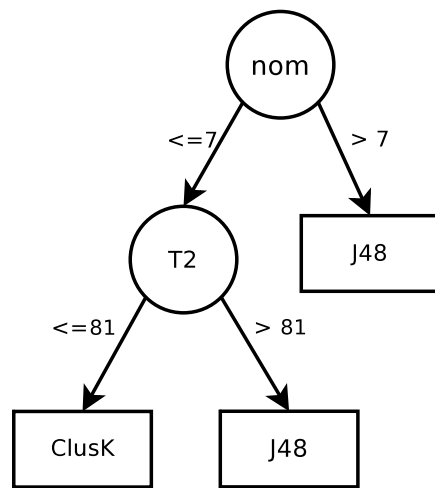
We have calculated the 13 complexity measures and the 13 meta-attributes for the 129 datasets, and we have built a training set in which each example corresponds to a dataset and each attribute is one of the 26 measures. In addition, we have included which method was better for each dataset regarding test accuracy (ClusEM, Clusk or J48) as our class attribute. Thus, we have a training set of 129 examples and 27 attributes.



**Fig. 1** Decision tree that describes the relationship between the measures and the most suitable algorithm to be used between ClusEM × J48

Our intention with this training set is to perform a descriptive analysis in order to understand which aspects of the data have a higher influence in determining the performance of the algorithms. In particular, we search for evidence that may help the user to choose a priori the most suitable method for classification. First, we have built a decision tree for each pairwise comparison (Clusk × J48 and ClusEM × J48) over the previously described training set. The idea is that the rules extracted from this decision tree may offer an insight on the reasons one algorithm outperforms the other. In other words, we are using a decision tree as a descriptive tool instead of using it as a predictive tool. This is not unusual, since the classification model provided by the decision tree can serve as an explanatory tool to distinguish between objects of different classes [32]. Figures 1 and 2 show these descriptive decision trees, which explain the behavior of roughly 90 % of the data.

First, we start by analyzing the scenarios in which one may choose to employ either ClusEM or J48. By analyzing decision tree in Fig. 1, we can notice that J48 is recommended for problems in which the N4 measure is above a given threshold. N4 provides an idea of data linearity, and



**Fig. 2** Decision tree that describes the relationship between the measures and the most suitable algorithm to be used between ClusK × J48

the higher its value, the more complex is the decision boundary between classes. The decision tree indicates that above a given threshold of  $N4$  ( $N4 > 0.182$ ), clustering provides no advantage for classifying objects. For the remaining cases, the decision tree shows that those problems deemed to be simpler by the attribute overlapping measures ( $F1$  and  $F4$ ) are better handled by J48. Specifically regarding  $F4$ , notice that the descriptive decision tree recommends employing J48 for problems in which 100 % of the instances can be correctly predicted by building separating hyperplanes that are parallel to one of the attribute axis ( $F4 > 0.998$ ). Thus, for simpler problems in which axis-parallel hyperplanes can separate the classes, J48 is an effective option. This conclusion is intuitive considering that J48 will have no particular difficulty in generating axis-parallel hyperplanes to correctly separate the training instances in easier problems. However, in more complex problems in which there are a few instances whose classes cannot be separated by axis-parallel hyperplanes, it is possible that a clustering procedure that simplifies the input space in sub-spaces can be more effective in solving the problem.

The decision tree in Fig. 1 also recommends employing J48 for problems whose sparsity (measured by  $T2$ ) is below a given threshold (the higher the value of  $T2$ , the denser the dataset). This implies that for sparser problems, in which J48 alone has problems in generating appropriate separating hyperplanes, clustering the training set may be a more effective option for classification. Though the dataset sparsity is not affected by clustering, one can assume that the generated sub-spaces are simpler to classify than the full sparse input space.

Next, we analyze the decision tree that recommends employing either ClusK or J48. The decision tree in Fig. 2 shows that for datasets whose number of nominal attributes surpass a given threshold ( $nom > 7$ ), J48 is the recom-

mended algorithm. This can be intuitively explained by the difficulty that most clustering algorithms have of dealing with nominal values. For the remaining cases, measure of sparsity  $T2$  is tested to decide between ClusK and J48. Notice that, similarly to the previous decision tree, clustering the dataset is recommended for sparser datasets, alleviating the difficulty on finding axis-parallel hyperplanes for class separation in sparse problems.

Our last recommendation is for users for whom interpretability is a strong need: if none of the scenarios highlighted before suggested that Clus-DTI is a better option than J48, it may be still worthwhile using it due to the reduced size of the trees it generates. It should be noticed that even though Clus-DTI may generate several decision trees for the same dataset, only one tree is used to classify each test instance, and hence only one tree needs to be interpreted at a time. We can think of the clustering step as a hidden (or *latent*) attribute that divides the data in subtrees, which are interpreted independently from each other. It is up to the final user to decide whether he or she is willing to waste some extra computational resources in order to analyze (potentially) more comprehensible trees.

## 5 Related work

There is much work that combines clustering and decision trees for descriptive/predictive purposes. For instance, Thomassey and Fiordaliso [33] propose employing clustering to generate sales profiles in the Textile-Apparel-Distribution industry, and then using these profiles as classes to be predicted by a decision-tree induction algorithm. Other example is the work of Samoilenko and Osei-Bryson [34], which proposes applying a methodology that combines both clustering and decision trees in data envelopment analysis (DEA) of transition economies. The cluster analysis step is responsible for detecting subsets in decision making units (DMU), whereas a later decision-tree step is employed to investigate the subset-specific nature of the DMUs used in the study.

Notwithstanding, only one work relates clustering and decision trees in the way we present in this paper, and that is the work of Gaddam et al. [35], which proposes cascading  $k$ -means and ID3 [36] for improving anomaly detection. In their approach, the training datasets are clustered in  $k$  disjoint clusters ( $k$  is empirically defined) by the  $k$ -means algorithm. A test example is first assigned to one of the generated clusters, and it is labeled according to the majority label of the cluster. An ID3 decision tree is built according to the training examples in each cluster (one tree per cluster) and it is also used for labeling the test example which was assigned to a particular cluster. The system also keeps track of the  $f$  nearest clusters from each test example, and it uses

a combining scheme that checks for an agreement between  $k$ -means and ID3 labeling.

Whereas Clus-DTI shares a fair amount of similarities to the work of Gaddam et al. [35], we highlight the main differences as follows: (i) Clus-DTI makes use of a more robust tree inducer, C4.5, and also allows the choice of a more sophisticated clustering algorithm, EM; (ii) Clus-DTI does not have a combining scheme for labeling test examples, which means our approach keeps its comprehensibility since we can directly track the rules responsible for labeling examples (which is not the case of the work in [35] and any other work that uses combining/voting schemes); (iii) Clus-DTI is not limited to anomaly detection applications; (iv) Clus-DTI automatically chooses the “ideal” number of clusters; (v) Clus-DTI is conceptually simpler and easier to implement than  $k$ -means+ID3, even though it makes use of more robust algorithms (C4.5 instead of ID3, and eventually EM over  $k$ -means).

## 6 Conclusions and future work

In this work, we have presented a new classification system which is based on the idea that clustering datasets may improve decision tree classification. Our algorithm, named Clus-DTI, makes use of well-known machine learning algorithms, namely  $k$ -means [11], Expectation Maximization [12] and C4.5 [1].

We have tested Clus-DTI using 27 public UCI datasets [18], which were transformed to binary class problems resulting in 129 distinct datasets. Experimental results indicated that Clus-DTI provided significantly smaller trees than J48, with no significant loss in accuracy. A deeper analysis was made in an attempt to relate the underlying structure of the datasets with the performance of Clus-DTI. A total of 26 different measures were calculated to assess the complexity of each dataset. Some of these measures were also used in classical meta-learning studies [26, 27]. Other measures were suggested in more recent works [20, 21] in an attempt to assess data geometrical complexity. Through these measures, we were able to generate decision trees for supporting a descriptive analysis whose goal was to point out scenarios in which Clus-DTI is a better option than J48 for data classification.

This work is a first effort in the study of the potential benefits of clustering on classification. It presents a new classification algorithm that can also be seen as a framework that is more elegant than simply pre-processing data through clustering. It has opened several venues for future work, as follows. We intend to implement alternative methods of choosing the best value of  $k$ , such as employing the GAP statistic [37], a measure that evaluates the difference (gap) between the within-cluster sum of squares of a given partition

with its expectation under a null reference distribution. The number of clusters defined by the partition with greater GAP is thus selected as “optimal”. Other alternatives for estimating the value of  $k$  include calculating the accuracy of Clus-DTI over a validation dataset and choosing the value of  $k$  that provides the best accuracy in that validation set.

We also intend to test different decision-tree induction algorithms in our framework, in order to evaluate whether our conclusions are generalizable for other algorithms. Finally, we intend to test Clus-DTI in artificial datasets so we can guarantee its effectiveness in a variety of distinct scenarios, such as linearly separable data, non-linearly separable data, imbalanced datasets, high-dimensional data, sparse data and other interesting problems.

**Acknowledgements** Our thanks to Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) and Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) for supporting this research. Also, we would like to thank Dr. Alex A. Freitas for his valuable comments, which helped improving this paper.

## References

1. Quinlan JR (1993) C4.5: programs for machine learning. Morgan Kaufmann, San Francisco
2. Breiman L, Friedman JH, Olshen RA, Stone CJ (1984) Classification and regression trees. Wadsworth, Belmont
3. Rokach L, Maimon O (2005) Top-down induction of decision trees classifiers—a survey. *IEEE Trans Syst Man Cybern, Part C, Appl Rev* 35(4):476–487
4. Freitas AA, Wieser DC, Apweiler R (2010) On the importance of comprehensible classification models for protein function prediction. *IEEE/ACM Trans Comput Biol Bioinform* 7(1):172–182
5. Basgalupp M, de Carvalho A, Barros RC, Ruiz D, Freitas A (2009) Lexicographic multi-objective evolutionary induction of decision trees. *Int J Bio-Inspir Comput* 1(1/2):105–117
6. Barros RC, Ruiz D, Basgalupp MP (2011) Evolutionary model trees for handling continuous classes in machine learning. *Inf Sci* 181:954–971
7. Freitas AA (2008) A review of evolutionary algorithms for data mining. In: *Soft computing for knowledge discovery and data mining*. Springer, New York, pp 79–111
8. Barros RC, Basgalupp M, de Carvalho A, Freitas AA (2011) A survey of evolutionary algorithms for decision-tree induction. *IEEE Trans Syst Man Cyber, Part C, Appl Rev* (in press)
9. Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. *IEEE Trans Evol Comput* 1(1):67–82
10. Barros RC, Basgalupp M, de Carvalho A (2011) Um algoritmo de indução de árvore de decisão baseado em agrupamento. In: *IX Encontro nacional de inteligência artificial*, pp 1–12
11. Lloyd S (1982) Least squares quantization in pcm. *IEEE Trans Inf Theory* 28(2):129–137
12. Dempster AP, Laird NM, Rubin DB (1977) Maximum likelihood from incomplete data via the EM algorithm. *J R Stat Soc* 39(1):1–38
13. Naldi M, Campello R, Hruschka E, Carvalho A (2011) Efficiency issues of evolutionary  $k$ -means. *Appl Soft Comput* 11:1938–1952
14. Vendramin L, Campello RJGB, Hruschka ER (2010) Relative clustering validity criteria: a comparative overview. *Stat Anal Data Min* 3:209–235

15. Campello R, Hruschka E, Alves V (2009) On the efficiency of evolutionary fuzzy clustering. *J Heuristics* 15:43–75
16. Rousseeuw P (1987) Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *J Comput Appl Math* 20:53–65
17. Jacobs RA, Jordan MI, Nowlan SJ, Hinton GE (1991) Adaptive mixtures of local experts. *Neural Comput* 3(1):79–87
18. Frank A, Asuncion A (2010) UCI machine learning repository. [Online]. Available: <http://archive.ics.uci.edu/ml>
19. Witten IH, Frank E (1999) Data mining: practical machine learning tools and techniques with java implementations. Morgan Kaufmann, San Mateo
20. Ho TK, Basu M (2002) Complexity measures of supervised classification problems. *IEEE Trans Pattern Anal Mach Intell* 24(3):289–300
21. Ho T, Basu M, Law M (2006) Measures of geometrical complexity in classification problems. In: *Data complexity in pattern recognition*. Springer, London, pp 1–23
22. Wilcoxon F (1945) Individual comparisons by ranking methods. *Biometrics* 1:80–83
23. Smith-Miles KA (2009) Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Comput Surv* 41:6:1–6:25
24. Rendell L, Sheshu R, Tchong D (1987) Layered concept-learning and dynamically variable bias management. In: *10th International joint conference on artificial intelligence*, pp 308–314
25. Michie D, Spiegelhalter DJ, Taylor CC, Campbell J (1994) *Machine learning, neural and statistical classification*. Ellis Horwood, Chichester
26. Brazdil P, Gama Ja, Henery B (1994) Characterizing the applicability of classification algorithms using meta-level learning. In: *European conference on machine learning*, Secaucus, NJ, USA, pp 83–102
27. Gama J, Brazdil P (1995) Characterization of classification algorithms. In: *7th Portuguese conference on artificial intelligence*, London, UK, pp 189–200
28. Kalousis A (2002) *Algorithm selection via meta-learning*. PhD dissertation, Université de Genève, Centre Universitaire d'Informatique
29. Vapnik VN (1995) *The nature of statistical learning theory*. Springer, New York
30. Hoekstra A, Duin RP (1996) On the nonlinearity of pattern classifiers. In: *Proceedings of the 13th ICPR*, pp 271–275
31. Lebourgeois F, Emptoz H (1996) Pretopological approach for supervised learning. In: *International conference on pattern recognition*, Washington, DC, USA, pp 256–260
32. Tan P-N, Steinbach M, Kumar V (2005) *Introduction to data mining*, 1st edn. Addison-Wesley, Longman, Boston
33. Thomassey S, Fiordaliso A (2006) A hybrid sales forecasting system based on clustering and decision trees. *Decis Support Syst* 42(1):408–421
34. Samoilenko S, Osei-Bryson K (2008) Increasing the discriminatory power of DEA in the presence of the sample heterogeneity with cluster analysis and decision trees. *Expert Syst Appl* 34(2):1568–1581
35. Gaddam SR, Phoha VV, Balagani KS (2007) K-means+id3: a novel method for supervised anomaly detection by cascading *k*-means clustering and id3 decision tree learning methods. *IEEE Trans Knowl Data Eng* 19:345–354
36. Quinlan JR (1986) *Induction of decision trees*. *Mach Learn* 1(1):81–106
37. Tibshirani R, Walther G, Hastie T (2001) Estimating the number of clusters in a data set via the GAP statistic. *J R Stat Soc, Ser B, Stat Methodol* 63(2):411–423