Machine Learning, 54, 67–92, 2004

ART: A Hybrid Classification Model

FERNANDO BERZAL* fberzal@decsai.ugr.es
JUAN-CARLOS CUBERO* jc.cubero@decsai.ugr.es
DANIEL SÁNCHEZ daniel@decsai.ugr.es
JOSÉ MARÍA SERRANO jmserrano@decsai.ugr.es
Dept. Computer Science and AI, University of Granada, Spain

Editor: Douglas Fisher

Abstract. This paper presents a new family of decision list induction algorithms based on ideas from the association rule mining context. ART, which stands for 'Association Rule Tree', builds decision lists that can be viewed as degenerate, polythetic decision trees. Our method is a generalized "Separate and Conquer" algorithm suitable for Data Mining applications because it makes use of efficient and scalable association rule mining techniques.

Keywords: supervised learning, classification, decision lists, decision trees, association rules, Data Mining

1. Introduction

Classification is a major problem in Artificial Intelligence. The aim of any classification algorithm is to build a classification model given some examples of the classes we are trying to model. The model we obtain can then be used to classify new examples or simply to achieve a better understanding of the available data.

In this paper, we present a new algorithm for building classifiers that can yield excellent results from huge data sets found in data mining problems. To ensure good scalability, we exploit association rule discovery methods to efficiently build a decision list, which can be viewed as a degenerate, polythetic decision tree, hence its name ART (Association Rule Tree). As any symbolic classification model, ART's main strength is that it provides understanding of and insight into the data.

Our method, as a decision list learner, is a generalized "Separate and Conquer" algorithm (Pagallo & Haussler, 1995), in contrast to the standard "Divide and Conquer" algorithms used to build decision trees. However, our induction process is faster than general decision list and rule inducers which need to discover rules one at a time. As with decision tree learners, ART is able to build classification models in an efficient and scalable way. Moreover, our classifiers tend to be smaller than the decision trees generated by standard TDIDT approaches.

In the following section we introduce the learning techniques our method is based on. Afterwards, we present a complete description of our classification model, whose intuitive

*Present address: Departamento de Ciencias de la Computación e Inteligencia Artificial, ETS Ingeniería Informática, Universidad de Granada, Granada 18071, Spain.

parameters can be automatically adjusted. This is followed by experimental results that show ART to be a competitive learning system in terms of accuracy and efficiency. A notable advantage of ART is efficiency when data cannot reside in main memory. Finally, we present our conclusions and pointers to future work.

2. Background

2.1. Rule learners and decision lists

There are many algorithms that induce sets of **if - then** rules directly from data. Some algorithms perform a beam search through the space of hypotheses such as INDUCE or AQ (Sestito & Dillon, 1994), as well as CN2 (Clark & Nibblett, 1989; Clark & Boswell, 1991) and CWS (Domingos, 1996). Genetic algorithms, such as REGAL (Giordana & Neri, 1996), also fall in this category.

Decision lists can be considered to be an extended **if - then - else if-**$\cdots$**-else-** rule. In fact, *k-DL* (decision lists with conjunctive clauses of size at most k) are polynomially learnable (Rivest, 1987). Although rule learning systems are computationally expensive, more efficient algorithms exist for decision lists, e.g. IREP (Fürnkranz & Widmer, 1994), RIPPERk (Cohen, 1995), and PNrule (Joshi, Agarwal, & Kumar, 2001). These algorithms are incremental, in the sense that they add one rule at a time to the classification model they build (usually through a "separate and conquer algorithm"). Some proposals, however, try to discover the whole set of rules in a single search, e.g. BruteDL (Segal & Etzioni, 1994) performs a depth-bound search of rules up to a certain length.

An alternative approach to obtain a set of **if-then** rules builds a decision tree using a "divide and conquer" algorithm. Rules can be easily extracted from decision trees, and there are methods, such as C4.5rules (Quinlan, 1993), which convert decision trees into decision lists, simplifying the set of rules that are derived from the decision tree.

2.2. Decision trees

Decision trees, also known as classification or identification trees, probably constitute the most popular and commonly-used classification model (e.g. see Quinlan, 1986b; Gehrke, Loh, & Ramakrishnan, 1999b).

The knowledge obtained during the learning process is represented using a tree where each internal node holds a question about one particular attribute (with one offspring per possible answer) and each leaf is labelled with one of the possible classes.

A decision tree may be used to classify a given example beginning at the root and following the path given by the answers to the questions at the internal nodes until a leaf is reached.

Decision trees are typically built recursively following a top-down approach (from general concepts to particular examples). That is the reason why the acronym TDIDT, which stands for Top-Down Induction on Decision Trees, is used to refer to this kind of algorithm.

The TDIDT algorithm family includes classical algorithms such as CLS (Concept Learning System), ID3 (Quinlan, 1986a), C4.5 (Quinlan, 1993), and CART (Classification And

Regression Trees) (Breiman et al., 1984), as well as more recent systems such as SLIQ (Mehta, Agrawal, & Rissanen, 1996), SPRINT (Shafer, Agrawal, & Mehta, 1996), QUEST (Loh & Shih, 1997), PUBLIC (Rastogi & Shim, 1998), RainForest (Gehrke & Ramakrishnan, 1998) and BOAT (Gehrke et al., 1999a). Other decision tree classifiers include BCT (Chan, 1989), which builds polythetic decision trees combining ideas from ID3 and CN2, and TX2steps (Elder, 1995), which performs a lookahead in decision tree construction.

2.3. Association rules

Association rule mining has been traditionally applied to databases of sales transactions (referred to as basket data). In this kind of database, a transaction T is a set of items, henceforth itemset, with a unique identifier and some additional information (e.g. date and customer).

An association rule is an implication $X \Rightarrow Y$ where X and Y are itemsets with empty intersection (i.e. with no items in common). The intuitive meaning of such a rule is that the transactions (or tuples) that contain X also tend to contain Y.

The confidence of an association rule $X \Rightarrow Y$ is the proportion of transactions containing X which also contain Y. The support of the rule is the fraction of transactions in the database which contain both X and Y.

Given a database, association rule mining algorithms try to extract all the association rules with support and confidence above user-specified thresholds, *MinSupp* and *MinConf* respectively.

Many algorithms have been proposed to solve this problem, from the original proposals, such as AIS (Agrawal, Imielinski, & Swami, 1993), SETM (Houtsma & Swami, 1993), and, in particular, Apriori (Agrawal & Srikant, 1994), to more advanced algorithms such as DHP (Park, Chen, & Yu, 1995), DIC (Brin et al., 1997), CARMA (Hidber, 1999), FP-Growth (Han, Pei, & Yin, 2000), and TBAR (Berzal et al., 2001). See Hipp, Güntzer, and Nakhaeizadeh (2000) for a recent survey of the problem and Han and Plank (1996) for a somewhat older comparison of some selected algorithms.

2.4. Related work

Some fundamental differences exist between classification and association rule discovery (Freitas, 2000). Association rules do not involve prediction, nor do they provide any mechanism to avoid underfitting and overfitting apart from the crude *MinSupport* user-specified threshold. An inductive bias is also needed to solve classification problems, i.e. a basis for favoring one hypothesis over another (e.g. Occam's razor). This bias, like any other bias, must be domain-dependent.

Association rules have, however, been used to solve classification problems directly. In Ali, Manganaris, and Srikant (1997), association rules are used to build partial classification models in domains where conventional classifiers would be ineffective. For example, traditional decision trees are problematic when many values are missing and also when the class distribution is very skewed.

In Wang, Zhou, and He (2000) a tree of rules is built from an arbitrary set of association rules without using an ad-hoc minimum support threshold. The authors have observed that predictivity often depends on high confidence, and rules of high support tend to have low confidence, so *MinSupport* pruning is not suitable for their classification purposes.

CBA is an algorithm for building complete classification models using association rules which was proposed in Liu, Hsu, and Ma (1998). In CBA [Classification Based on Associations], all "class association rules" are extracted from the available training dataset (i.e. all the association rules containing the class attribute in their consequent), and the most adequate rules are selected to build an "associative classification model", which uses a default class to make it complete. This classifier builder uses a brute-force exhaustive global search, and yields excellent results when compared to C4.5. In Liu, Ma, and Wong (2000c), CBA performance was "improved" allowing multiple minimum support thresholds for the different problem classes and reverts to traditional TDIDT classifiers when no accurate rules are found.

A similar strategy to that of CBA is used to classify text documents into topic hierarchies in Wang, Zhou, and Liew (1999). All the generalized association rules with the class attribute in their consequent are extracted, these rules are ranked, and some of them are selected to build a classifier which takes context into account, since class proximity is important when classifying documents into topics.

Hybrid approaches have also been suggested in the literature. LB (Meretakis & Wüthrich, 1999), which stands for "Large Bayes", is an extended Naïve Bayes classifier which uses an Apriori-like frequent pattern mining algorithm to discover frequent itemsets with their class support. This class support is an estimate of the probability of the pattern occurring with a certain class. The proposed algorithm achieves good results. However, it lacks the understandability of symbolic models such as decision trees.

Emerging patterns are itemsets whose support increases significantly from one dataset to another (Dong & Li, 1999). They have been used to build classifiers following the LB philosophy. For example, CAEP (Dong et al., 1999) finds all the emerging patterns meeting some support and growth rate thresholds for each class. It then computes an aggregated differentiating score to determine the most suitable class for a given instance. This computation allows the algorithm to perform well when the class populations are unbalanced, although it gives no further insight into the data.

Liu, Hu, and Hsu (2000a) propose the use of a hierarchical representation consisting of general rules and exceptions in order to replace the usual flat representation model where too many association rules hamper the understanding of the underlying data. The same approach is followed in Liu, Hu, and Hsu (2000b) in order to obtain a good summary of the knowledge contained in an arbitrary set of rules. In some way, ART takes a further step in that direction, as we will see in the following sections.

3. ART classification model

Instead of discovering rules one at a time, as most decision list learners do, ART discovers multiple rules simultaneously. ART builds partial classification models using sets of

association rules. The instances in the input dataset which are not covered by the selected association rules are then grouped together in 'else' branches to be further processed following the same algorithm.

3.1. Building the classifier

ART constructs a decision list using a greedy algorithm where each decision is not revoked once it has been taken. Since it employs an association rule mining algorithm to efficiently build partial classification models, it requires the typical user-specified thresholds used in association rule mining, *MinSupp* (minimum support for the frequent itemsets) and *MinConf* (minimum association rule confidence), although the latter can be omitted, as we shall see. In ART, the minimum support threshold is a percentage of the current dataset size and it implicitly restricts the tree branching factor. Therefore, primary or candidate keys in the input dataset will never be selected by ART.

3.1.1. ART overview. The special kind of decision list ART obtains can be considered as a degenerate, polythetic decision tree. Unlike most traditional TDIDT algorithms, ART is able to use several attributes simultaneously to branch the decision tree. This ability improves the performance of decision tree learning in terms of both *higher prediction accuracy and lower theory complexity* (Zheng, 2000). An example of the degenerate decision trees or decision lists formed by ART is shown in figure 2 (page 76). Note that the tree/list is composed of sets of rules that share the same attributes. ART learns a set of such rules iteratively, gradually piecing together the complete tree/list.

In our context, *a good rule* is an accurate rule, i.e. an association rule with a high enough confidence value so that it can be helpful in order to build an accurate classifier. As we will see in Section 3.1.3, the *best set of good rules* is the most promising set of rules according to some preference criterion. In other words, it is the set of rules which best seems to serve our purpose of building a predictive model (from a heuristic point of view, as is evident). ART will choose such a set to grow the decision list.

In its quest for candidate hypotheses, ART begins by looking for simple association rules such as $\{A_i.a_i\} \Rightarrow \{C.c_j\}$, where A is an attribute, a is its value, C is the class attribute and c is one of the problem classes. The attribute A with *the best set of good rules* $\{A_i.a_i\} \Rightarrow \{C.c_j\}$ is then selected to branch the tree, corresponding to the next set of same-attribute rules. A leaf node is generated for *each good rule* and all the data instances not covered by any of these rules are grouped in an 'else' branch to be further processed in the same way.

When no suitable association rules are found for any single attribute-value pair $A.a$, ART looks for more complex association rules. It begins with rules of the form $\{A_1.a_1\ A_2.a_2\} \Rightarrow \{C.c_j\}$. If no good candidate rules are found, ART then searches for rules with three attribute-value pairs in their antecedents, such as $\{A_1.a_1\ A_2.a_2\ A_3.a_3\} \Rightarrow \{C.c_j\}$, and so on. An upper bound on the size of the left-hand side (LHS) of the association rules is then advisable to stop the search in the rare case when no suitable rules are found. This parameter is designed as *MaxSize* and its maximum possible value equals the number of

predictive attributes in the training dataset. In fact, this parameter can be set to that value by default, unless the user explicitly changes it.

ART therefore begins by using simple hypotheses to classify the training data and makes more complex hypothesis only when no simple hypotheses are found to work well. ART thus incorporates an explicit preference for simpler models using Occam's Razor as its inductive bias. Despite recent criticisms about using Occam's razor in the KDD field (Domingos, 1998, 1999), we believe Occam's Razor is still appropiate for classification tasks. It should be noted that an inductive bias is necessary for any classification task, since given a set of observed facts, the number of hypotheses that imply these facts is potentially infinite (Freitas, 2000).

A pseudo-code description of the ART algorithm is sketched in figure 1.

```
function ART (data, MaxSize, MinSupp, MinConf): classifier;
// data:      Training dataset
// MaxSize:   Maximum LHS itemset size
//            (default value = number of predictive attributes)
// MinSupp:   Minimum support threshold
//            (default value = 0.05 = 5% )
// MinConf:   Minimum confidence threshold
//            (automatic selection by default)

k = 1;          // LHS itemset size
list = null;    // Resulting decision list (degenerate tree)

while ( (list is null) and (k ≤ MaxSize) )

    // Rule mining
    Find all the good rules from input data with
    k items in the LHS and the class attribute in the RHS
        e.g. {A1.a1 .. Ak.ak} ⇒ {C.cj}

    if there are candidate rules to grow the list

      // Rule selection
      Select the best set of rules with the same set of attributes
      {A1..Ak} in the LHS according to the preference criterion.

      // Tree branching
      list = List resulting from the selected rules
             {A1.a1 .. Ak.ak} ⇒ {C.cj}, where
             all training examples not covered by the selected
             association rules are grouped into an 'else' branch
             which is built calling the algorithm recursively:
             data = uncovered data // Transaction trimming
             list.else = ART (data, MaxSize, MinSupp, MinConf);

    else
      k = k + 1;

if list is null // no decision list has been built
  list = default rule labelled with the most frequent class;

return list;
```

Figure 1. ART algorithm outline.

3.1.2. Rule mining: Candidate rules. The first step in the ART algorithm is to discover potentially predictive rules in the training dataset. These rules will be used to grow the decision list which ART builds. An algorithm is needed to look for underlying rules in the input dataset.

The simplest possible rule mining algorithm is to discover all the association rules that include the class attribute as their consequent and satisfy the user-specified *MinSupp* and *MinConf* thresholds.

Since dense datasets are common in the relational databases typically used for classification problems (i.e., datasets with a relatively small number of missing values), an algorithm like TBAR is the best choice (Berzal et al., 2001) because it takes advantage of the First Normal Form to reduce the size of the candidate set in the association rule mining process (i.e., the number of potentially relevant itemsets).[1]

The *MinSupp* parameter can be a fixed number of tuples, or a percentage of the size of the current training dataset. In the first case, an absolute minimum support threshold is established beforehand and is fixed during all the stages of learning. Using a relative minimum support threshold, the actual support is adapted to the size of the remaining dataset. For example, if *MinSupp* is set at 0.1 and we begin with 1000 tuples, the absolute minimum support will be 100 tuples at the head of the list while it will be lower in later stages of the algorithm. If there are N tuples left in the training dataset, $0.1 * N$ will be used as the minimum support threshold in order to obtain all the frequent itemsets in the remaining dataset, which might not be frequent in the complete dataset. When setting *MinSupp* as a percentage of the size of the remaining dataset, this parameter adjusts itself to the size of the current dataset, so it does not need to be tuned by the end-user. A value between 0.05 and 0.1 seems reasonable for building decision lists. See Section 3.4 for a more formal discussion of the *MinSupp* threshold value effect on the ART decision list properties.

Let us now examine how to work with the *MinConf* threshold. This parameter is directly related to the confidence of each of the association rules considered and thus, it should be a near-to-1 value (0.9, for instance). But we have found that *MinConf* should not be finely tuned by the expert user. As we will show that it is better to let ART adjust it automatically.

In any case, if the expert user so requires, *MinConf* can be manually tuned. A typical example of this situation can be found in the MUSHROOM dataset from the UCI Machine Learning Repository, where a single classification mistake could have disastrous consequences. In this particular case, a false positive is not allowable because a poisonous mushroom would be labeled as edible. Obviously, that is not advisable, since a classification mistake could provoke health problems, even death. A stringent 100% minimum confidence threshold might be required for the MUSHROOM dataset. In cases such as these, a user-established *MinConf* threshold must be used to ensure that the classification model obtained by ART has the desirable properties (e.g., no false positives).

The MUSHROOM dataset is an extreme example, however. Setting a minimum confidence threshold beforehand can be counterproductive if this threshold is not realistic, since no classification model would be obtained if *MinConf* were too high for the actual dataset. In such a case, no association rules would be discovered and no suitable decision list could be built. Moreover, a higher *MinConf* threshold does not imply greater classifier accuracy, and,

from a practical point of view, our algorithm training time could be significantly increased (see Section 4).

Consequently, we need to introduce an automatic *MinConf* threshold selection method into ART. The idea underlying any such algorithm is that once the most accurate association rule has been found (i.e., the one with the best confidence value), only similarly accurate rules are used to build the list. An heuristic we have found to work quite well is to consider only those rules with confidence above *MaxConf*-Δ in each step, where *MaxConf* stands for the maximum confidence among the discovered association rules. This heuristics uses a parameter Δ to establish a "tolerance" interval for the confidence of the rules. We select the best possible rule and only allow slightly worse rules to be included in each set of good rules. Using this rather restrictive approach, the algorithm ensures that no bad rules will be considered. In the worst case, a good rule might not be considered at the current level of the decision list. In any case, it will be considered later when processing the remaining data, where it will hold with even more support (since the dataset is trimmed every time that new rules are added to the decision list).

Our experiments have demonstrated that the automatic confidence threshold selection obtains nearly optimal solutions (when, e.g., Δ is made equal to *MinSupp*). In our opinion, a user-established minimum confidence threshold should only be used when our problem domain requires it and it therefore becomes strictly necessary (as in the MUSHROOM example).

3.1.3. Rule selection: Preference criterion. Once we have obtained some suitable rules for classifying input data (i.e., a partial classification model has been built), some of them must be appended to the current, partial decision list.

Our algorithm checks for the existence of at least one set of suitable attributes $\{A_1\ A_2\ ..\ A_k\}$ for continuing the decision list. The selected set of attributes must lead to predictive and accurate rules of the form $\{A_1.a_1\ A_2.a_2\ ..\ A_k.a_k\} \Rightarrow \{C.c_j\}$, which will be used to classify input data as belonging to class c_j.

ART tries to make good use of the information obtained from the rule mining process. All the rules obtained at each stage of the algorithm involve k-itemsets in their left-hand side. Each k-itemset corresponds to a set of k different attributes. The rules corresponding to each set of k different attributes are grouped together and heuristics are employed to choose the best candidate among these sets in order to continue the list.

ART follows a simple heuristics which consists in choosing the set of attributes which correctly classifies more instances in the training dataset (using the previously obtained rules), provided that the corresponding rules verify the *MinConf* constraint. Such a heuristic maximizes the number of classified examples and, thus, minimizes the number of unclassified examples. ART tends to reduce the length of the decision list following Occam's Economy Principle.

3.1.4. 'Else' branches. Once a set of rules with the same set of attributes in their antecedents has been selected during the rule selection phase, we proceed now to grow the list using the discovered information.

- Each of the selected rules is added to the list.
- All the training examples which are not covered by the selected rules are grouped together into an 'else' branch to be processed in later stages of our algorithm.

In some sense, ART embeds the philosophy of the algorithms proposed by Liu Hu, and Hsu (2000a, 2000b), who try to organize and summarize the set of all the discovered rules intuitively by using general rules, summaries and exceptions. The idea is to replace a set of potentially too many rules by a more concise representation of the discovered knowledge, which is also easier to work with. The ART approach achieves this goal while creating its classification model, while Liu et al. attain it with hindsight.

The use of 'else' branches in decision lists can make classification models harder to understand by humans, since the meaning of a given rule depends on its position in the list (Segal & Etzioni, 1994). However, it should be noted that ART also differs from conventional decision list learners because it discovers multiple rules at a time, so that the rules learnt in a given iteration are mutually exclusive, whereas rules are strictly ordered in standard decision lists. This fact reduces the decision list depth and partially mitigates the understandability problem mentioned above, which is inherent to the decision list model.

Once we have described our proposed classification model, we present one application where we have obtained interesting results in order to illustrate ART main features.

3.2. An example application: The splice dataset

Our aim is to determine the type of a DNA splice junction (exon/intron, intron/exon, or neither) given a primate splice-junction gene sequence, which consists of 60 nucleotides. The following problem description is a verbatim transcript from the UCI Machine Learning Repository documentation:

> Splice junctions are points on a DNA sequence at which 'superfluous' DNA is removed during the process of protein creation in higher organisms. The problem posed in this dataset is to recognize, given a sequence of DNA, the boundaries between exons (the parts of the DNA sequence retained after splicing) and introns (the parts of the DNA sequence that are spliced out). This problem consists of two subtasks: recognizing exon/intron boundaries (referred to as EI sites), and recognizing intron/exon boundaries (IE sites). (In the biological community, IE borders are referred to as "acceptors" while EI borders are referred to as "donors".)

For this particular problem, ART builds a classifier which achieves excellent classification accuracy and whose size is still manageable. TDIDT algorithms, such as C4.5, obtain slightly better classification accuracy (above 90% using cross-validation) but they also require much larger decision trees which are harder for humans to understand. The C4.5 decision tree, for example, would require between three and four times the space occupied by our tree even after pessimistic pruning.

Moreover, ART exploits the symmetries around the junction (which occurs between P30 and P31) to attain a better understanding of the underlying patterns. Observe, for example, the P29–P31, P28–P32 and P25–P35 pairs used in the decision list. Typical TDIDT

```
P30 = A : TYPE = N (473|62)
P30 = C : TYPE = N (441|24)
P30 = T : TYPE = N (447|57)
else
  P28 = A and P32 = T : TYPE = EI (235|33)
  P28 = G and P32 = T : TYPE = EI (130|20)
  P28 = C and P32 = A : TYPE = IE (160|31)
  P28 = C and P32 = C : TYPE = IE (167|35)
  P28 = C and P32 = G : TYPE = IE (179|36)
  else
    P28 = A : TYPE = N (106|14)
    P28 = G : TYPE = N (94|4)
    else
      P29 = C and P31 = G : TYPE = EI (40|5)
      P29 = A and P31 = A : TYPE = IE (86|4)
      P29 = A and P31 = C : TYPE = IE (61|4)
      P29 = A and P31 = T : TYPE = IE (39|1)
      else
        P25 = A and P35 = G : TYPE = EI (54|5)
        P25 = G and P35 = G : TYPE = EI (63|7)
        else
          P23 = G and P35 = G : TYPE = EI (40|8)
          P23 = T and P35 = C : TYPE = IE (37|7)
          else
            P21 = G and P34 = A : TYPE = EI (41|5)
            else
              P28 = T and P29 = A : TYPE = IE (66|8)
              else
                P31 = G and P33 = A : TYPE = EI (62|9)
                else
                  P28 = T : TYPE = N (49|6)
                  else
                    P24 = C and P29 = A : TYPE = IE (39|8)
                    else
                      TYPE = IE (66|39)
```

Figure 2. ART classifier for the SPLICE dataset.

algorithms cannot use such associations and thus their applicability to problems where correlations play an important role is restricted. Although they might obtain better classification accuracy, ART provides simpler and much more understandable classification models.

3.3. Using the classifier

The classifier obtained by ART can be used to classify unlabelled instances as any other decision list classifier. Beginning at the head of the list, a given example is checked against each rule in sequence (i.e., against each rule within a mutually-exclusive set, and following an 'else' branch if necessary) until a matching rule is found. Any example will reach a matching rule eventually (including possibly a terminal default rule), and it will be labelled with the most common class that was found in the training data from which that rule was created. The ART classification process is described in figure 3.

When building the classifier, null values can be automatically sent to the 'else' branch, since they are not covered by the discovered association rules. However, when classifying

```
function classify (art, instance): class;
// Input:     art = ART classifier
//            instance = Unlabeled instance
// Output:    class = Assigned class

Match the value(s) of the instance attribute(s) with the
      value(s) in the next mutually-exclusive set of rules

if the value(s) correspond(s) to any of the rules
  return the corresponding rule class
else if there exists an else branch // follow it
  return classify(art.else,instance);
else
  return default class;
```

Figure 3. Unlabelled data classification.

data with unknown values for some attributes, another alternative could be followed, such as the one used by C4.5, for instance.

A simple rule-based procedure to evaluate an ART classifier exists. ART decision lists may end up in a default class value (produced when no association rules are found to grow the decision list further). Even worse, when the association rules used to build the list cover all the examples in the input dataset, a given instance may lead to nowhere in the decision list. In this unlikely, but possible, case, that instance would be labelled with the most common class covered by the current sublist (i.e., the most common class in the training data which led to that sublist).

3.4. ART classifier properties

In this section, we will discuss several key properties of our method for building classifiers:

Search strategy: ART performs an exhaustive global search of potentially interesting rules in the training dataset, but it makes that search local with respect to the remaining dataset not covered by rules already in the decision list. In this way, the efficiency which characterizes the heuristic greedy search used in typical TDIDT algorithms and other rule learners such as CN2 is combined with the power of the exhaustive search performed by the association rule mining algorithm. In fact, trying to classify as many remaining examples as possible helps ART to make better local decisions when building the classification model.

In some respects, ART follows Michalski's STAR methodology (Sestito & Dillon, 1994, chapter 3), as CN2 does, since it generates rules iteratively until a complete classification model has been built. It should be noted however that ART, as other decision list inducers, removes both positive and negative examples covered by the generated rules from the current dataset, while STAR algorithms must keep all negative examples when trying to build new hypotheses, which leads to more complex rules. This removal of covered instances is referred to as 'transaction trimming' in Data Mining (Park, Chen, & Yu, 1997) and it helps to reduce the number of I/O operations needed to build the classifier.

It should also be noted that ART searches for sets of rules in parallel, while other algorithms try to find one rule at a time.

Although a greedy algorithm was chosen to build the ART classifier, other approaches would also be feasible. For example, a beam search could be performed to find better classifiers. However, efficiency is a must in Data Mining problems and greedy algorithms are the best choice in this case.

Robustness (outliers & primary keys): The use of concepts taken from association rule mining helps to build more robust classifiers, minimizing the effects of noise in the input data.

The minimum support threshold makes isolated outliers harmless since they are not taken into account when deciding how to build the classifier. Moreover, the support threshold also removes the problems faced by other TDIDT algorithms when some attributes are nearly keys (e.g. ID3 would always choose these kinds of attributes because they minimize the entropy in the resulting subtrees) without the need for more artificial mechanisms (such as C4.5 gain ratio criterion).

ART therefore provides a uniform treatment of outliers (which do not contribute to association rules significantly), and primary and candidate keys (whose values do not have enough support to generate association rules on their own). Both outliers and primary keys are thorny issues for traditional TDIDT algorithms.

List complexity: The length of the decision list is restricted by the minimum support threshold.

Given an absolute minimum support threshold *MinSupp* as a normalized value between 0 and 1, the ART decision list will have at most *1/MinSupp* levels (i.e., number of sets of same-attribute rules), because at each level *n*MinSupp* instances will be trimmed at least, where *n* is the number of instances in the training dataset.

When an absolute minimum support threshold is used, no more than *MaxSize** (1/*MinSupp)* scans over the input data will be needed to build the complete classifier, since all the association rules can be obtained with, at the most, *MaxSize* scans over the training data using simple algorithms such as Apriori. In other words, ART is *O(n)* on the size of the training dataset. This fact is essential for successful data mining.

The list "branching factor" (i.e., the number of rules within same-attribute rule sets) is also determined by the minimum support threshold. 1/*MinSupp* is an upper bound on the branching factor because no more than 1/*MinSupp* rules can be selected at each level of the list (an extreme case would occur when all training instances are classified at the same level of the list using the maximum possible number of rules, all of them with the minimum allowable support).

4. Empirical results

We have implemented ART in Java 2 using Sun JDK 1.3. Our program accesses data stored in relational databases through JDBC (which stands for 'Java Database Connectivity', the standard call-level interface of the Java programming language). Our implementation of ART makes use of TBAR (Berzal et al., 2001), an Apriori-like algorithm for finding frequent

Table 1. Datasets used in our experiments.

Dataset	Examples	Attributes	Classes
AUDIOLOGY	226	70	24
CAR	1728	7	4
CHESS	3196	36	2
HAYES-ROTH	160	5	3
LENSES	24	6	3
LUNG CANCER	32	57	3
MUSHROOM	8124	23	2
NURSERY	12960	9	5
SOYBEAN	683	36	19
SPLICE	3175	61	3
TICTACTOE	958	10	2
TITANIC	2201	4	2
VOTE	435	17	2

itemsets (Agrawal & Srikant, 1994). We also used AspectJ, an aspect-oriented extension of the Java programming language (Kiczales et al., 2001), in order to monitor I/O operations in our experiments.

All the results reported in this section were obtained using 10-CV (ten-fold cross-validation). The tests were carried out on a Pentium III 1100 MHz PC running MS Windows NT 4.0 WorkStation with 128 MB of RAM. The back-end database system used in our experiments was InterBase 6 although any other database system would work (in fact, we also tested ART against Oracle and IBM DB2 servers).

Table 1 shows the datasets[2] we used in our experiments, which were downloaded from the UCI Machine Learning Repository.

In this section we compare the performance of ART classifiers against several well-known classifiers, which we translated into Java from their original C implementations. ART results where obtained using the automatic minimum confidence support threshold selection described in Section 3.1.2 (with $\Delta = MinSupp$), $MinSupp = 5\%$, and $MaxSize = 3$, which seem reasonable values for ART parameters.

C4.5 was selected as reference TDIDT algorithm (using Quinlan's gain ratio criterios and pessimistic pruning, CF = 0.25), while AQR and CN2-STAR were employed as standard rule inducers. Three decision list learners (CN2-DL, IREP, and RIPPER) were also tested, as well as two reference classifiers (Naive Bayes and a default classifier which assigns the most common class to any given instance). Alternative classifiers were also tested and yielded similar results, so they are omitted here.

4.1. Accuracy

As Table 2 shows, ART classifier accuracy is comparable to or even better than the accuracy achieved by other well-known classifiers in the larger datasets (e.g. NURSERY and CAR)

Table 2. ART accuracy (using automatic confidence threshold selection and 5% minimum support) vs. other common classifiers.

Dataset	ART	C4.5	AQR	CN2 (STAR)	CN2 (DL)	IREP	RIPPER $k = 2$	Naive Bayes	Default classifier
Size > 1000									
NURSERY	99.1 ± 0.2	96.2 ± 0.2	91.7 ± 0.8	98.1 ± 0.2	99.0 ± 0.2	89.4 ± 2.2	96.7 ± 1.2	88.0 ± 0.3	33.3 ± 0.0
MUSHROOM	98.5 ± 0.3	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	94.3 ± 0.9	51.8 ± 1.5
SPLICE	89.3 ± 1.5	94.1 ± 1.4	79.2 ± 1.3	92.3 ± 0.5	90.2 ± 1.6	91.0 ± 2.1	93.1 ± 1.7	51.9 ± 1.7	51.9 ± 1.7
CHESS	97.7 ± 1.1	99.2 ± 0.5	97.1 ± 0.8	99.4 ± 0.6	99.3 ± 0.5	99.2 ± 0.3	99.2 ± 0.3	62.5 ± 1.3	52.2 ± 0.2
TITANIC	78.6 ± 0.8	79.1 ± 0.7	67.7 ± 1.0	75.8 ± 0.7	79.1 ± 0.7	78.3 ± 0.6	78.3 ± 0.6	68.0 ± 0.7	67.7 ± 0.5
CAR	98.6 ± 1.4	92.9 ± 1.2	85.1 ± 2.0	93.9 ± 0.9	95.8 ± 1.7	74.9 ± 2.5	78.4 ± 2.6	70.0 ± 1.8	70.0 ± 1.8
Average	93.63%	93.57%	86.79%	93.25%	93.88%	89.47%	90.96%	72.46%	54.50%
Size < 1000									
TICTACTOE	81.6 ± 2.7	83.8 ± 1.9	82.3 ± 4.2	98.0 ± 0.8	92.4 ± 1.8	97.6 ± 0.8	97.5 ± 1.0	65.3 ± 0.4	65.3 ± 0.4
SOYBEAN	91.5 ± 2.7	93.7 ± 2.4	83.2 ± 3.7	92.1 ± 2.4	93.4 ± 1.2	88.1 ± 3.3	91.1 ± 4.1	58.7 ± 3.2	13.2 ± 0.1
VOTES	95.9 ± 2.9	95.9 ± 2.5	93.3 ± 4.2	94.7 ± 2.1	92.2 ± 2.7	94.0 ± 2.1	94.9 ± 2.3	89.2 ± 3.6	61.4 ± 7.4
AUDIOLOGY	65.2 ± 8.5	81.4 ± 6.3	66.4 ± 9.2	75.6 ± 6.9	80.1 ± 4.6	65.4 ± 8.2	73.9 ± 9.1	23.4 ± 8.1	25.1 ± 8.7
HAYES-ROTH	84.4 ± 9.8	73.8 ± 11.5	65.0 ± 13.5	76.2 ± 11.1	76.2 ± 9.2	68.8 ± 16.5	78.1 ± 9.4	61.9 ± 11.7	29.4 ± 9.7
LUNG CANCER	40.8 ± 20.6	43.3 ± 36.7	25.8 ± 24.8	39.2 ± 20.4	35.8 ± 27.9	35.0 ± 21.7	45.0 ± 29.9	43.3 ± 13.3	40.0 ± 11.1
LENSES	70.0 ± 34.0	81.7 ± 32.0	65.0 ± 39.8	76.7 ± 39.6	73.3 ± 41.6	56.7 ± 47.3	65.0 ± 39.8	63.3 ± 40.0	63.3 ± 40.0
Average	75.63%	79.07%	68.71%	78.93%	77.63%	72.22%	77.93%	57.88%	42.53%
Average	83.94%	85.76%	77.05%	85.54%	85.13%	80.18%	83.94%	64.61%	48.05%

Table 3. Statistical tests performed to check if the differences observed between ART and other classifiers accuracy are significant.

	Student's t-test		Wilcoxon's test	
ART vs. C4.5	$p \leq 0.3507$		$p \leq 0.3475$	
ART vs. AQR	$p \leq 0.0032$	++	$p \leq 0.0100$	++
ART vs. CN2-STAR	$p \leq 0.3901$		$p \leq 0.5405$	
ART vs. CN2-DL	$p \leq 0.3942$		$p \leq 0.5860$	
ART vs. IREP	$p \leq 0.1645$		$p \leq 0.1548$	
ART vs. RIPPER	$p \leq 0.0128$	++	$p \leq 0.0100$	++
ART vs. Naive Bayes	$p \leq 0.0004$	++	$p \leq 0.0100$	++
ART vs. Default	$p \leq 0.0001$	++	$p \leq 0.0100$	++

while its performance slightly degrades in the smaller datasets, such as AUDIOLOGY and LENSES, due to the nature of the association rule process (which is devised to efficiently manage huge datasets).

Table 3 shows the statistical tests we performed to check if ART classifier accuracy significantly differs from other classifiers accuracy.[3] We can conclude that ART is competitive with a variety of standard approaches.

However, accuracy is not the only factor to be considered when comparing classifiers. We should also take into account the resulting models complexity and the efficiency of the learning process.

4.2. Complexity

Figure 4 illustrates the complexity of the resulting classifiers, using the number of generated rules to measure classifier complexity.

Pure rule inducers following Michalski's STAR methodology (i.e. AQR and CN2-STAR) build the most complex classifiers. On the other hand, decision lists are notably smaller (let us note the logarithmic scale in the figure). Decision tree complexity, using C4.5 in our experiments, lies somewhere between decision lists and rule inducers.

In terms of classifier complexity, ART is better than STAR and TDIDT algorithms (even after pruning), while ART is similar to alternative decision list inducers. This fact is specially interesting if we take into account the results reported in the following section.

4.3. Efficiency

Tables 4 and 5 show the training time required by different algorithms for the larger datasets used in our experiments (those with more than 1000 tuples). The algorithms are ordered attending to their average training time across those datasets. Table 5 is included to reflect the existing differences among the alternative learning algorithms when they are forced to use secondary storage, as any of them would do to process datasets frequently found in Data Mining applications (i.e. datasets which do not fit into main memory).

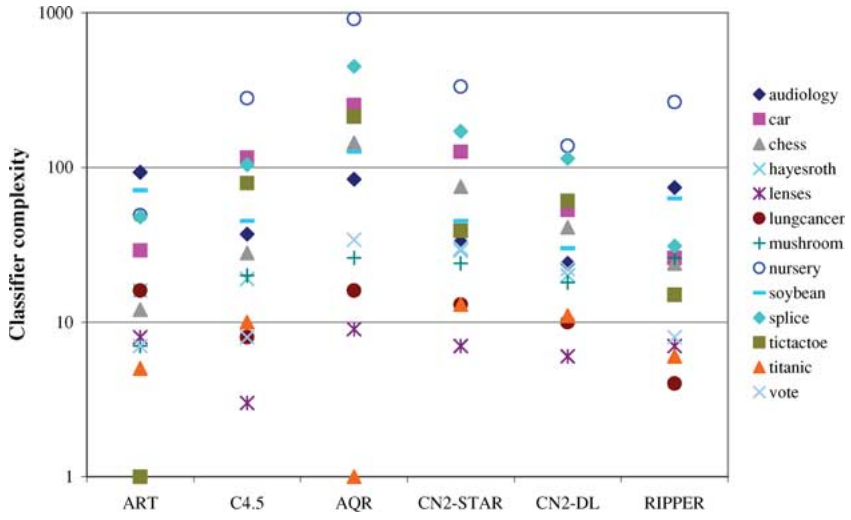

Figure 4. Classifier complexity.

ART requires more time than the traditional TDIDT induction process, as exemplified by C4.5, since ART searches in a larger solution space, which is partially controlled by ART *MaxSize* parameter. When $MaxSize = 1$, ART is able to build classifiers as efficiently as any TDIDT classifier, although this hinders ART ability to use multi-variate splits. On the other hand, when $MaxSize > 3$, ART performance might deteriorate since the number of frequent patterns in a given dense dataset might exponentially increase with its size.

With respect to decision list and rule inducers, ART is even one order of magnitude faster than AQR, CN2, IREP, and RIPPER in some cases because of its search strategy. Where previous rule inducers tried to find one rule at a time, ART looks for sets of rules reducing the number of database scans it must perform to evaluate candidate solutions. These differences are dramatically exacerbated when the training dataset does not fit into main memory (Table 5).

Table 4. Average training time for the larger datasets.

Algorithm	NURSERY	MUSHROOM	SPLICE	CHESS	TITANIC	CAR
C4.5	0.8s	0.8s	0.9s	0.9s	0.1s	0.2s
CN2-DL	15.9s	1.7s	23.7s	1.4s	0.2s	0.3s
IREP	61s	4.2s	19.7s	2.6s	0.2s	0.5s
AQR	63s	1.8s	64s	3.3s	0.1s	1.1s
ART	6.2s	1.6s	188s	6.8s	1.1s	4.8s
RIPPER	236s	7.0s	39.5s	4.6s	0.3s	1.5s
CN2-STAR	310s	8.0s	217s	14.1s	0.2s	4.2s

Table 5. Average training time for the larger datasets when disc access is forced. When memory resources are scarce, ART is only slower than C4.5.

Algorithm	NURSERY	MUSHROOM	SPLICE	CHESS	TITANIC	CAR
C4.5	17s	4s	9s	7s	1.2s	4s
ART	101s	13s	605s	57s	4.7s	13s
RIPPER	45s	719s	3062s	819s	6.8s	9s
IREP	5634s	415s	4389s	505s	12.0s	101s
CN2-DL	1743s	129s	4710s	155s	12.7s	51s
AQR	5906s	112s	12297s	403s	0.6s	223s
CN2-STAR	29552s	836s	29257s	4528s	21.5s	1023s

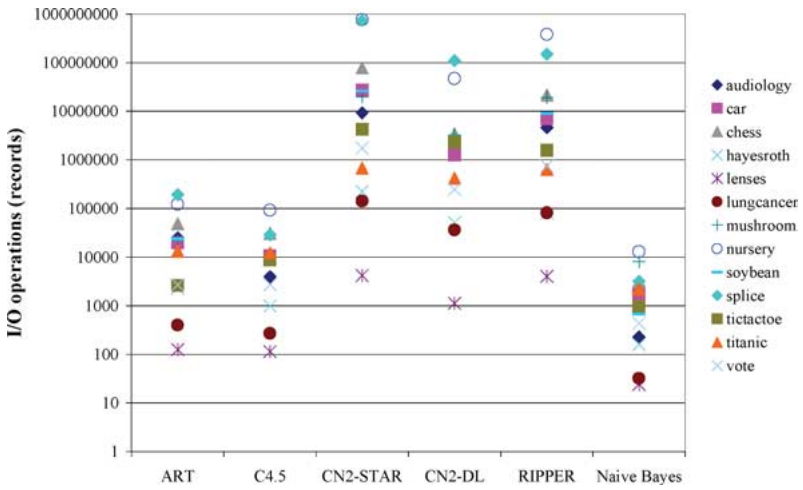

Figure 5. I/O cost for different algorithms in terms of the number of records fetched during the training process.

In Data Mining applications, CPU time is not the only relevant factor to be considered when evaluating competing alternatives. Figures 5 through 7 illustrate the I/O cost associated with each learning algorithm.

Figure 5 shows the number of times a record is accessed, while figure 6 records the number of times a dataset is sequentially scanned. It should be noted that the scanned dataset is only a fraction of the whole training dataset once the classification model has been partially built. The Naive Bayes classifier is included as a reference, since its I/O cost is optimal: it only needs to scan the training dataset once. Our efficient RainForest-like implementation of C4.5 (Gehrke & Ramakrishnan, 1998) performs two dataset scans at each internal node of the tree: one to collect the statistics which are necessary to evaluate alternative splits, another to branch the tree. On the other hand, decision list and STAR inducers perform one scan for each formulated hypothesis. This fact entails a I/O bottleneck when datasets do

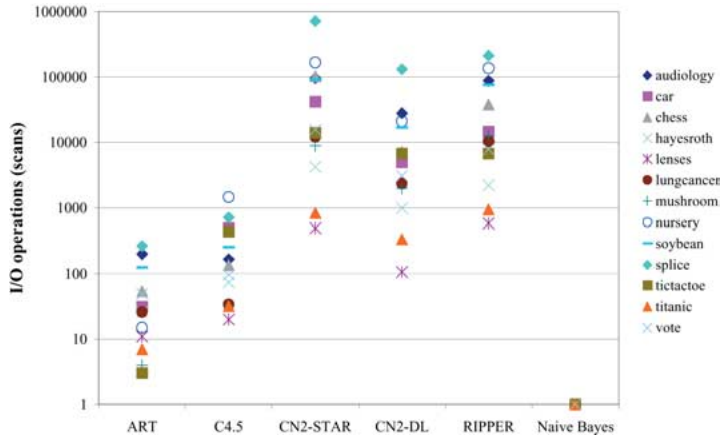

Figure 6. Number of times a dataset is scanned during classifier training.

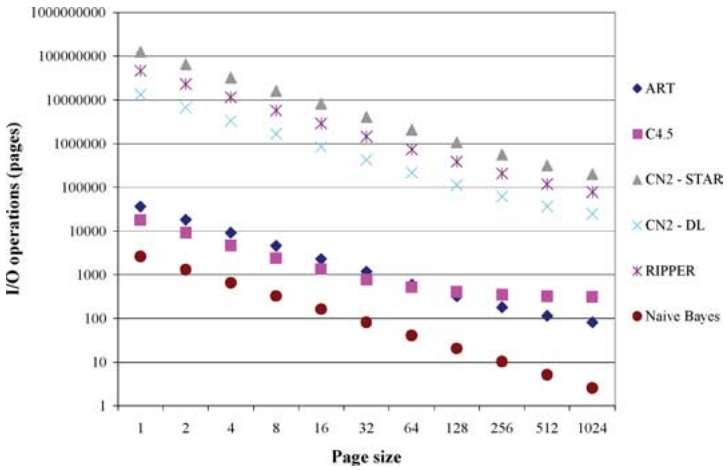

Figure 7. Number of pages read by each algorithm for different page sizes. The page size indicates the number of training examples a page contains.

not to fit into main memory. In contrast, ART I/O cost, as happened with C4.5, is bound by the resulting classifier complexity: ART performs a maximum of $MaxSize + 1$ scans each time it looks for association rules to be added to its classification model. The overall I/O operations ART performs is similar to C4.5 cost because the additional search space ART explores helps achieve more compact classifiers, as seen in Section 4.2, reducing thus the need for dataset scans.

In order to complete our experimentation, we also measured the number of disk pages read by each algorithm for different page sizes, as illustrated in figure 7. Since ART, CN2,

and RIPPER are iterative algorithms, the number of disk pages read by those algorithms proportionally decrease with the page size. C4.5, on the other hand, follows a recursive top-down strategy which fragments the training dataset into disjunct subsets, hence a non-linearity is shown in figure 7.

In summary, ART exhibits excellent scalability properties which make it suitable for Data Mining tasks. Its search strategy, which employs efficient association rule mining algorithms, makes ART orders of magnitude more efficient than alternative rule and decision list inducers, whose I/O requirements constrain their use in real-world situations unless sampling is employed. With respect to TDIDT algorithms, although ART requires more computing resources, its additional cost is reasonable if we take into account the properties of the classification models it obtains, which are discussed in previous sections.

4.4. Threshold selection

This section summarizes some experiments we have performed to check the validity of ART default parameter settings.

4.4.1. Minimum support threshold. The *MinSupp* minimum support threshold helps us to set the desired granularity level for the problem at hand, and also the time needed to build the classifier. Although ART accuracy, complexity, and cost across a range of minimum support thresholds vary depending on the particular datasets (figures 8 through 10), some general trends can be observed. A low minimum support usually yields poor accuracy and a slightly degraded training time while a high threshold might not achieve good accuracy either, since potentially interesting patterns in the dataset might never be considered.

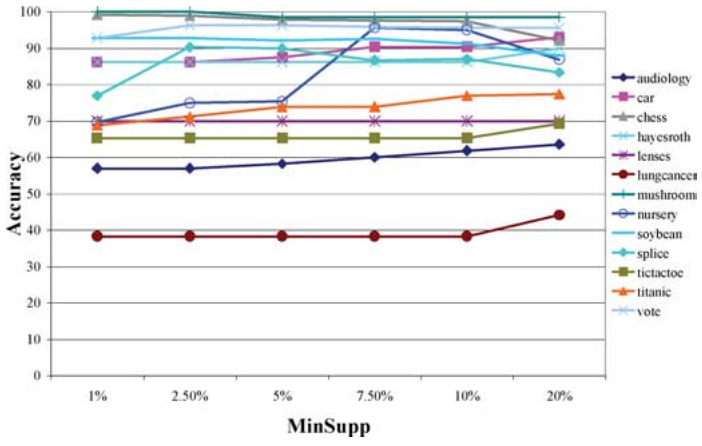

Figure 8. ART accuracy for different minimum support thresholds.

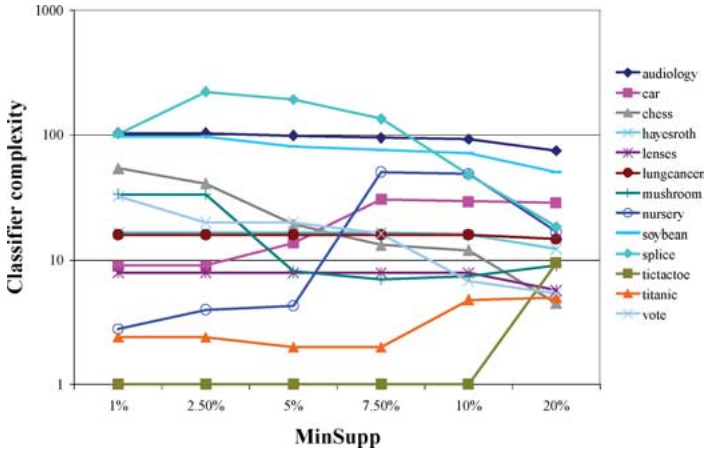

Figure 9. ART classifier complexity for different minimum support thresholds.

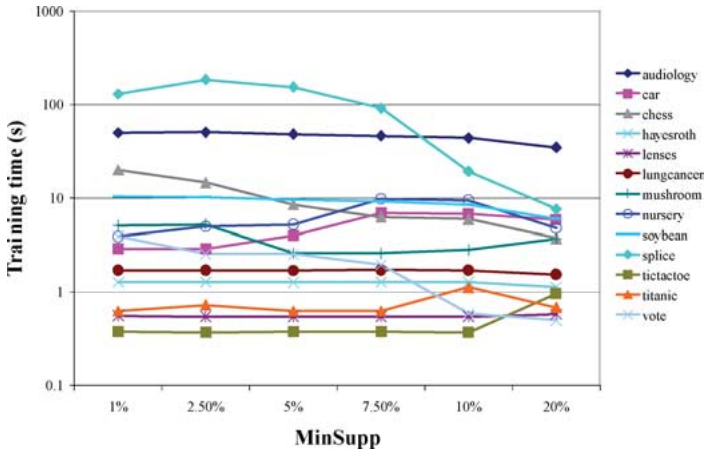

Figure 10. ART training time for different minimum support thresholds.

4.4.2. Heuristic confidence threshold selection. Regarding the minimum confidence threshold, it should be noted that a threshold established beforehand can be certainly used to establish a minimum desired accuracy in some cases (such as the MUSHROOM dataset), although increasing a minimum confidence threshold does not imply higher classifier accuracy. In fact, we have empirically observed that the automatic *MinConf* confidence threshold selection described in Section 3.1.2 achieves nearly optimal accuracy results when compared to tuning that parameter by trial and error.

Figures 11 through 13 summarize the results we have obtained when varying the Δ tolerance margin in our threshold selection heuristic. Increasing this parameter helps us to

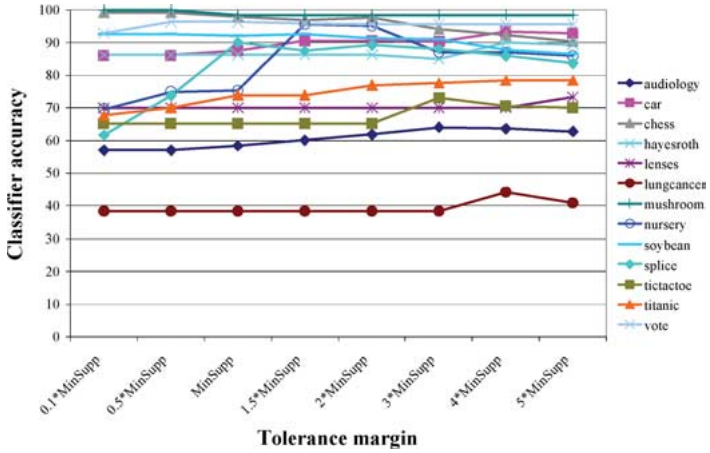

Figure 11. ART accuracy results obtained when varying the Δ parameter.

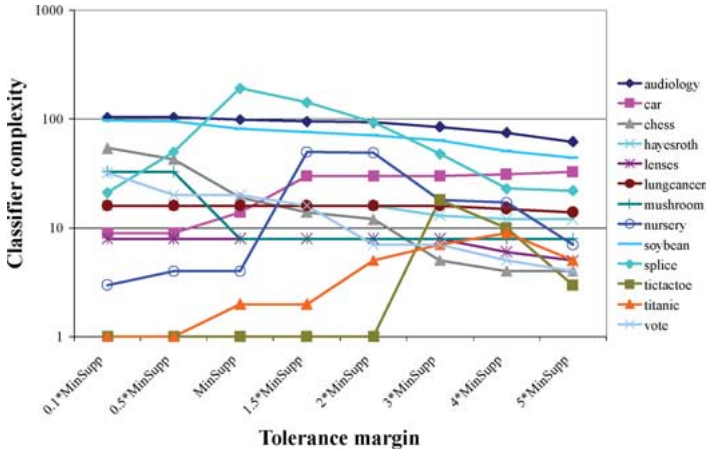

Figure 12. ART classifier complexity when varying the Δ parameter.

achieve better accuracy results sometimes, specially if we keep this parameter below the final classifier error rate. Unfortunately, this error rate is unknown beforehand, so we have to be conservative when setting the Δ parameter in order to avoid and unnecessary degradation in ART accuracy. Since no general cost and complexity trends can be confirmed from the experiments, we use an intermediate Δ value in our experiments as a trade-off between the low flexibility implied by a low Δ value and the possible loss of accuracy which might occur if we set the Δ parameter too high. In this sense, the minimum support threshold used in the association rule mining algorithm is a suitable value for the Δ parameter since it makes the tolerance margin proportional to the granularity level of the association rule mining process.

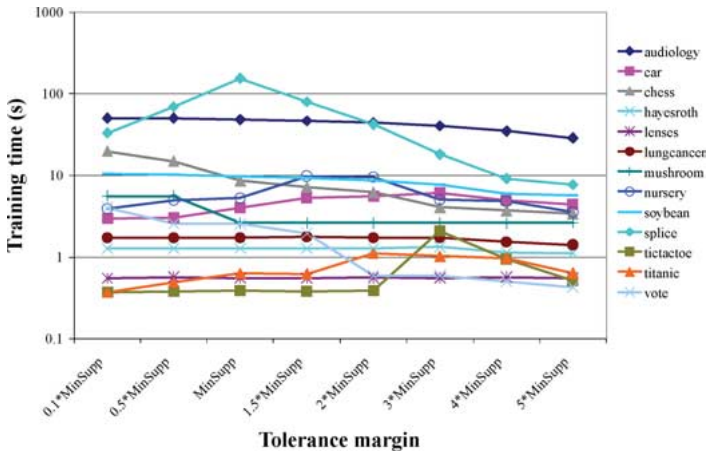

Figure 13. ART training time required for different Δ parameter values.

5. Conclusions

In this paper we have presented a new strategy to build compact, robust, and scalable classification models. Our proposal has empirically obtained promising results, as it is reported in Section 4. ART classifier accuracy, complexity, and training cost tradeoffs make it an appealing and powerful technique to be added to the data miner's toolbox.

For instance, we have achieved interesting results using ART to classify DNA splice junctions, where ART discovers and exploits the symmetries around DNA junctions to attain a better model of the underlying data (Section 3.2).

ART has demonstrated that it can build classifiers which stand out because of their simplicity and robustness. ART classifiers are easy for human users to understand. They handle both noise (i.e. outliers) and primary keys in the input data seamlessly. The association rule mining algorithm intertwined within ART provides a simple and effective mechanism to tackle a wide variety of situations without the need to use more specific, complex and artificial techniques to solve each problem.

Scalability is another interesting issue concerning ART. ART is scalable and efficient. Unlike traditional decision list inducers, which learn rules one at a time, ART simultaneously discovers multiple rules. Moreover, ART does not suffer from the I/O bottleneck common to alternative rule and decision list inducers, since it employs an efficient association rule mining algorithm to generate hypotheses. ART is therefore suitable for handling the huge datasets usually found in real-world problems.

In this article, we have also proposed a heuristic intended to automatically select the best set of discovered rules. Our heuristics obtains excellent results without the need for any user-established parameters. In any case, a minimum confidence threshold can be used to set the minimum allowable classifier accuracy if the problem at hand requires it. This capability is really interesting in some problems where no errors are allowed or minimum accuracy constraints are to be met, as we saw in Section 3.1.2.

5.1. Future work

Dealing with rules involving numerical attributes is in our immediate research agenda, since most real-life problems include such attributes. Continuous-valued attributes may appear in the rule antecedents and also in their consequents (in order to solve regression problems). Although we have not mentioned it before, quantitative association rules have been widely studied in the literature (Skirant & Agrawal, 1996; Miller & Yang, 1997; Aggarwal, Suu, & Yu, 1998; Aumann & Lindell, 1999) and can be easily employed to extend ART capabilities. Clustering techniques at the basis of quantitative association rule mining algorithms can be applied in a straightforward way, although we should resort to additional information when available in order to mine more accurate rules (i.e. the class attribute value distribution could be specially helpful). In fact, any discretization technique might prove useful for building ART classifiers and further study is required.

In this paper, two ART parameters (the *MinSupp* minimum support threshold and the *MaxSize* maximum antecedent itemset size) should be adjusted to tune our algorithm performance. Their automatic selection is still an open research problem.

It would also be worthwhile studying alternative relevance measures and preference criteria for the discovered rules. Using a different adaptive threshold for each problem class has been attempted in order to be able to discover rules for rare classes, but no significant accuracy improvements were achieved. Nor were they obtained when using more complex heuristics during the rule selection phase. This remains an open issue regarding ART classifiers.

Further study on the suitability of ART for incremental learning would be desirable too, since efficient methods to update a classifier given a set of database updates are important in Data Mining applications.

Notes

1. The First Normal Form of relational databases states that every attribute is atomic and each relation instance has values along the same attributes. This is the standard format of data commonly assumed by classification algorithms, though not typically assumed in association rule discovery systems.
2. This paper only addresses ART classifiers with categorical attributes, although ART classifiers can be built for numerical attributes using any discretization technique, as any other decision list or TDIDT learner does.
3. Wilcoxon's matched-pairs signed-ranks test was included because the typical t-test assumes that the differences are Normal distributed and we cannot assure that.

References

Aggarwal, C. C., Sun, Z., & Yu, P. S. (1998). Online algorithms for finding profile association rules. In *Proceedings of the 1998 ACM CIKM 7th International Conference on Information and Knowledge Management* (pp. 86–95). Bethesda, Maryland, USA.

Aggarwal, C. C., & Yu, P. S. (1998a). A new framework for itemset generation. In *Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems* (pp. 18–24). Seattle, Washington.

Aggarwal, C. C., & Yu, P. S. (1998b). Mining large itemsets for association rules. In *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*.

Agrawal, R., Imielinski, T., & Swami, A. (1993). Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data* (pp. 207–216). Washington, D.C.

Agrawal, R., & Srikant, R. (1994). Fast algorithms for mining association rules. In *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases* (pp. 487–499). Santiago de Chile, Chile.

Ali, K., Manganaris, S., & Srikant, R. (1997). Partial classification using association rules. In *Proceedings of the 3rd International Conference on Knowledge Discovery in Databases and Data Mining* (pp. 115–118). Newport Beach, California, USA.

Aumann, Y., & Lindell, Y. (1999). A statistical theory for quantitative association rules. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 261–270). San Diego, California, USA.

Berzal, F., Cubero, J. C., Marín, N., & Serrano, J. M. (2001). TBAR: An efficient method for association rule mining in relational databases. *Data & Knowledge Engineering, 37:1*, 47–64.

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and Regression Trees*. Wadsworth, California, USA.

Brin, S., Motwani, R., Ullman, J. D., & Tsur, S. (1997). Dynamic itemset counting and implication rules for market basket data. In *Proceedings of the ACM SIGMOD International Conference on Management of Data* (pp. 255–264). Tucson, Arizona, USA.

Chan, P. K. (1989). Inductive learning with BCT. In *Proceedings of the 6th International Workshop on Machine Learning* (pp. 104–108). Ithaca, NY.

Clark, P., & Boswell, R. (1991). Rule induction with CN2: Some recent improvements. In Y. Kodratoff (ed.), *Machine Learning—EWSL-91* (pp. 151–163). Berlin: Springer-Verlag.

Clark, P., & Nibblett, T. (1989). The CN2 induction algorithm. *Machine Learning Journal (Kluwer Academic Publishers), 3:4*, 261–183.

Cohen, W. (1995). Fast effective rule induction. In *Proc. 12th International Conference on Machine Learning* (pp. 115–123). Morgan Kaufmann.

Domingos, P. (1996). Linear-time rule induction. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)* (pp. 96–101).

Domingos, P. (1998). Occam's two Razors: The sharp and the blunt. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98)* (pp. 37–43). New York City, USA.

Domingos, P. (1999). The role of Occam's Razor in knowledge discovery. *Data Mining and Knowledge Discovery, 3:4*, 409–425.

Dong, G., & Li, J. (1999). Efficient mining of emerging patterns: Discovering trends and differences. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 43–52). San Diego, CA USA.

Dong, G., Zhang, X., Wong, L., & Li, J. (1999). CAEP: Classification by aggregating emerging patterns. In *Proceedings of the Second International Conference on Discovery Science* (pp. 30–42). Tokyo, Japan.

Elder IV, J. F. (1995). *Heuristic Search for Model Structure: The Benefits of Restraining Greed*. AI & Statistics–95, Ft. Lauderdale, Florida, pp. 199–210.

Freitas, A. A. (2000). Understanding the crucial differences between classification and discovery of association rules—A position paper. *SIGKDD Explorations, 2:1*, 65–69.

Fürnkranz, J., & Widmer, F. (1994). Incremental reduced error pruning. In *Machine Learning: Proceedings of the 11th Annual Conference*. New Brunswick, New Jersey: Morgan Kaufmann.

Gehrke, J., Ganti, V., Ramakrishnan, R., & Loh, W.-Y. (1999a). BOAT—optimistic decision tree construction. In *Proceedings of the 1999 ACM SIGMOD international conference on Management of Data* (pp. 169–180). Philadelphia, PA, USA.

Gehrke, J., Loh, W.-Y., & Ramakrishnan, R. (1999b). Classification and regression: Money can grow on trees. In *Tutorial Notes for ACM SIGKDD 1999 International Conference on Knowledge Discovery and Data Mining* (pp. 1–73). San Diego, California, USA.

Gehrke, J, Ramakrishnan, R., & Ganti, V. (2000). RainForest–A framework for fast decision tree construction of large datasets. *Data Mining and Knowledge Discovery, 4:2/3*, 127–162.

Giordana, A., & Neri, F. (1996). Search-intensive concept induction. *Evolutionary Computation, 3:4*, 375–416.

Han, J., Pei, J., & Yin, Y. (2000). Mining frequent patterns without candidate generation. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data* (pp. 1–12). Dallas, TX, USA.

Han, J. L., & Plank, A. W. (1996). Background for association rules and cost estimate of selected mining algorithms. In *CIKM '96, Proceedings of the Fifth International Conference on Information and Knowledge Management* (pp. 73–80). Rockville, Maryland, USA.

Hidber, C. (1999). Online association rule mining. In *Proceedings of the 1999 ACM SIGMOD international conference on Management of Data* (pp. 145–156). Philadelphia, PA, USA.

Hipp, J., Güntzer, U., & Nakhaeizadeh, G. (2000). Algorithms for association rule mining—A general survey and comparison. *SIGKDD Explorations, 2:1*, 58–64.

Houtsma, M., & Swami, A. (1993). Set-oriented mining for association rules. IBM Research Report RJ9567, IBM Almaden Research Center, San Jose, California.

Joshi, M. V., Agarwal, R. C., & Kumar, V. (2001). Mining needles in a haystack: Classifying rare classes via two-phase rule induction. In *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data* (pp. 91–101). Santa Barbara, California.

Kiczales, G., Hilsdale, E., Hugunin, J., Kersten, M., Palm, J., & Griswold, W. G. (2001). Getting started with Aspect. *J. Communications of the ACM, 44:10*, 59–65.

Liu, B., Hsu, W., & Ma, Y. (1998). Integrating classification and association rule mining. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98)* (pp. 80–86). New York City, USA.

Liu, B., Hu, M., & Hsu, W. (2000a) Intuitive representation of decision trees using general rules and exceptions. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000)*. Austin, Texas.

Liu, B., Hu, M., & Hsu, W. (2000b) Multi-level organization and summarization of the discovered rule. In *Proceedings of the sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 208–217). Boston, MA, USA.

Liu, B., Ma, Y., & Wong, C. K. (2000c). Improving an association rule based classifier. In *Proceedings of the Fourth European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD-2000)*. Lyon, France.

Loh, W.-Y., & Shih, Y.-S. (1997). Split selection methods for classification trees. *Statistica Sinica, 7*, 815–840.

Mehta, M., Agrawal, R., & Rissanen, J. (1996). SLIQ: A fast scalable classifier for data mining. *Advances in Database Technology—Proceedings of the Fifth International Conference on Extending Database Technology (EDBT'96)* (pp. 18–32). Avignon, France.

Meretakis, D., & Wüthrich, B. (1999). Extending naïve Bayes classifiers using long itemsets. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 165–174). San Diego, CA, USA.

Miller, R. J., & Yang, Y. (1997). Association rules over interval data. In *Proceedings of the ACM SIGMOD Conference on Management of Data* (pp. 452–461). Tucson, AZ, USA.

Pagallo, G., & Haussler, D. (1990). Boolean feature discovery in empirical learning. *Machine Learning, 5*, 71–99.

Park, J. S., Chen, M. S., & Yu, P. S. (1995). An effective hash-based algorithm for mining association rules. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data* (pp. 175–186). San Jose, California.

Park, J. S., Chen, M. S., & Yu, P. S. (1997). Using a hash-based method with transaction trimming for mining association rules. *IEEE Transactions on Knowledge and Data Engineering, 9:5*, 813–825.

Quinlan, J. R. (1986a). Induction on decision trees. *Machine Learning, 1*, 81–106.

Quinlan, J. R. (1986b). Learning Decision Tree Classifiers. *ACM Computing Surveys, 28:1*, 71–72.

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

Rivest, R. L. (1987). Learning decision lists. *Machine Learning Journal, 2:3*, 229–246.

Segal, R., & Etzioni, O. (1994). Learning decision lists using homogeneous rules. *AAAI 1994, 12th National Conference on Artificial Intelligence* (pp. 619–625). Seattle, WA, USA.

Sestito, S., & Dillon, T. S. (1994). *Automated Knowledge Acquisition*. Sydney: Prentice Hall.

Srikant, R., & Agrawal, R. (1996). Mining quantitative association rules in large relational tables. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data* (pp. 1–12). Montreal, Quebec, Canada.

Rastogi, R., & Shim, K. (1998). PUBLIC: A decision tree classifier that integrates building and pruning. In *VLDB'98, Proceedings of 24th International Conference on Very Large Data Bases* (pp. 404–415). New York City, New York, USA.

Shafer, J. C., Agrawal, R., & Mehta, M. (1996). SPRINT: A scalable parallel classifier for data mining. In *VLDB'96, Proceedings of 22nd International Conference on Very Large Data Bases* (pp. 544–555). Mumbai (Bombay), India.

Wang, K., Zhou, S., & He, Y. (2000). Growing decision trees on support-less association rules. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 265–269). Boston, MA, USA.

Wang, K., Zhou, S., & Liew, S. C. (1999). Building hierarchical classifiers using class proximity. In *VLDB'99, Proceedings of 25th International Conference on Very Large Data Bases* (pp. 363–374). Edinburgh, Scotland, UK.

Zheng, Z. (2000). Constructing X-of-N attributes for decision tree learning. *Machine Learning, 40:1*, 35–75.

Received March 5, 2001
Revised November 12, 2002
Accepted December 11, 2002
Final manuscript December 11, 2002