A Generic Framework for Rule-Based Classification

Arnaud Giacometti, Eynollah Khanjari Miyaneh
Patrick Marcel, Arnaud Soulet

L.I. Université François Rabelais de Tours
41000 Blois, France
Eynollah.khanjari@etu.univ-tours.fr
{arnaud.giacometti, patrick.marcel, arnaud.soulet}@univ-tours.fr

Abstract. Classification is an important field of data mining problems. Given a set of labeled training examples the classification task constructs a classifier. A classifier is a global model which is used to predict the class label for data objects that are unlabeled. Many approaches have been proposed for the classification problem. Among them, rule-induction, associative and instance-centric approaches have been closely integrated with constraint-based data mining. There also exist several classification methods based on each of these approaches, e.g. AQ, CBA and HARMONY respectively. Moreover, each classification method may provide one or more algorithms that exploit particular local pattern extraction techniques to construct a classifier. In this paper, we proposed a generic classification framework that encompasses all the mentioned approaches. Based on our framework we present a formal context to define basic concepts, operators for classifier construction, composition of classifiers, and class prediction. Moreover, we proposed a generic classifier construction algorithm (ICCA) that incrementally constructs a classifier using the proposed operators. This algorithm is generic in the sense that it can uniformly represent a large class of existing classification algorithms. We also present the properties under which different optimization possibilities are provided in the generic algorithm.

Key words: classification, classifier, classification rule, prediction, local patterns, global patterns.

1 Introduction

Many pattern extraction methods have been proposed in the constraint-based data mining field. On the one hand, individual patterns are declaratively specified under a local model using various class of constraints. The local pattern mining algorithms, e.g. a typical association rule mining algorithm, extracts individual patterns that are capable of describing some portions of the whole underlying database from which they have been generated. The descriptive property of the local patterns are independent from each other. On the other hand, any arbitrary combination of individual patterns can be assumed as a global pattern

[14]. In a classification task, it should also be described how the global pattern is applied to comply with the user's request, i.e. class label prediction for unclassified data objects. Therefore, classification is a predictive global modeling task among data mining problems. Given a set of (labeled) training examples, i.e. a training database, the classification task constructs a classifier. A classifier is a global model which not only describes the whole training database achieving some level of accuracy, but also is used to predict the class label for data objects that are unlabeled. Intuitively, not only the problem of finding a (minimal) set of patterns describing the whole training database and achieving the maximum accuracy is intractable, but also this problem is intractable even for the binary-class classification problem [10] w.r.t the domain size of the training data set. Moreover, although the local pattern extraction methods can be adapted and exploited for classifier construction, however, the problem remains intractable for general measure functions. More clearly, when the pattern evaluation measure is an arbitrary measure it lacks some useful properties to be used to reduce the search space of patterns.

1.1 Related works

Considering the limitations mentioned above, many approaches have been proposed aiming at the construction of an (approximately) accurate classifier. Among them, the following approaches are well-known in the literature: decision-tree, rule-induction, association-based, instance-centric and hybrid ones. More precisely, a number of classification methods have been presented in the light of each of these approaches where each method provides one or more particular algorithms to construct a classifier. We clarify this by some examples. ID3 [11] and C4.5 [12] are based on decision-tree induction approach which directly construct a classification model from the training database following a top-down and divide-and-conquer paradigm. AQ [9] and CN2 [1] are two methods based on rule-induction approach where the methods induce one rule at a time and all the data examples covered by that rule are removed from the search space. The associative approach integrates association rule mining with classification. CBA [8], CMAR [7] and CorClass [18] are examples of methods based on associative approach. In the instance-centric approach the final set of class association rules is directly extracted such that for each training example it includes at least one of the most interesting rules (correctly) covering that example. HARMONY [16] and DeEPs [5] are two instance-centric classifiers. Note that our framework is based on rule-based approach so that we take into consideration the extraction of k rules at a time which extends the traditional rule-induction approach, i.e. one rule at a time. Moreover, by rule-based approach we mean the employment of local patterns to construct a classifier, i.e. a global model, such that some order on the extracted rules are to be provided. Therefore, in our rule-based classification approach (RBC) the local patterns are not to be considered individual and independent from each other. In this sense, we does not assume the decision-tree induction as a rule-based approach since a decision-tree algorithm directly constructs a global model without employing any individual local pattern during the construction process. Excepting [14] which presents a theoretically exact formalism for the problem of constraint-based global pattern mining, there is no a general framework for the classification problem in the literature. In fact, in [14] the authors defined a general constraint-based framework and tried to adapt the properties from local pattern mining for use in pattern set mining. However, it suffers from the combinatorial explosion of pattern space (or more precisely, pattern set spaces) in practical and real-word problems.

1.2 Motivation and Contributions

The generic approach which is presented in this paper, has mainly been motivated by the fact that it is a longterm goal in the data mining (and especially in the inductive database) community to provide a generic pattern mining framework so that different pattern types can be uniformly specified taking into account both structural and operational aspects. It is also interested to provide a formal and generic framework such that different approaches and methods for the extraction of global patterns (of one or more types) can be uniformly represented. Despite many and diverse classification approaches and methods, there is not a generic framework for the classification problem. This motivated us to propose a generic framework for rule-based classification respecting the two aspects. In the structural aspect it is desirable to formally specify data objects and patterns in a uniform and language-based manner. For the operational aspect, some primitives, basic operators, and compositional operators are needed for the construction of a classification model. In this paper all the concepts of our approach are specified in a formal framework. The generality of the framework can be interpreted from different viewpoints. First, we present a generic definition for data-dependent order over rules and classifiers as well as basic operators. In fact, we provide a set of operators for rule extraction, construction and composition of classifiers, and class prediction of data objects. Second, our approach is generic in the sense that it encompasses, if not most, many of the existing classification approaches. In other words, the rule-based approach is a conceptual integration of rule-induction, associative, and instance-centric approaches. Third, from an algorithmic point of view, we proposed a generic classifier construction algorithm (ICCA) that incrementally constructs a classifier using the proposed operators. This algorithm is generic in the sense that it can uniformly represent a large class of existing classification algorithms, e.g. AQ [9], CN2 [1], CBA [8], CMAR [7], FOIL [13], PRM [17], CorClass [18] and HARMONY [16]. We formally represent all these algorithms in terms of our generic algorithm. This is a non exhaustive list of classifiers and then, many others can be described, as for example DeEPs [5], CAEP [3], JEP [6], RIPPER [2] and so on. We also present the properties under which different optimizations can be provided in the generic algorithm.

This paper is organized as follows: Section 2 provides basic definitions and introduces several operators and their properties for the classification problem in a language-based formal framework. Section 3 introduces the generic incremental classification algorithm. In Section 4 some existing classifiers are represented in terms of our rule-based approach and the algorithms are reformulated in ICCA form. In Section 5 we introduce a comparative study of different classification methods based on properties of the rule-based approach and ICCA. Finally Section 6 concludes the main characteristics of the framework and discusses future works.

2 Formal Framework

In this section, we give the formal definitions of the notions used throughout the paper to describe the rule-based classification process. This process consists of using a training set of labeled objects (called dataset of examples in what follows, see Section 2.1) from which classification rules are extracted (see Section 2.2) by the employment of basic operations (see Section 2.3) for building a classifier (see Section 2.4), and using it to predict the class label of a given unlabeled object (see Section 2.5).

2.1 Objects and examples

Objects Let $\mathcal{A}$ be a set of attribute names and $\mathcal{D}$ be a set of constants. An object o is a tuple over $\mathcal{A}$, which is called the schema of o, noted $sch(o)$, and for a given $A \in \mathcal{A}$, $o(A) \in \mathcal{D}$ denotes the value of the attribute A for o. Let $\mathcal{O}$ be the set of objects.

Data examples Let $Class$ be a set of class labels. A data example (example for short) over a given schema $\mathcal{A}$ is a tuple $\langle o, c\rangle$ where $o \in \mathcal{O}$, $\mathcal{A}$ is the schema of o and $c \in Class$. Given an example $e = \langle o, c\rangle$, we note $c = cl(e)$. In what follows, we consider examples over a fixed schema $\mathcal{A}$, and we denote by $\mathcal{E}$ this set of examples over $\mathcal{A}$.

A data set E is a subset of $\mathcal{E}$, for which $|\{c/\exists o \in \mathcal{O}, \langle o, c\rangle \in E\}|$ is denoted by $NbClass$. Given a class label $c_j \in Class$ and a data set $E \subseteq \mathcal{E}$, we denote by E_j the set of examples in E of class c_j, i.e., $E_j = \{e \in E | cl(e) = c_j\}$.

2.2 Classification Rules

We now recall the classical definitions of classification rules.

Rules Given the set of examples $\mathcal{E}$ over a schema $\mathcal{A}$, a rule is a tuple $\langle o, c\rangle$, where $o \in \mathcal{O}$, the schema of o is a schema $\{A_1, \ldots, A_n\} \subseteq \mathcal{A}$ and $c \in Class$. A rule is noted $o \rightarrow c$, and we use $|r|$ to denote $|sch(o)|$. We denote by $\mathcal{R}$ the set of rules.

Rule specialization A rule $r = o \rightarrow c$ is more general than a rule $r' = o' \rightarrow c'$ if $sch(o) \subseteq sch(o')$ and $\forall A \in sch(o), o'(A) = o(A)$. This is noted $r \prec_{\mathcal{R}} r'$. If two rules satisfy a given rule quality our approach prefers the more general one, regardless of their class labels.

Coverage A rule $r = o \rightarrow c$ covers an example $e = \langle o', c' \rangle$, noted $r \lhd e$, if $sch(o) \subseteq sch(o')$ and $\forall A \in sch(o), o'(A) = o(A)$.

Support and confidence Let $r = o \rightarrow c$ be a rule and E be a data set, the support and confidence of r are defined as usual by, respectively: $sup(r, E) = \frac{|\{e \in E | r \lhd e \wedge cl(r) = cl(e)\}|}{|E|}$ and $conf(r, E) = \frac{|\{e \in E | r \lhd e \wedge cl(r) = cl(e)\}|}{|\{e \in E | r \lhd e\}|}$.

2.3 Theory and Top$_k$

We now introduce the two basic operations used in the description of the classification process. Note that we give here a generic definition of these operations, in the sense that they are given for any language $\mathcal{L}$.

The first operation is the theory computation operation Th that extracts from a data set the elements of a given language $\mathcal{L}$ satisfying a given selection predicate.

Selection predicate Given a set $\mathcal{L}$, a selection predicate for $\mathcal{L}$ is used to express a condition on the elements of $\mathcal{L}$ w.r.t. $\mathcal{E}$. It is a boolean function on $\mathcal{L} \times 2^{\mathcal{E}}$.

Theory Given a set $\mathcal{L}$, a data set $E \subseteq \mathcal{E}$ and a selection predicate q for $\mathcal{L}$, a theory of elements of $\mathcal{L}$ is the set $Th(\mathcal{L}, E, q) = \{\varphi \in \mathcal{L} | q(\varphi, E) = true\}$.

The second operation, Top$_k$, extracts the k best elements of a language $\mathcal{L}$ w.r.t. a given order on $\mathcal{L}$. This order may depend on a given data set. To this end we first define the notion of data dependent order.

Data dependent order ($\mathcal{E}$-order) A data dependent order (or $\mathcal{E}$-order) on a given set $\mathcal{L}$ is a relation $\alpha \subseteq \mathcal{L} \times \mathcal{L} \times 2^{\mathcal{E}}$ such that, for a given $E \in 2^{\mathcal{E}}$, the relation $\alpha[E] = \{\langle \varphi, \varphi' \rangle | \langle \varphi, \varphi', E \rangle \in \alpha\}$ is an order on $\mathcal{L}$. In what follows, we note $\alpha(\varphi, \varphi', E) = true$ if $\langle \varphi, \varphi', E \rangle \in \alpha$.

Top$_k$ Given a set $\mathcal{L}$, a dataset E and α an $\mathcal{E}$-order, the top k elements of $\mathcal{L}$ w.r.t. the $\mathcal{E}$-order α is the set $Top_k(\mathcal{L}, E, \alpha) = \{\varphi \in \mathcal{L} / |\{\varphi' \in \mathcal{L} / \alpha(\varphi', \varphi, E) = true\}| < k\}$

Example 1. Given a dataset $E \subseteq \mathcal{E}$, the language $\mathcal{R}$ of classification rules, and the predicate q such that for a given t, $\forall r \in \mathcal{R}, q(r, E) = true$ if $conf(r, E) > t$, $Th(\mathcal{R}, E, q)$ extracts those classification rules from E having confidence greater than t.

An example of $\mathcal{E}$-order on $\mathcal{R}$ is α_{csc}, that we will use in the subsequent sections, defined for every $r, r' \in \mathcal{R}^2$ and $E \subseteq \mathcal{E}$ by $\alpha_{csc}(r, r', E) = true$ iff :

– $conf(r, E) > conf(r', E)$, or
– $conf(r, E) = conf(r', E)$ and $sup(r, E) > sup(r', E)$, or
– $conf(r, E) = conf(r', E)$ and $sup(r, E) = sup(r', E)$ and $|r| < |r'|$.

2.4 Classifiers

We now propose a definition of a rule-based classifier. Informally, a rule-based classifier consists of a set of rules, used in a given order during the prediction process, to classify unlabeled objects.

Definition 1. (Classifier) *Let $\mathcal{C}$ be the set of all classifiers. A classifier is a tuple $\langle R, <_R \rangle$ where R is a set of rules and $<_R$ is an order on R. If $C = \langle R, <_R \rangle$ is a classifier and r is a rule, we use $|C|$ to denote $|R|$ and $r \in C$ to denote $r \in R$.*

Classifier inclusion Let $C_1 = \langle R_1, <_{R_1} \rangle$ and $C_2 = \langle R_2, <_{R_2} \rangle$ be two classifiers. We say that C_1 is included in C_2, noted $C_1 \sqsubseteq C_2$, if $R_1 \subseteq R_2$ and $<_{R_1} \subseteq <_{R_2}$

Examples covered Let E be a dataset and C be a classifier. We distinguish three categories of examples covered by a classifier:

- $covered(E, C) = \{e \in E | \exists r \in C, r \lhd e\}$
- $covered^+(E, C) = \{e \in E | \exists r \in C, r \lhd e \wedge cl(r) = cl(e)\}$
- $covered^-(E, C) = \{e \in E | \exists r \in C, r \lhd e \wedge cl(r) \neq cl(e)\}$

Default classifier A default classifier is a function from $2^{\mathcal{E}}$ to $\mathcal{C}$, that gives a particular classifier used to provide a default class label. In this paper, given a set of examples E, we restrict to the function outputing the classifier $\langle \{\emptyset \to c\}, \emptyset \rangle$ where c is such that $nb(c, E) = max\{nb(c', E) | c' \in \{cl(e) | e \in E\}\}$ where for a given c', $nb(c', E) = |\{e \in E / cl(e) = c'\}|$.

Operators on classifiers Our description of the classifier construction process is incremental. The following operators show how a classifier can be obtained using operators on classifiers.

Concatenation Let $C_1 = \langle R_1, <_{R_1} \rangle$ and $C_2 = \langle R_2, <_{R_2} \rangle$ be two classifiers such that $R_1 \cap R_2 = \emptyset$. The concatenation of the two classifiers is defined as $C_1 \centerdot C_2 = \langle R_1 \cup R_2, <_{\centerdot} \rangle$, where $<_{\centerdot} = <_{R_1} \cup <_{R_2} \cup (R_1 \times R_2)$. Note that operator $\centerdot$ does not commute, since $\times$ does not commute.

Union Let $C_1 = \langle R_1, <_{R_1} \rangle$ and $C_2 = \langle R_2, <_{R_2} \rangle$ be two classifiers such that $R_1 \cap R_2 = \emptyset$. The union of the two classifiers is defined as $C_1 \cup C_2 = \langle R_1 \cup R_2, <_{\cup} \rangle$, where $<_{\cup} = <_{R_1} \cup <_{R_2}$.

Difference Let $C_1 = \langle R_1, <_{R_1} \rangle$ and $C_2 = \langle R_2, <_{R_2} \rangle$ be two classifiers. The difference $C_1 \setminus C_2$ is the classifier $\langle R_1 \setminus R_2, <_{R_1} \setminus \{\langle r, r' \rangle \in <_{R_1} | r \in R_2 \vee r' \in R_2\} \rangle$.

By abuse of notation we sometime write $C_1 \setminus R_2$ to denote $C_1 \setminus \langle R_2, \emptyset \rangle$ and $R_2 \setminus C_1$ to denote $R_2 \setminus R_1$.

2.5 Prediction

Finally we give the definitions used for describing the predication process. A class prediction operator is a mapping from $\mathcal{O} \times \mathcal{C}$ to $\mathcal{E}$. Below are two examples of a class prediction operator.

Best rule prediction Let $C = \langle R, <_R \rangle$ be a classifier such that $<_R$ is a total order, and o be an object. $BestRule(o, C) = \langle o, cl(r_p)\rangle$ where $r_p = max_{<_R}\{r \in R\}$.

Aggregate prediction For a rule r and a dataset E, let w be a function from $\mathcal{R} \times 2^{\mathcal{E}}$ to $\mathbb{R}$ that gives the value of a measure (e.g., confidence) of r w.r.t E.

Let $C = \langle R, <_R \rangle$ be a classifier, $cl(C)$ be the class labels of C i.e., the set $\{c \in Class | \exists o \in \mathcal{O}, o \rightarrow c \in R\}$. Given an object o, a class label $c_i \in cl(C)$, a data set E and an aggregate function agg of signature $2^{\mathbb{R}} \rightarrow [0, 1]$ and an $\mathcal{E}$-order α defined by $\alpha(r, r', E) = true$ if $w(r, E) > w(r', E)$, we define for each class c_i the set $R(c_i) = Top_k(\{o' \rightarrow c_i \in R | o' \lhd o\}, E, \alpha)$ of the k best rules covering o, and $W(c_i) = agg\{w(r, E) | r \in R(c_i)\}$ the aggregate of the measure value of these rules.

Then $AggPred_{k,w,agg}(o, C) = \langle o, c_p\rangle$ where c_p is such that $W(c_p) = max\{W(c_i) | c_i \in cl(C)\}$.

Error Rate Given a set of examples E, a classifier C and a prediction operator $Pred$, we denote by $Error(E, C, Pred)$ the ratio of examples in E misclassified by C using $Pred$, i.e. $Error(E, C, Pred) = \frac{|\{\langle o,c\rangle \in E \ | \ \langle o,c\rangle \neq Pred(o,C)\}|}{|E|}$.

3 ICCA: A Generic Incremental Classifier Construction Algorithm

In this section we present a generic classifier construction algorithm for building any classifier presented under rule-induction, associative, or instance-centric approaches, and therefore, any combination of them.

Given a set of examples E, a classifier C can be seen as the best set of rules w.r.t a data-dependent order O over classifiers: $C = Top_1(\mathcal{C}, E, O)$. Obviously, evaluating the interestingness of a classifier (i.e., the definition of O) is a hard task [4]. Assuming that O is known, finding the best classifier remains a challenge due to the huge search space $\mathcal{C}$. To the best of our knowledge, only one work [14] addresses the *exact* resolution of this problem. It aims to extract a set of (individual) pattern sets each satisfying a constraint at pattern set level. The adaptation of local pattern mining techniques into pattern set mining suffers from efficiency as well as scalability viewpoints. On the one hand, the efficiency of this approach relies on specific *boundability* property of O. On the other hand, the proposed algorithm doesn't seem to be feasible on large databases. For this purpose, most of the existing classification methods use heuristic algorithms for building a rule-based classifier C which is an *approximate* solution of $Top_1(\mathcal{C}, E, O)$. Basically, such methods try to build a global optimal set of rules by iteratively selecting the best rule (i.e., local optimal rule).

3.1 Principles and Parameters

Our generic classifier construction algorithm is named $ICCA$ (Incremental Classifier Construction Algorithm). Intuitively, it iteratively constructs a classifier based on a training database starting from an empty classifier. The basic principle is that at each iteration of $ICCA$, a set of best rules w.r.t. a local selection predicate is concatenated to the classifier constructed at the previous iteration, thus making the resulting classifier more accurate (and more predictive).

Before presenting $ICCA$ (see Algorithm 1) in more details, let us first start with a short description of $ICCA$'s parameters:

- $\mathcal{R}$ is a rule language.
- E is a training database.
- K is the number of rules that should be extracted at each iteration of the algorithm.
- $OrdGen$ is a function used to generate an $\mathcal{E}$-order on $\mathcal{R}$ from a classifier (i.e., mainly a set of rules). For instance, if C is a classifier, and E is the set of examples not yet covered by the rules of C, $OrdGen(C) = \alpha$ where α is such that, for every $r, r' \in \mathcal{R}^2$ $\alpha(r, r', E) = true$ iff the confidence of r is greater than the confidence of r' or, if they are equal, the size of r is smaller than the size of r'.
- $PredGen$ is a function used to describe a selection predicate for $\mathcal{R}$ under which the extraction of a set of rules is taking place. For example, if C is a classifier and E is the set of examples not yet covered by the rules of C, $PredGen(C) = q$ where q is such that, for every $r \in \mathcal{R}$ we have $q(r, E) = true$ iff the support of r is greater than a given threshold.
- V is a function used to describe a selection predicate for $\mathcal{C}$, in order to evaluate the quality (or accuracy) of a classifier w.r.t the whole training dataset. For example, if C is a classifier and E a dataset, $V(C, E)$ can be such that it outputs $true$ iff C covers all examples of E.
- O is an $\mathcal{E}$-order on $\mathcal{C}$. For example, if C_1 and C_2 are two classifiers and E is a dataset, $O(C_1, C_2, E) = true$ iff the number of examples covered by C_1 is greater than that of C_2.

The next Section describes the algorithm more detailed.

3.2 The *ICCA* Algorithm

Suppose that $ICCA$ is called with: $ICCA(\mathcal{R}, E, k, OrdGen, PredGen, V, O)$. $ICCA$ starts from an empty classifier (Line 1) and constructs its output classifier by iterating until the classifier satisfies the V validation predicate or no more rules can be added to the classifer (Line 8). At each step of the loop, a classifier is constructed with a set of k best rules along with a (local) ordering relation over them. The rules are extracted with the Th operation (Line 5), where the classifier constructed during the previous step is taken into account for generating the selection predicate (Line 4) and ignoring the rules already extracted (Line 5). Out of the rules extracted with Th, the best ones are extracted with the Top_k operation (Line 6), where here again the rules extracted during the previous steps are taken into account for generating the $\mathcal{E}$-order used for ranking (Line 4). Then a new classifier is obtained by concatenating the classifier constructed

Algorithm 1 $ICCA(\mathcal{R}, E, K, OrdGen, PredGen, V, O)$

```
Input: A rule language $\mathcal{R}$, a training dataset $E$, an integer $K$, an order generator
    function $orderGen$, a predicate generator function $PredGen$, a classifier validation
    predicate $V$ and a classifier $\mathcal{E}$-order $O$
Output: A classifier $C$
 1: $i = 0$ and $C_0 = \langle \emptyset, \emptyset \rangle$
 2: repeat
 3:     $i = i + 1$
 4:     $\alpha_i = OrdGen(C_{i-1})$ and $q_i = PredGen(C_{i-1})$
 5:     $R_i = Th(\mathcal{R} \setminus C_{i-1}, E, q_i)$
 6:     $T_i = Top_K(R_i, E, \alpha_i)$
 7:     $C_i = C_{i-1} \cdot \langle T_i, \alpha_i[E] \rangle$
 8: until $(V(C_i, E) = true)$ or $C_i = C_{i-1}$
 9: $C = Top_1(\{C_k \mid 1 \leq k \leq i\}, E, O)$
10: return $C$
```

during the previous steps with the newly extracted best rules along with the order relation on them (Line 7). Finally, out of all the classifiers constructed during the loop, the best one w.r.t O is returned (Line 9 and 10).

The proposed operators and ICCA uniformly describe in a as declarative as possible fashion various classification approaches and algorithms, that have their own requirements and properties. To the best of our knowledge, this is the first time that all the related definitions are formally specified in the classification context. In order to illustrate the generality of our framework, the next section shows how our approach integrates and represents requirements of different classification methods using ICCA. Optimization aspects are also discussed in Section 5.

4 Representation of Existing Classifiers in *ICCA* Framework

In this section, we show how existing classification algorithms can be described in the *ICCA* framework. Some algorithms are described in detail (*AQ*, *CN*2, *CBA*, *FOIL* and *HARMONY*). Due to lake of space, we only summarize the main features of other algorithms (*CMAR*, *CorClass* and *PRM*). It is important to note that the diversity of the methods, and hence exploited heuristics, makes the notations to be rather complicated. Further details, e.g. interestingness measures, can be found in the respective references for each method.

4.1 AQ

For multi-class classification problem, *AQ* needs to be applied multiple times, each time mining the rules for one class. In our framework, it means that *ICCA* will be executed for each class in *Class*. Then, a global classifier is obtained using the union operator between classifiers. More formally, the AQ classifier, denoted by C_{AQ}, is defined by:

$$C_{AQ} = C'_{AQ} \centerdot C_{default}(E \setminus covered(E, C'_{AQ}))$$

where $C'_{AQ} = \bigcup_{j=1}^{NbClass} AQ_j$ and for every $j \in \{1, \ldots, NbClass\}$:

$$AQ_j = ICCA(\mathcal{R}, E, 1, OrdGen_{AQ_j}, PredGen_{AQ_j}, V_{AQ_j}, O_{AQ_j})$$

and the functions $OrdGen_{AQ_j}$, $PredGen_{AQ_j}$, V_{AQ_j}, and O_{AQ_j} are defined for every classifier $C, C_1, C_2 \in \mathcal{C}$ and class $c_j \in Class$ by:

- $OrdGen_{AQ_j}(C) = \alpha_{AQ_j}$ where for every $r, r' \in \mathcal{R}^2$ and $E \subseteq \mathcal{E}$, we have $\alpha_{AQ_j}(r, r', E) = true$ iff:
  - $sup(r, E') > sup(r', E')$ or
  - $sup(r, E') = sup(r', E') \wedge |r| < |r'|$

  where $E' = E_j \setminus covered(E_j, C)$.
- $PredGen_{AQ_j}(C) = q_j^{seed}$ where for every $r \in \mathcal{R}$ and $E \subseteq \mathcal{E}$, $q_j^{seed}(r, E) = true$ iff:
  - $cl(r) = c_j$, and
  - $r \lhd seed$ where $seed = max_{\prec_{\mathcal{E}}}(E_j \setminus covered(E_j, C))$, i.e. r covers correctly an example of E of class c_j which is not yet classified correctly by any rule of C, and
  - $(\nexists e \in E \setminus E_j)(r \lhd e)$, i.e. r does not cover an example of E that is of class $c_k \neq c_j$.
- For every $E \subseteq \mathcal{E}$, $V_{AQ_j}(C, E) = true$ iff $E_j = covered(E_j, C)$, i.e. C covers all examples of E of class c_j.
- For every $E \subseteq \mathcal{E}$, $O_{AQ_j}(C_1, C_2, E) = true$ iff $covered(E_j, C_1) \supset covered(E_j, C_2)$, i.e. C_1 covers more example of E of class c_j than C_2.

In practice, note that the last constraint in the definition of q_j^{seed} should be relaxed to overcome the problem of multi-label (contradictory) training examples.

Given an object $o \in \mathcal{O}$ and a classifier C_{AQ}, the prediction operator used by AQ is defined by: $Predict_{AQ}(o, C_{AQ}) = AggPred_{+\infty, w, agg}(o, C_{AQ})$, where for every rule $r \in \mathcal{R}$ and data sets $E \subseteq \mathcal{E}$, $w(r, E)$ is the support of rule r in E, and agg is the probabilistic sum ($Psum$) aggregation [15].

4.2 CN2

By comparison with AQ, the $CN2$ algorithm builds directly one classifier for multi-class problems. Formally, the $CN2$ classifier, denoted by C_{CN2}, is defined by:

$$C_{CN2} = C'_{CN2} \centerdot C_{default}(E \setminus covered(E, C'_{CN2})) \text{ with}$$
$$C'_{CN2} = ICCA(\mathcal{R}, E, 1, OrdGen_{CN2}, PredGen_{CN2}, V_{CN2}, O_{CN2})$$

where the functions $OrdGen_{CN2}$, $PredGen_{CN2}$, V_{CN2} and O_{CN2} are defined for every classifier $C, C_1, C_2 \in \mathcal{C}$ as follows:

– $OrdGen_{CN2}(C)= \alpha_{CN2}$ where for every $r, r' \in \mathcal{R}^2$ and $E \subseteq \mathcal{E}$, we have $\alpha_{CN2}(r, r', E) = true$ iff:

  - $entropy(r, E') < entropy(r', E')$, or
  - $entropy(r, E') = entropy(r', E')$ and $|r| < |r'|$

  where $E' = E \setminus covered(E, C)$.
– $PredGen_{CN2}(C) = q_{CN2}$ where for every $r \in \mathcal{R}$ and $E \subseteq \mathcal{E}$, $q_{CN2}(r, E) = true$ iff $F(r, E') \geq \tau$, where F is a significance measure such as χ^2 (or *Likelihood ratio*), τ is a minimum threshold and $E' = E \setminus covered(E, C)$.
– For every $E \subseteq \mathcal{E}$, $V_{CN2}(C, E) = true$ iff $E = covered(E, C)$.
– For every $E \subseteq \mathcal{E}$, $O_{CN2}(C_1, C_2, E) = true$ iff $|C_1| > |C_2|$.

The result classifier C_{CN2} is totally ordered. Therefore,the *BestRule* prediction operator can be used by $CN2$, i.e. for every object $o \in \mathcal{O}$, we have: $Pred_{CN2}(o, C_{CN2}) = BestRule(o, C_{CN2})$.

4.3 CBA

As $CN2$, CBA builds directly one classifier for multi-class problems. Formally, the CBA classifier, denoted by C_{CBA}, can be specified as follows:

$$C_{CBA} = ICCA(\mathcal{R}, E, 1, OrdGen_{CBA}, PredGen_{CBA}, V_{CBA}, O_{CBA})$$

where the functions $OrdGen_{CBA}$, $PredGen_{CBA}$, V_{CBA}, and O_{CBA} are defined for every classifier $C, C_1, C_2 \in \mathcal{C}$ by:

– $OrdGen_{CBA}(C) = \alpha_{CBA}$ where for every $r, r' \in \mathcal{R}^2$ and $E \subseteq \mathcal{E}$, we have $\alpha_{CBA}(r, r', E) = true$ iff $\alpha_{csc}(r, r', E) = true$.
– $PredGen_{CBA}(C) = q_{CBA}$ where for every $r \in \mathcal{R}$ and $E \subseteq \mathcal{E}$, $q_{CBA}(r, E) = true$ iff:

  - $sup(r, E) \geq \alpha$ and $conf(r, E) \geq \beta$, and
  - for all direct generalization r' of r, $PessError(r', E) > PessError(r, E))$, where *PessError* is the pessimistic error rate, and
  - $(\exists e \in E \setminus covered(E, C))(r \lhd e \wedge cl(r) = cl(e))$.

– For every $E \subseteq \mathcal{E}$, $V_{CBA}(C, E) = true$ iff $E = covered(E, C)$.
– For every $E \subseteq \mathcal{E}$, $O_{CBA}(C_1, C_2, E) = true$ iff $Error(C'_1, E, Pred_{CBA}) \leq Error(C'_2, E, Pred_{CBA})$ where $C'_i = C_i \centerdot C_{default}(E \setminus covered(E, C_i))$ $(i = 1, 2)$.

As for $CN2$, since the result classifier C_{CBA} is totally ordered, CBA can used the *BestRule* prediction operator. For every object $o \in \mathcal{O}$, we have: $Pred_{CBA}(o, C_{CBA}) = BestRule(o, C_{CBA})$.

CMAR and CorClass In this paper, we do not describe $CMAR$ in detail since $CMAR$ is mainly an extension of CBA. In comparison with CBA, $CMAR$ selects only positively correlated rules (by χ^2 testing). Moreover, instead of removing an example as soon as it is covered by a rule, it only removes examples that are covered by more than δ rules, where δ is a parameter of $CMAR$. Finally, in the prediction operator used by $CMAR$, instead of confidence, $w(r, E)$ is the $weighted{-}\chi^2$; this measure is used to overcome the minority class favoring problem.

In comparison with CBA and $CMAR$, $CorClass$ directly extracts the k rules with the highest significance measures (χ^2, information gain, etc.) on the data set, meaning that $ICCA$ iterates only once. On the other hand, the classifiers built by $CorClass$ are evaluated using different prediction operators (Best Rule or Aggregate prediction with different weighted combinations of rules).

4.4 FOIL

As the algorithm AQ, the algorithm $FOIL$ has to be applied on each class for multi-class problems. Moreover, in order to compare rules, $FOIL$ uses a specific gain measure defined as follows.

Definition 2. (Foil Gain). *Given two rules r_1 and r_2 such that $r_2 \prec_{\mathcal{R}} r_1$, the gain to specialize r_2 to r_1 w.r.t. a set of examples E is defined by:*

$$gain(r_1, r_2, E) = |P_1|(log(\frac{|P_1|}{|P_1| + |N_1|}) - log(\frac{|P_2|}{|P_2| + |N_2|}))$$

where $P_2 = covered^+(E, C_2)$, $N_2 = covered^-(E, C_2)$, $P_1 = covered^+(P_2 \cup N_2, C_1)$, $N_1 = covered^-(P_2 \cup N_2, C_1)$ with $C_i = \langle \{r_i\}, \emptyset \rangle$ $(i = 1, 2)$.

Formally, the $FOIL$ classifier, denoted by C_{FOIL}, is specified by $C_{FOIL} = \bigcup_{j=1}^{NbClass} C^j_{FOIL}$ where for every $j \in \{1, \ldots, NbClass\}$:

$$C^j_{FOIL} = ICCA(\mathcal{R}, E, 1, OrdGen^j_{FOIL}, PredGen^j_{FOIL}, V^j_{FOIL}, O^j_{FOIL})$$

where the functions $OrdGen^j_{FOIL}$, $PredGen^j_{FOIL}$, V^j_{FOIL}, and O^j_{FOIL} are defined for any classifiers C, C_1, C_2 by:

- $OrdGen^j_{FOIL}(C) = \alpha^j_{FOIL}$ where for every $r_1, r_2 \in \mathcal{R}^2$ and $E \subseteq \mathcal{E}$, we have $\alpha^j_{FOIL}(r_1, r_2, E) = true$ iff $gain(r_1, r_1 \wedge r_2, E') > gain(r_2, r_1 \wedge r_2, E')$ where $r_1 \wedge r_2$ is the most specific rule that is more general than r_1 and r_2 ($r_1 \wedge r_2 = min_{\prec_{\mathcal{R}}}\{r \in \mathcal{R} \mid r \prec_{\mathcal{R}} r_1, r \prec_{\mathcal{R}} r_2\}$) and $E' = E_j \setminus covered(E_j, C)$.
- $PredGen^j_{FOIL}(C) = q^j_{FOIL}$ where for every $r \in \mathcal{R}$ and $E \subseteq \mathcal{E}$, we have $q^j_{FOIL}(r, E) = true$ iff $cl(r) = c_j$ and $|r| \leq L$ where L is a parameter of $FOIL$.
- For every $E \subseteq \mathcal{E}$, $V^j_{FOIL}(C, E) = true$ iff $E_j = covered(E_j, C)$.
- For every $E \subseteq \mathcal{E}$, $O^j_{FOIL}(C_1, C_2, E) = true$ iff $|C_1| > |C_2|$.

Given a classifier C_{FOIL}, $FOIL$ can use the prediction operator defined for every object $o \in \mathcal{O}$ by: $Predict_{FOIL}(o, C_{AQ}) = AggPred_{+\infty,w,agg}(o, C_{FOIL})$, where for every rule $r \in \mathcal{R}$ and data sets $E \subseteq \mathcal{E}$, $w(r, E)$ is the confidence of rule r in E, and agg is the sum aggregation function.

PRM In this paper, we do not describe PRM in detail since PRM is mainly an extension of $FOIL$. By comparison with $FOIL$, after an example is correctly covered by a rule, PRM does not remove it. More precisely, a weight is associated to every example of the data set, and when an example is correctly covered by a new rule, its weight is decreased. These weights are mainly used to evaluate the gain of a rule specialization. They are also used to stop the incremental construction of a classifier. On the other hand, note that these weights are only used during the construction of the classifier ; They are not used by the prediction operator of PRM.

4.5 HARMONY

In comparison with the previous classification methods, $HARMONY$ uses an instance-centric rule generation procedure, meaning that it mines for each training example the K highest confidence rules that cover it correctly. Then, $HARMONY$ orders the rules w.r.t. their class label, confidence and support.

In order to specify the classifier build by $HARMONY$, we introduce in the following definition of an operator *orderby* that allows to re-order the rules of a classifier.

Definition 3. (Order By.) *Given a rule $\mathcal{E}$-order α, the orderby operator is defined for every classifier $C = \langle R, >_R \rangle$ and data set E by: $orderby_{\alpha}(C, E) = \langle R, \alpha[E] \rangle$.*

Using this definition, the $HARMONY$ classifier, denoted by C_{HARM}, is defined for every data set E by: $C_{HARM} = orderby_{\alpha_{ALL}}(\bigcup_{e \in E} C^e_{HARM}, E)$ where:

- For every r, r' in $\mathcal{R}$ and data set $E \subseteq \mathcal{E}$, $\alpha_{ALL}(r, r', E) = true$ if:
  - $cl(r) = cl(r')$ and $conf(r, E) > conf(r', E)$, or
  - $cl(r) = cl(r')$, $conf(r, E) = conf(r', E)$ and $sup(r, E) > sup(r', E)$.
- For every training example e, C^e_{HARM} is specified by:

  $$C^e_{HARM} = ICCA(\mathcal{R}, E, k, OrdGen_{HARM}, PredGen^e_{HARM}, V_{HARM}, O_{HARM})$$

  where the functions $OrdGen_{HARM}$, $PredGen^e_{HARM}$, V_{HARM} and O_{HARM} are defined for every classifier $C, C_1, C_2 \in \mathcal{C}$ by:
  - $OrdGen_{HARM}(C) = \alpha_{HARM}$ where for every $r, r' \in \mathcal{R}^2$ and $E \subseteq \mathcal{E}$, we have $\alpha_{HARM}(r, r', E) = true$ iff $conf(r, E) > conf(r', E)$.
  - $PredGen^e_{HARM}(C) = q^e$, where for every $r \in \mathcal{R}$ and $E \subseteq \mathcal{E}$, $q^e(r, E) = true$ iff $r \lhd e$ and $cl(r) = cl(e)$ and $sup(r, E) \geq \tau$ where τ is a minimum support threshold.

- For every $E \subseteq \mathcal{E}$, $V_{HARM}(C, E) = true$ for every classifier C and training data set E.
- For every $E \subseteq \mathcal{E}$, $O_{HARM}(C_1, C_2, E) = true$ iff $|C_1| > |C_2|$.

Given an object $o \in \mathcal{O}$ and a classifier C_{HARM}, the prediction operator used by $HARMONY$ is defined by: $Pred_{HARM}(o, C_{HARM}) = AggBestPred_{k,w,agg}(o, C_{HARM})$ where for every rule $r \in C$ and data set E, $w(r, E) = conf(r, E)$ and agg is the sum aggregation function.

5 Analyzing and Optimizing ICCA

This section aims at analyzing the properties of the different classification methods within our framework in order to compare them. These properties also provide interesting optimizations allowing us to improve the efficiency of ICCA.

5.1 A comparative study of classification methods in our framework

Before detailing the comparison between all the classification methods, we introduce two important properties on the parameters of ICCA. The latter has an important impact on the construction of classifiers.

At first, many methods use the same rule $\mathcal{E}$-order to rank the rules all along the construction of the classifier. This property is formally defined below:

Definition 4 (P1: Constant Order Generator). *An order generator function OrdGen is constant iff there exists a rule $\mathcal{E}$-order α such that $\forall C \in \mathcal{C}$, $OrdGen(C) = \alpha$.*

In other terms, whenever P1 is satisfied by a method (e.g., CBA or HARMONY), the ranking over the rules is not modified by the rules added in the classifier. Typically, $OrdGen_{CBA}$ stemming from information about confidence, support and generalization on the entire database, is independent of the classifier. In Section 5.2, P1 is taken into account by sorting the rules only once before starting the iterative phase of the algorithm.

Moreover, most of the methods start from an initial collection of rules and refine it during the classifier construction. The following definition expresses this property:

Definition 5 (P2: Monotone Predicate Generator). *A predicate generator function PredGen is monotone iff $\forall C_1, C_2 \in \mathcal{C}$ such that $C_1 \sqsubseteq C_2$, one have $PredGen(C_2) \Rightarrow PredGen(C_1)$.*

Intuitively, Property P2 is satisfied for all the methods which compute a smaller theory when the classifier contains more rules. ICCA can benefit from this property by computing only once the theory at the initialization step. For instance, $PredGen_{CBA}$ satisfies Property P2 since the larger the classifier, the more selective the predicate $(\exists e \in E \setminus covered(E, C))(r \lhd e \wedge cl(r) = cl(e))$. Thereafter, for each step, the algorithm reduces the previous theory by removing

Features		AQ	CN2	CBA	CMAR	FOIL	PRM	CorC	HARM
Representing Level	Number of calls of ICCA	$NbClass$	1	1	1	$NbClass$	$NbClass$	1	$\|E\|$
Parameters of ICCA	P1: constant $OrdGen$			×	×			×	×
	P2: monotone $PredGen$	×		×	×			×	×
	K: number of rules per iteration in ICCA	1	1	1	≥ 1	1	≥ 1	1	1
Resulting classifier	order type (P=Partial, T=Total)	P	T	T	T	P	P	T	P
Prediction	BestRule		×	×				×	
	AggPred w	$conf$			$wght$-χ^2	$Laplace$	$Laplace$		$(wght\text{-})conf$
	agg	$Psum$			sum	avg	avg		avg/sum
	k	$+\infty$			$+\infty$	k	k		k or $+\infty$

Table 1. Comparison of the classifiers based on features in RBC framework ($wght$ means weighted).

uninteresting rules (e.g., already covered rules) according to the classifier in progress instead of computing a new theory.

Based on our generic framework, Table 5.1 sums up and compares many various classification methods including those listed in Section 4. The first row indicates for each method the number of required calls to ICCA. The second part describes the main parameters of ICCA. Then, it provides information about the satisfaction of Properties P1 and P2 which lead to the optimizations described below. The number of selected rules for each iteration in ICCA is also specified. The third part depicts the order type of the final classifier. Finally, the last part summarizes the used prediction method(s) with their own parameters (i.e., w, agg and k).

Of course, we observe that *BestRule* prediction is always performed on totally ordered rules. More interestingly, Table 5.1 shows that Property $P1$ implies Property $P2$. Indeed, when the order is constant during the extraction (i.e., the interest of each rule remains the same), an uninteresting rule in a given step is not relevant in the subsequent steps.

5.2 ICCA^{opt}: a generic optimized algorithm

This section presents an optimized version of ICCA skecthed by Algorithm 2 which inputs and result are similar to those of ICCA (see Section 3.2). We just describe here the different optimizations relying on Properties P1 and P2 given in the previous section. Indeed, these properties decrease the computation cost of the theory which is the most difficult step of ICCA (see Algorithm 1, Line 5). Let us note that other sophisticated optimizations based on properties of predicate q_i (e.g., anti-monotonicity, boundable constraints) are not discussed here.

Line 2 computes R_0 which is a superset of all theories obtained with any classifier C_i. In other terms, the predicate $PredGen^* = \bigvee_{C \in \mathcal{C}} PredGen(C)$ is a constant predicate and allows us to restrain the language of rules potentially interesting for the classifier construction. In particular, as $\forall i \geq 0$, $C_i \sqsubseteq C_{i+1}$, we straightforwardly deduce that $PredGen^* = PredGen(\langle \emptyset, \emptyset \rangle)$ when P2 is satisfied. For instance, $PredGen_{CBA}{}^*(r, E) = sup(r, E) \geq \alpha \wedge conf(r, E) \geq \beta \wedge (\nexists r' \in$

Algorithm 2 $ICCA^{opt}(\mathcal{R}, E, K, OrdGen, PredGen, V, O)$

```
Input: A rule language $\mathcal{R}$, a training data set $E$, an integer $K$, an order generator function $OrdGen$,
    a predicate generator function $PredGen$, a classifier validation predicate $V$ and a classifier $\mathcal{E}$-
    order $O$
Output: A classifier $C$
 1: $i = 0$ and $C_0 = \langle \emptyset, \emptyset \rangle$
 2: $R_0 = Th(\mathcal{R}, E, PredGen^*)$
 3: if $OrdGen$ is constant then $>_0 = OrdGen(C_0)[E]$ and $C_{to} = \langle R_0, >_0 \rangle$
 4: repeat
 5:   $i = i + 1$
 6:   $q_i = PredGen(C_{i-1})$
 7:   if $OrdGen$ is constant then
 8:     $T_i = \emptyset$ and $C_{cur} = C_{to}$
 9:     while $(|T_i| < k) \wedge (C_{cur} \neq \emptyset)$ do
10:       $R_i = max_{>_0}(C_{cur})$
11:       $C_{cur} = C_{cur} \setminus R_i$
12:       $T_i = T_i \cup Th(R_i, E, q_i)$
13:     od
14:     $C_{to} = C_{to} \setminus T_i$
15:   else
16:     $\alpha_i = OrdGen(C_{i-1})$
17:     if $PredGen$ is isotone then
18:       $R_i = Th(R_{i-1} \setminus C_{i-1}, E, q_i)$
19:     else
20:       $R_i = Th(R_0 \setminus C_{i-1}, E, q_i)$
21:     end if
22:     $T_i = Top_K(R_i, E, \alpha_i)$
23:   end if
24:   $C_i = C_{i-1} \circ \langle T_i, \alpha_i[E] \rangle$
25: until $(V(C_i, E) = true)$ or $C_i = C_{i-1}$
26: $C = Top_1(\{C_k \mid 1 \leq k \leq i\}, E, O)$
27: return $C$
```

$\mathcal{R})(r$ is a direct specialization of $r' \wedge PessError(r', E) < PessError(r, E))$ (see CBA description in Section 4).

Line 3 computes the $\mathcal{E}$-order $>_0$ and ranks the rules with C_{to} iff $OrdGen$ satisfies P1. In such case, $>_0$ and C_{to} are used all along the main loop for computing T_i by selecting the best rules (w.r.t $>_0$) from C_{to} satisfying q_i (Line 9-13).

When $PredGen$ is monotone, Line 18 directly finds the theory R_i by selecting the rules among $R_{i-1} \setminus C_{i-1}$ and satisfying q_i. As $R_{i-1} \setminus C_{i-1}$ is smaller (and, in general, much smaller) than R_0 (which is used when $PredGen$ is not monotone, see Line 20), this optimization really improves the algorithm's efficiency.

6 Conclusion

This paper proposes a generic framework for rule-based classification: rule-induction, association-based or instance-centric classifiers. This framework encompasses a broad spectrum of existing methods including AQ, CN2, CBA, CMAR, FOIL, PRM, CorClass and HARMONY. The classifier construction of such approaches is uniformly described thanks to the general algorithm ICCA. We similarly give two general prediction functions: *BestRule* and *AggPred*. Finally, we compare the main features of the described methods within our framework. In particular, two key properties allow us to improve the efficiency of ICCA by reducing the computational cost of theories.

Further work addresses the generalization of this framework to other kinds of patterns (e.g., sequences or trees) in order to naturally extend existing classifiers to more complex data. We would like also to implement ICCA and to test other classification methods by defining new ICCA's parameters. Furthermore, it would be interesting to examine the interest of our approach for other global model construction such as clustering.

References

1. Peter Clark and Tim Niblett. The cn2 induction algorithm. *Mach. Learn.*, 3(4):261–283, 1989.
2. William W. Cohen. Fast effective rule induction. In Armand Prieditis and Stuart Russell, editors, *Proc. of the 12th International Conference on Machine Learning*, pages 115–123, Tahoe City, CA, july 1995. Morgan Kaufmann.
3. Guozhu Dong, Xiuzhen Zhang, Limsoon Wong, and Jinyan Li. Caep: Classification by aggregating emerging patterns. In *Discovery Science*, pages 30–42, 1999.
4. Arno J. Knobbe and Eric K. Y. Ho. Pattern teams. In *PKDD*, pages 577–584, 2006.
5. Jinyan Li, Guozhu Dong, and Kotagiri Ramamohanarao. Instance-based classification by emerging patterns. In *PKDD '00: Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery*, pages 191–200, London, UK, 2000. Springer-Verlag.
6. Jinyan Li, Guozhu Dong, and Kotagiri Ramamohanarao. Making use of the most expressive jumping emerging patterns for classification. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 220–232, 2000.
7. Wenmin Li, Jiawei Han, and Jian Pei. Cmar: Accurate and efficient classification based on multiple class-association rules. In *ICDM '01: Proceedings of the 2001 IEEE International Conference on Data Mining*, pages 369–376, Washington, DC, USA, 2001. IEEE Computer Society.
8. Bing Liu, Wynne Hsu, and Yiming Ma. Integrating classification and association rule mining. In *Knowledge Discovery and Data Mining*, pages 80–86, 1998.
9. Ryszard S. Michalski. On the quasi-minimal solution of the general covering problem. In *Proceedings of the V International Symposium on Information Processing (FCIP 69)(Switching Circuits*, volume A3, pages 125–128, 1969.
10. Yasuhiko Morimoto, Takeshi Fukuda, Hirofumi Matsuzawa, Takeshi Tokuyama, and Kunikazu Yoda. Algorithms for mining association rules for binary segmentations of huge categorical databases. In *VLDB '98: Proceedings of the 24rd International Conference on Very Large Data Bases*, pages 380–391, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.
11. J. Ross Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.
12. J. Ross Quinlan. *C4.5: programs for machine learning.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.
13. J. Ross Quinlan and R. Mike Cameron-Jones. FOIL: A midterm report. In *Machine Learning: ECML-93, European Conference on Machine Learning, Proceedings*, volume 667, pages 3–20. Springer-Verlag, 1993.
14. Luc De Raedt and Albrecht Zimmermann. Constraint-based pattern set mining. In *SDM*, 2007.

15. Michalski Ryszard S., Mozetic Igor, Hong Jiarong, and Lavrac Nada. The aq15 inductive learning system: An overview and experiments. In *Reports of the Intelligent Systems Group, ISG 86-20, UIUCDCS-R-86-1260*, 1986.
16. Jianyong Wang and George Karypis. Harmony: Efficiently mining the best rules for classification. In *SDM*, 2005.
17. Xiaoxin Yin and Jiawei Han. Cpar: Classification based on predictive association rules. In *SDM*, 2003.
18. Albrecht Zimmermann and Luc De Raedt. Corclass: Correlated association rule mining for classification. In *Discovery Science*, pages 60–72, 2004.