RULE SET QUALITY MEASURES FOR INDUCTIVE LEARNING ALGORITHMS

by

RALF KLINKENBERG, 1971 -

A THESIS

Presented to the Faculty of the Graduate School of the

UNIVERSITY OF MISSOURI - ROLLA

In Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE IN COMPUTER SCIENCE

1996

Approved by

_____	_____
Daniel C. St. Clair (Advisor)	Ralph W. Wilkerson

_____

Cihan H. Dagli

PUBLICATION THESIS OPTION

This thesis has been prepared in accordance with the format used by The American Society of Mechanical Engineers (ASME) PRESS for the proceedings of the annual conference **Artificial Neural Networks In Engineering (ANNIE)**.

Pages 1-46 were published in the proceedings of the Artificial Neural Networks In Engineering Conference 1996 (ANNIE '96) in an abridged version (Klinkenberg and St. Clair, 1996). Appendices A to F have been added for purposes normal to thesis writing.

ABSTRACT

Symbolic inductive learning systems that induce concept descriptions from examples are valuable tools in the task of knowledge acquisition for expert systems. Since inductive learning methods produce distinct concept descriptions when given identical training data, questions arise as to the quality of the different rule sets produced. This work provides several techniques for comparing and analyzing rule sets. These techniques measure the accuracy, generalization, time and space complexity, and domain coverage of rule sets. Based on these metrics, the performance of four different inductive learning systems is compared. These systems are Michalski et al.'s AQ15 (1986a; 1986b; Hong, Mozetic, and Michalski, 1986; Wnek et al., 1995), Quinlan's C4.5 (1993), Clark and Niblett's CN2 (Clark and Niblett, 1989; Clark and Boswell, 1991), and Janikow's Genetic-based Inductive Learning system (GIL) (1991; 1993). The comparison is based on rule sets generated by these algorithms for six real world data sets including data sets from medical, botanic, economic, and political domains.

ACKNOWLEDGEMENTS

This work was made possible with the encouragement and support of my advisor Dr. Daniel St. Clair and my other two thesis committee members Dr. Ralph Wilkerson and Dr. Cihan Dagli. I am grateful to the German-American Fulbright Commission for providing me with a scholarship for the Fall Semester 1994 and the Winter Semester 1995 and to the Department of Economics at University of Missouri-Rolla for supporting me with a research assistantship during the Summer Session 1995.

I am thankful to the researchers who allowed me to use their implementations of their inductive learning algorithms. Dr. Ryszard Michalski and Eric Bloedorn from George Mason University, Virginia, allowed me to use their AQ15c program and supported me in its use. Dr. Peter Clark from University of Texas at Austin, Texas, made the revised CN2 learning system available to me and Dr. Cezary Janikow from University of Missouri-St. Louis, Missouri, provided me with the code of his GIL program.

I am also grateful to Dr. David Aha and Patrick Murphy, the past librarians, and Christopher Merz, the current librarian, for compiling and maintaining the UCI Repository of Machine Learning Databases (University of California at Irvine, California). All data sets used for the comparison of inductive learning algorithms in this thesis were obtained from this repository.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Figure

LIST OF TABLES

Table

RULE SET QUALITY MEASURES FOR INDUCTIVE LEARNING ALGORITHMS

Ralf H. Klinkenberg
Computer Science Department
University of Missouri-Rolla
Rolla, Missouri 65401
e-Mail: klinkenberg@ls8.informatik.uni-dortmund.de

Daniel C. St. Clair
University of Missouri-Rolla
Engineering Education Center
St. Louis, Missouri 63121
e-Mail: stclair@umrgec.eec.umr.edu

ABSTRACT:

Symbolic inductive learning systems that induce concept descriptions from examples are valuable tools in the task of knowledge acquisition for expert systems. Since inductive learning methods produce distinct concept descriptions when given identical training data, questions arise as to the quality of the different rule sets produced. This work provides several techniques for comparing and analyzing rule sets. These techniques measure the accuracy, generalization, time and space complexity, and domain coverage of rule sets. Based on these metrics, the performance of four different inductive learning systems is compared. These systems are Michalski et al.'s AQ15 (1986a; 1986b; Hong, Mozetic, and Michalski 1986; Wnek et al., 1995), Quinlan's C4.5 (1993), Clark and Niblett's CN2 (Clark and Niblett, 1989; Clark and Boswell 1991), and Janikow's Genetic-based Inductive Learning system (GIL) (1991; 1993). The comparison is based on rule sets generated by these algorithms for six real world data sets including data sets from medical, botanic, economic, and political domains.

INTRODUCTION

Symbolic inductive learning algorithms learn class descriptions from examples. All information about an example must be expressible in terms of a fixed collection of properties or *attributes*, where the value domain of each attribute is either discrete symbolic, discrete or continuous numeric, or partially ordered. Each example (*instance*) is described by a vector of attribute values and belongs to one of a set of mutually exclusive *classes*. Since the class of each training instance has to be known to the learning algorithm before-hand, this form of learning is called *supervised learning*, as opposed to unsupervised learning in which appropriate groupings of training cases are found by the learning algorithm itself by analyzing and clustering the training data. The induction task is to develop a set of rules that can determine the class of an instance from its attribute values.

Since inductive learning methods generate distinct sets of rules when given identical training data, questions arise concerning the quality of these different rule sets. For most applications, for example, it is desirable to have rule sets which have a high classification accuracy on previously unseen cases and which are small enough to be comprehensible for human experts. For other applications, different criteria may be relevant. The objective of this research is to develop techniques for comparing and analyzing rule sets and to apply these techniques to compare the performance of four inductive learning systems on six real world data sets. These techniques measure the accuracy, generalization, time and space complexity, and domain coverage of rule sets.

The inductive learning systems compared here are Michalski et al.'s AQ15 (1986a; 1986b; Hong, Mozetic, and Michalski, 1986; Wnek et al., 1995), Quinlan's C4.5 (1993), Clark and Niblett's CN2 (Clark and Niblett, 1989; Clark and Boswell, 1991), and Janikow's Genetic-based Inductive Learning system (GIL) (1991; 1993). These algorithms were chosen, because they are well known in the machine learning literature. They generate different types of rule sets, i.e. unordered rule sets, ordered rule sets, also known as decision lists (Rivest, 1987), and decision trees respectively. The rule set quality measures proposed in this work allow the comparison of the rule sets of these different types and thereby make a comparison of a broad class of learning algorithms possible. The variety of these measures allows the

users to base their evaluation of a particular rule set or learning algorithm on the criteria that are most relevant for their application.

DESCRIPTION OF THE DIFFERENT LEARNING ALGORITHMS AND THE TYPES OF RULE SETS THEY GENERATE

This chapter describes the basic concepts of the inductive learning algorithms used for the comparison and the different types of rules sets they produce. As described in further detail below, AQ15 generates either ordered or unordered rule sets, C4.5 produces either decision trees or ordered rules sets, CN2 builds either ordered or un-ordered rule sets, and GIL generates unordered rule sets.

AQ15

The inductive learning system AQ15 was developed by Michalski et al. (1986a; 1986b; Hong, Mozetic, and Michalski 1986). In AQ15, decision rules are represented as expressions in **Variable-valued Logic System 1 (VL_1)** notation, a multi-valued logic propositional calculus with typed variables (Michalski and Larson, 1975). The representations of decision rules in CN2 and GIL are very similar to AQ15's representation and can be viewed as subsets of VL_1. In VL_1, a *selector* relates a variable or attribute to a value or a disjunction of values using one of the relational operators $<$, $\leq$, $=$, $\neq$, $\geq$, or $>$. A conjunction of selectors forms a *complex*. The following complex built of two selectors states that the outlook has to be rainy or cloudy and that the temperature has to be less than 60 degrees for a particular weather condition:

(Outlook = rainy or cloudy) and (Temperature < 60)

For any class, the examples given for that class in the training set are its *positive events* and the examples for all other classes are its *negative events*. A *cover* is a disjunction of complexes describing all positive examples and none of the negative examples of a class. In AQ15, a cover is formed for each class separately and defines the condition part of a corresponding decision rule for that class. The following is an example of a decision rule:

(Outlook = sunny or cloudy) and (Temperature > 60)
or (Windy = true) and (Temperature > 70) => Class (Nice)

The **AQ15** system is based on the AQ algorithm as originally described in (Michalski, 1969; Michalski and McCormick, 1971) and implements the STAR method of inductive learning (Michalski and Larson, 1983). When building a decision rule, AQ performs a heuristic search through a space of logical expressions to determine those that account for all positive and no negative examples. Because there are usually many such *complete* and *consistent* expressions, the goal of AQ is to find the most preferred one, according to flexible extra-logical criteria. These criteria are defined by the user to reflect the needs of the application domain. Figure 1 shows a pseudo-code listing of the AQ15 and STAR algorithms, which form rules from the set of all available selectors.

For each class *C*, the AQ15 algorithm is applied in steps to generate a cover for *C*, each step producing one complex of the cover, until all positive examples are covered. Each step starts with one selected positive example, a *seed*. The STAR algorithm generates a *star*, a

```
Let POS be a set of positive examples of class C.
Let NEG be a set of negative examples of class C.

Procedure  AQ15 (POS, NEG) :          // Find a cover for class C
  Let COVER be the empty cover.
  While COVER does not cover all positive examples in POS :
    Select a SEED, a positive example previously not covered by COVER.
    Let STAR be STAR (SEED, NEG), a set of maximally general complexes that
      cover SEED but no negative example in NEG.
    Let BEST be the best complex in STAR according to the user-defined preference
      criteria.
    Add BEST as an extra disjunct to COVER.
  Return COVER.

Procedure  STAR (SEED, NEG) :         // Find maximally general complexes covering
                                      // SEED but no negative example in NEG
  Let STAR be the set containing the empty complex, which covers the whole domain.
  While any complex in STAR covers some negative examples in NEG :
    Select a negative example E_NEG covered by a complex in STAR.
    Specialize complexes in STAR to exclude E_NEG by :
      Let EXTENSION be the set of all selectors that cover SEED but not E_NEG.
      Let STAR be the set {x ∧ y | x ∈ STAR, y ∈ EXTENSION}.
      Remove all complexes in STAR subsumed by other (more general) complexes.
    While size of STAR > maxstar, the user-defined maximum star size:
      Remove the worst complex from STAR according to the user-defined preference
        criteria.
  Return STAR.
```

Figure 1: The AQ15 And STAR Algorithms (Clark and Niblett, 1989; Wnek et al., 1995).

set of all maximally general complexes, which cover the seed but none of the negative examples, and then selects the best complex from the star according to the user defined criteria. The STAR procedure starts with the entire event space as initial star. As long as a complex in the star covers a negative event, the complexes in the star are specialized to exclude this negative event by adding one or several selectors. If the number of complexes kept in the star exceeds *maxstar*, a user-defined parameter, the star is trimmed according to the user-defined criteria. For this study, the criterion was to "maximize the number of positive examples covered" first and then, in case of a tie, to "minimize the number of selectors."

The AQ15 system can generate unordered and ordered rules. In case of **ordered rules**, the rules for class *n* assume that the rules for the classes *1* to *n-1* are not satisfied. Hence the rule for the last class is a default rule and the rules constructed for the other classes may be less complex than in case of unordered rules, because the rules for class *n* do not need to exclude the instances of the classes *1* to *n-1*. To classify an instance in case of ordered rules, the ordered list of rules is examined to find the first whose complex is satisfied by the instance. The predicted class is then the one nominated by this rule. If no rule's complex is satisfied, the instance is predicted to belong to the default class.

In case of **unordered rules**, each complex is associated with a pair of weights, *t* and *u*, representing the *total* number of events explained by this complex and the number of events explained *uniquely* by this complex, respectively. The *t*-weight may be interpreted as a measure of the representativeness of a complex as a concept description. The complex with the highest *t*-weight may be interpreted as describing the most typical examples of the concept. The complex with the lowest *u*-weight can be viewed as describing rare, exceptional cases. If the learning events from which rules are derived are noisy, such "light" complexes may be indicative of errors in the data.

Two methods of recognizing the concept membership of an instance are distinguished: the *strict* match and the *analogical* match. In the strict match, one tests whether an instance satisfies the condition part of a rule. In the analogical match, one determines the degree of similarity or conceptual closeness between the instance and the condition part. Suppose one has a *t*-weight ordered disjunction of complexes and one removes from it the lightest

complex, i.e. the complex with the smallest *t*-weight. The so truncated description will not strictly match events that uniquely satisfy the truncated complex. However, by applying the analogical match, these events may still come out to be the most similar to the correct concept and thus be correctly classified. A truncated description is of course simpler, but carries a potentially higher risk of recognition error and requires a more sophisticated evaluation. One can proceed further and remove the next "light" complex from the cover and observe the performance. Each such step produces a trade-off between the complexity of the description on one side and the risk factor and the evaluation complexity on the other.

When strictly matching a new instance against a set of unordered rules, three outcomes are possible: there may be only one exact match, *single match*; more than one exact match, *multiple match*; or no exact match. Each category requires a different method of determining the concept of the new event. In case of a single exact match, the event is simply classified according to the matched rule.

If there are several exact matches, a resolution of the conflict becomes necessary and a heuristic called *Estimate of Probability* (EP) is used to predict the class of an event. This estimate is described in Figure 2. If there is no exact match, a heuristic called *Measure of Fit* (MF) as shown in Figure 3 is used. In this case the event belongs to a part of the decision space that is not covered by any decision rule and this calls for analogical matching. The measure of best fit of a class can be interpreted as a combination of "closeness" of the events to a class and an estimate of the prior probability of the class. The AQ15c implementation

Estimate of Probability (*EP*) to predict the class of an event covered by different rules of several classes:
Let $C_1, ..., C_n$ denote decision classes and *e* an event to be classified. For each decision class C_i one has a rule that consists of a disjunction of complexes (*Cpx*), which in turn are conjunctions of selectors (*Sel*).

* *EP* of a complex Cpx_j :
  Ratio of the weight of the complex (the number of learning examples covered by it) by the total number of learning examples, if the complex is satisfied by the event *e*, and 0 otherwise.
* *EP* of a class C_i :
  Probabilistic sum of EPs of its complexes.

The most probable class is the one with the largest EP, i.e. the one whose satisfied complexes cover the largest number of training examples.

Figure 2: Estimate Of Probability (Michalski et al., 1986a; 1986b).

Measure of Fit (*MF*) to predict uncovered events:
* The *MF* of a selector is one, if the selector is satisfied. Otherwise, this measure is proportional to the amount of the decision space covered by the selector (number of disjunctively linked attribute values in the selector divided by the total number of the attribute's possible values).
* The *MF* of a complex is defined of the product of *MFs* for a conjunction of its constituent selectors, weighted by the proportion of learning examples covered by the complex.
* The *MF* of a class is obtained as a probabilistic sum for a disjunction of complexes.

Figure 3: Measure Of Fit (Michalski et al., 1986a; 1986b).

(Wnek et al., 1995) used for this study can only handle discrete or discretized attribute value domains with finitely many attribute values and treats continuous numeric attribute values as discrete values.

C4.5

Quinlan's inductive learning system C4.5 (Quinlan, 1993) evolved from the top-down decision tree induction program ID3 (Quinlan, 1986). Like ID3, C4.5 generates a classifier in the form of a *decision tree*, a structure that is either a *leaf*, indicating a class, or a *decision node* that specifies some test to be carried out on a single attribute value, with one branch and subtree for each possible outcome of the test. The test at a decision node is very similar to a selector in a decision rule as described above, but may have more than two outcomes. Figure II.4 shows a decision tree with a continuous numeric attribute (A_1) and several discrete symbolic attributes (A_2, A_3, A_4) for the tests. The non-leaf nodes of the decision tree are labeled with the names of the attributes whose values are tested at these nodes. The branches starting at an inner node are labeled with the possible outcomes of such a test. The leaf nodes are labeled with the predicted class.

A decision tree can be used to classify a case by starting at the root of the tree and moving through it until a leaf is encountered. At each non-leaf node, the case's outcome for the test at this node is determined and attention shifts to the root of the subtree corresponding to the outcome of this test. When this process finally leads to a leaf, the class of the case is predicted to be that recorded at the leaf.

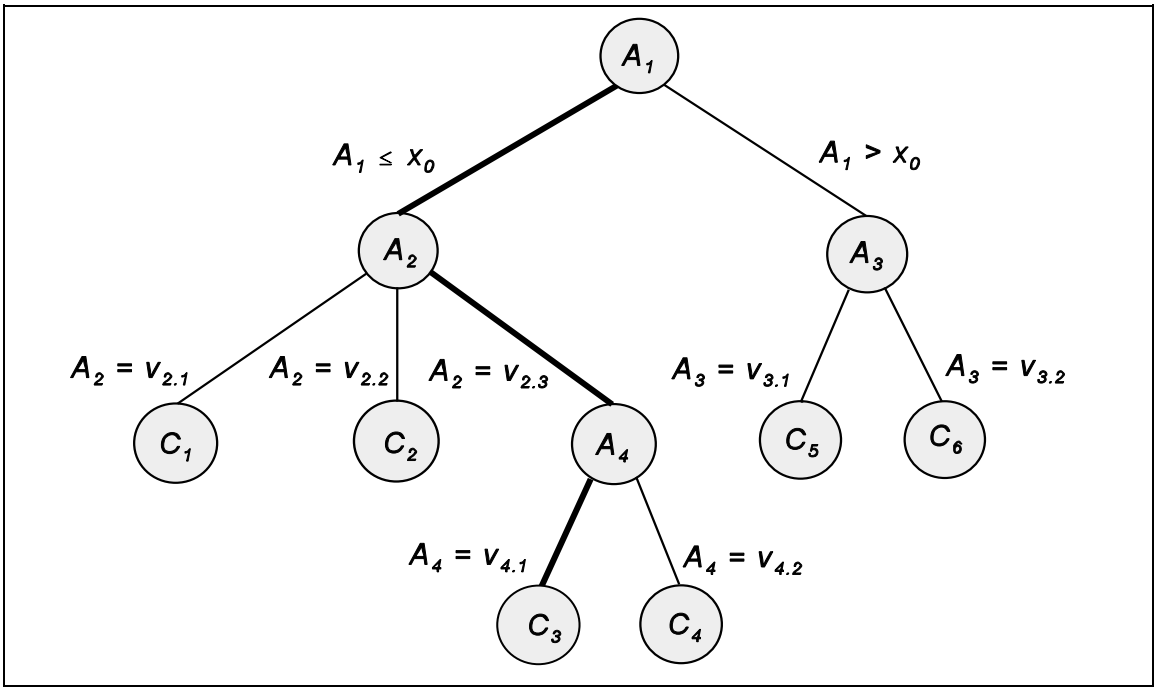

Figure 4: Example Of A Decision Tree.

Let T be a set of training cases and let the classes be denoted $\{C_1, C_2, ..., C_k\}$.

Procedure **BuildTree** (T):

There are three possible situations at a node where T is given during the tree building process:

* T contains one or more cases, all belonging to a single class C_j:
  The decision tree for T is a leaf identifying class C_j.
* T contains no cases:
  The decision tree is again a leaf, but the class to be associated with the leaf most be determined from information other than T. C4.5 uses the most frequent class at the parent of this node to label this leaf node.
* T contains cases that belong to a mixture of classes:
  According to the gain ratio criterion, a test based on a single attribute is chosen that has one or more mutually exclusive outcomes $\{O_1, O_2, ..., O_n\}$. T is partitioned into subsets $T_1, T_2, ..., T_n$, where T_i contains all the cases in T that have outcome O_i of the chosen test. The decision tree for T consists of a decision node identifying the test, and one branch for each possible outcome. The same tree-building method is applied recursively to each subset of training cases, so that the I-th branch leads to the decision tree constructed from the subset T_i of training cases.

Figure 5: Skeleton Of The Decision Tree Building Method Employed By C4.5 (Quinlan, 1993).

The skeleton of the divide and conquer method employed by C4.5 to build a decision tree from a set T of training cases is shown in Figure 5. While ID3 used the *information gain criterion* to select the best attribute at a particular node to keep the decision tree as shallow as possible, C4.5 employs the *gain ratio criterion*, because the information gain criterion has a strong bias in favor of attribute tests with many outcomes. Suppose a possible test X with n outcomes that partitions the set of training cases into subsets T_1, T_2, ..., T_n is given, let $freq(C_i,T)$ stand for the number of cases in a set T that belong to class C_i, and let $|T|$ denote the number of cases in the set T. As explained in Quinlan (1993), the average amount of *information* needed to identify the class of a case in T is

$$Info(\ T\ ) = -\sum_{j=1}^{k} \frac{freq(\ C_j\ ,\ T\ )}{|\ T\ |} \cdot \log_2 \frac{freq(\ C_j\ ,\ T\ )}{|\ T\ |} \qquad \text{bits.}$$

This quantity is also known as *entropy* of the set T. After T has been partitioned in accordance with the n outcomes of a test X, the expected information requirement is

$$Info_X\ (\ T\ ) = \sum_{i=1}^{n} \frac{T_i}{|\ T\ |} \cdot Info(\ T_i\ )$$

and hence the information gained by partitioning in accordance with the test X is measured by

$$Gain(\ X\ ) = Info(\ T\ ) - Info_X\ (\ T\ )$$

The gain criterion selects a test to maximize this *information gain* which is also known as *mutual information* between the test X and the class.

The information content of a message pertaining to a case that indicates not the class but the outcome of the test is defined as

$$SplitInfo(\ X\ ) = -\sum_{i=1}^{n} \frac{|\ T_i\ |}{|\ T\ |} \cdot \log_2 \frac{|\ T_i\ |}{|\ T\ |}$$

This represents the potential information generated by dividing *T* into *n* subsets, whereas the information gain measures the information relevant to classification that arises from the same division. The proportion of information generated by the split that is useful, i.e. that appears helpful for classification is computed as

$$GainRatio(X) = \frac{Gain(X)}{SplitInfo(X)}$$

The gain ratio criterion selects a test to maximize the ratio above, subject to the constraint that the information gain must be at least as large as the average gain over all tests examined.

After generating a decision tree as described above, C4.5 uses a *tree pruning* method to simplify the tree by discarding one or more subtrees and replacing them with leaves or one of their branches. Assuming that it is possible to predict the error rate of a tree and its subtrees (including leaves), the following pruning approach can be applied. Start from the root node of the tree and examine each non-leaf subtree. If replacement of this subtree with a leaf or with its most frequently used branch leads to a lower predicted error rate, then prune the tree accordingly. Since the error rate for the whole tree decreases as the error rate of any of its subtrees is reduced, this process will lead to a tree whose predicted error rate is minimal with respect to the allowable forms of pruning.

The C4.5 system employs the following pessimistic estimate to predict the error. When *N* training cases are covered by a leaf, *E* of them incorrectly, the *resubstitution error* rate for this leaf is *E/N*. Naively regarding this as observing *E* events in *N* trials and regarding the *N* training cases as a sample, which it is not, the probability of such an event (resubstitution error) over the entire population of cases is estimated using the posterior distribution of this probability usually summarized by a pair of confidence limits. For a user-defined confidence level *CF*, the upper limit on this probability can be found from the confidence limits for the binomial distribution, here denoted by $U_{CF}(E,N)$. Appendix A shows how this confidence interval can be computed. The C4.5 system simply takes this upper limit as the predicted error rate at a leaf. To simplify the accounting, error estimates for leaves and subtrees are computed assuming that they were used to classify a set of unseen cases of the same size as the training set. So, a leaf covering N training cases with a predicted error rate of $U_{CF}(E,N)$

Procedure **TransformTreeToRules** (*T*):
* Every path from the root of the unpruned tree *T* to a leaf gives one initial rule.
* Each such rule is simplified by removing conditions that do not seem helpful for discriminating the nominated class from other classes, using a pessimistic estimate of the accuracy of the rule.
* For each class in turn, all the simplified rules for that class are sifted to remove rules that do not contribute to the accuracy of the set of rules as a whole.
* The sets of rules for the classes are then ordered to minimize the number of false positive errors, that is the number of cases falsely classified to belong to a particular class by the set of rules for that class.
* Finally, the majority class in the set of cases covered by no rule is chosen as default class.

Figure 6: Steps In The Transformation Of A Decision Tree Into An Ordered Rule List.

would give rise to a predicted $N \cdot U_{CF}(E,N)$ errors. Similarly, the number of predicted errors associated with a tree or a subtree is just the sum of the predicted errors of its branches.

The C4.5 inductive learning system can also transform the generated decision tree to a *set of ordered rules* including a *default rule*. Figure 6 shows a summary of the steps in the rule transformation process. For the transformation to a rule set, every path from the root of the unpruned tree to a leaf gives one initial rule, whose left-hand side is the conjunction (*complex*) of all attribute-based tests (*selectors*) established by the path and whose right-hand side specifies the class predicted at the leaf. The highlighted path from the root of the decision tree in Figure II.4 to the leaf node labeled class C_3, for example, can be interpreted as the rule

$$(A_1 \leq x_0) \text{ and } (A_2 = v_{2.3}) \text{ and } (A_4 = v_{4.1}) \quad \Rightarrow \text{ Class } (C_3)$$

If the path to each leaf node was transformed into a production rule, the resulting collection of rules would classify cases exactly as the tree and, as a consequence of their tree origin, the rules would be mutually exclusive and hence their order would not matter. Antecedents of individual rules may contain irrelevant conditions (*selectors*). Let rule *R* be of the form "*A* => Class (*C*)" and a more general rule R_0 "A_0 => Class (*C*)", where A_0 is obtained from *A* by deleting one selector *X* from the complex *A*. Each training case that satisfies the shorter antecedent A_0 either does or does not belong to class *C* and either does or does not satisfy selector *X*. The number of cases in each of these four groups can be organized into a 2 x 2 *contingency table*:

	Class C	Other Classes
Satisfies selector X	Y_1	E_1
Does not satisfy selector X	Y_2	E_2

The C4.5 system predicts the error rate of rules using the same pessimistic error estimate already employed for decision tree pruning. The estimate of the error rate of rule R can be set to $U_{CF}(E_1, Y_1+E_1)$ and that of rule R_0 to $U_{CF} (E_1+E_2, Y_1+Y_2+E_1+E_2)$. If the pessimistic error rate of rule R_0 is not greater than that of the original rule R, the selector X can be deleted.

While one or more selectors in a rule can be deleted as above, that selector is removed that produces the lowest pessimistic error rate of the generalized rule and the error rates for the other conditions are recalculated. This is done until no more selectors can be deleted from the rule. The *rule generalization process* is repeated for each path in the unsimplified decision tree. The resulting rules are no longer exhaustive and mutually exclusive. Therefore a simple form of *conflict resolution* is implemented: The rules are ordered and a *default rule* is introduced for otherwise uncovered events. First, all rules for each single class are grouped together and the rules within these rule subsets are ordered according to the Minimum Description Length (MDL) principle (Rissanen, 1983). The MDL principle states that the best theory derivable from the training data will minimize the number of bits required to encode the total message consisting of the classification theory (rule set) together with its associated exceptions. For each class in turn, C4.5 considers all possible subsets of simplified rules for this class, if there are not too many of them, and uses simulated annealing to find a good subset otherwise. Thereby rules that do not contribute to the accuracy of the set of rules as a whole are removed. The sets of rules for the classes are then ordered to minimize the number of false positive errors, that is the number of cases falsely classified to belong to the particular class by its set of rules. This is done by ordering the rule sets by increasing number of their false positive errors. As default class for uncovered events the majority class in the set of uncovered training cases is chosen. Finally, if there are one or more rules whose omission would actually reduce the number of classification errors of the total rule set on the training cases, the first such rule is discarded and the set checked again.

CN2

The inductive learning system CN2 was developed by Clark and Niblett (1987; 1989) and later modified by Clark and Boswell (1991). The objective behind the design of CN2 was to modify the AQ algorithm by retaining its beam search through the space of complexes, but removing its dependency on specific training instances during search. While the AQ algorithm searches only the space of complexes that are completely consistent with the training data, CN2 extends its search space to rules that do not perform perfectly on the training data by broadening the specialization process to examine all specializations of a complex, in much the same way that ID3 and C4.5 respectively consider all attribute tests when growing a node in a tree. A cutoff method similar to decision tree pruning is applied to halt specialization when no further specializations are statistically significant. The modified version of CN2 produces either an *ordered set of if-then rules* of the form "If <complex> then predict <class>" like the original CN2 version or an *unordered set of if-then rules*, both including a default rule. The <complex> in the condition part of the rule is a VL_1 complex.

The control procedure of the **CN2 algorithm for ordered rules**, shown in Figure 7, iteratively calls the beam search procedure, shown in Figure 8, to find the best complex until no more best complexes are found and appends a rule to the rule set with this best complex as condition and the most common class among the instances covered by this complex as

```
Procedure  CN2_ordered (EXAMPLES, CLASSES):        // Control procedure for
                                                   // ordered rules
  Let RULE_LIST be the empty list.
  Repeat
    Let BEST_COMPLEX be FindBestComplex (EXAMPLES).
    If  BEST_COMPLEX is not null
    then  let CLASS be the most common class of examples covered by
              BEST_COMPLEX,
          add rule  "If  BEST_COMPLEX  then  predict CLASS"  to the end of
              RULE_LIST, and
          remove from EXAMPLES all examples covered by BEST_COMPLEX.
  Until BEST_COMPLEX is null.
  If there are any examples left in EXAMPLES
  then  let CLASS be the most common class in EXAMPLES and
        add the default rule  "Predict CLASS"  to the end of RULE_LIST.
  Return RULE_LIST.
```

Figure 7: The CN2 Ordered Rules Algorithm (Clark and Boswell, 1991).

```
Procedure  FindBestComplex (EXAMPLES [, CLASS][1]):        // Find the best complex
                                                           // (search procedure)
  Let MGC be the most general complex (= "true").
  Let STAR be the set of  containing only MGC (= {MGC}).
  Let NEW_STAR be the empty set (= {}).
  Let BEST_COMPLEX be null.
  While STAR is not empty:
    For each COMPLEX in STAR:
      For each possible attribute test TEST not already tested on in COMPLEX:
        Let NEW_COMPLEX be a specialization of COMPLEX, formed by adding
          TEST as an extra conjunction to  COMPLEX
          (i.e. NEW_COMPLEX = COMPLEX & TEST).
        If  NEW_COMPLEX is better than BEST_COMPLEX
          and  NEW_COMPLEX is statistically significant
        then  let BEST_COMPLEX be COMPLEX
                  (i.e. BEST_COMPLEX = COMPLEX).
        Add NEW_COMPLEX to NEW_STAR.
        If  size of  NEW_STAR > maxstar  (a user-defined constant)
        then  remove the worst complex in NEW_STAR.
    Let STAR = NEW_STAR.
  Return BEST_COMPLEX.
```

[1] CLASS is only required for generating unordered rules.

Figure 8: The CN2 Rule Search Algorithm (Clark and Boswell, 1991).

```
Procedure  CN2_unordered (ALL_EXAMPLES, CLASSES):      // Control procedure
                                                       // for unordered rules
  Let RULE_SET be the empty set.
  For each CLASS in CLASSES:
    Let RULES be CN2_ForOneClass (ALL_EXAMPLES, CLASS).
    Add RULES to RULE_SET
  Return RULE_SET.

Procedure  CN2_ForOneClass (EXAMPLES, CLASS):          // Control procedure
                                                       // for one class
  Let RULES be the empty set.
  Repeat
    Let BEST_COMPLEX be FindBestComplex (EXAMPLES, CLASS).
    If  BEST_COMPLEX is not null
    then  add the rule  "If  BEST_COMPLEX  then  predict CLASS"  to RULES
          and remove from EXAMPLES all examples in CLASS covered by
                BEST_COMPLEX.
  Until BEST_COMPLEX is null.
  Return RULES.
```

Figure 9: The CN2 Unordered Rules Algorithm (Clark and Boswell, 1991).

prediction. The instances covered by a rule are removed from the training set. The last rule in CN2's rule list is a default rule predicting the most common class among the uncovered training examples.

The **beam search procedure** to find the best complex corresponds to the STAR procedure of the AQ algorithm. The pruned general-to-specific search retains a size-limited set or star of "best complexes found so far." The system examines only specializations of this set, carrying out a beam search of the space of complexes. A complex is specialized by either adding a new selector to the conjunction or by removing a disjunctive element in one of its selectors. To use the induced ordered rules to classify a new example, CN2 tries each rule in order until one is found whose conditions are satisfied by the example and assigns the class predicted by this rule to the example.

The control procedure for the **CN2 algorithm for unordered rules** is shown in Figure 9. The CN2 algorithm can be easily modified to generate an unordered rule set by changing only the control procedure and leaving the beam search procedure apart from its ranking function for complexes unchanged. The main modification to the algorithm is to iterate the search for each class in turn, removing only covered examples of the current class when a rule has been found. Unlike for ordered rules, the negative examples remain because now each rule must independently stand against all negatives. The covered positives must be removed to stop CN2 from repeatedly finding the same rule. To effect this rule search for each class in turn, the heuristic must be applied differently: with ordered rules the predicted class is simply taken as the one with the most covered examples in it, but with unordered rules the predicted class is fixed to be the class selected to be in turn by the revised control procedure (parameter CLASS of FindBestComplex in Figure 8). To apply unordered rules to classify new examples, all rules are tried and those whose conditions were all satisfied are collected. If a clash occurs, i.e. more than one class is predicted by the collected rules, a probabilistic method is employed to resolve the clash. Each rule is tagged with the distribution of covered examples among classes and these distributions are summed to find the most probable class.

Two **heuristic decisions** must be made in FindBestComplex during the learning process. First a *ranking function for the complexes* is needed to determine wether a new complex should replace the currently best complex and which complex has to be removed if the

maximum star size is exceeded. Like ID3, the original CN2 version used the information-theoretic *entropy* measure (Kalbfleish, 1979) for this purpose (the lower the entropy the better the complex). This function prefers complexes covering a large number of examples of a single class and few examples of other classes, but it tends to select very specific rules covering only a few training examples. The modified version of CN2 employs the *Laplacian error estimate* (Clark and Boswell, 1991) instead. The expected accuracy, one minus the Laplace expected error estimate, is given by the formula:

$$LaplaceAccuracy(\ n_{class}\ ,\ n_{covered}\ ,\ n\ ) = \frac{n_{class} + 1}{n_{covered} + n}$$

where n is the number of classes, n_{class} is the number of examples in the predicted class covered by the rule, and $n_{covered}$ is the total number of examples covered by the rule. This estimate avoids the downward bias of the entropy measure of favoring very specific complexes in the general-to-specific search. A final check is included to ensure the expected accuracy is at least better than that of a default rule predicting the class for all examples.

The second evaluation function tests wether a complex is statistically *significant*, i.e. wether it locates a regularity unlikely to have occurred by chance and thus reflects a genuine correlation between attribute values and classes in the training data. To test significance, CN2 uses the *likelihood ratio statistic* (Kalbfleish, 1979). This is given by:

$$LikelihoodRatio(\ F,\ E\ ) = 2 \cdot \sum_{i=1}^{n} f_i \cdot \log \frac{f_i}{e_i}$$

where the distribution $F = (\ f_1, ..., f_n\ )$ is the observed frequency distribution of examples among classes satisfying a given complex and $E = (\ e_1, ..., e_n\ )$ is the expected frequency distribution of the same number of examples under the assumption that the complex selects examples randomly from the training set. Under suitable assumptions, one can show that this statistic is distributed approximately as χ^2 (chi-squared) with *n-1* degrees of freedom. Thus the two functions Laplacian error estimate and significance serve to determine whether complexes found during the search are both “good” (have high accuracy when predicting the

majority class covered) and "reliable" (the high accuracy on the training data is not just due to chance). The CN2 algorithm uses these two functions repeatedly during search for the "best" complex that also passes some minimum treshold of reliability until no more reliable complexes can be found.

GIL

Janikow's Genetic-Based Inductive Learning system (GIL) is a knowledge-intensive genetic algorithm approach for supervised learning (Janikow, 1991; 1993). *Genetic algorithms (GAs)* are adaptive methods of searching a solution space by applying operators modeled after the natural genetic inheritance and simulating the Darwinian struggle for survival of the fittest. In general, a genetic algorithm (GA) performs a multi-directional search and encourages information formation and exchange between such directions. It does so by maintaining a population of proposed solutions for a given problem (*chromosomes*). The population undergoes a *simulated evolution*: relatively "good" solutions are more likely to produce offsprings, which subsequently replace the "worse" ones. The estimate of the quality of a solution is based on an *evaluation function*, which plays the role of an environment. Each iteration, called a *reproduction cycle*, is performed in three steps (see Figure 10). During the *selection step* a new population is formed from stochastically best samples (with replacement). Then, during the *recombination step*, some of the members of the newly selected population are altered. Finally, in the third step, all such altered individuals are evaluated. The recombination is based on the application of two operators, *mutation* and

Variable naming: Let *t* be the current point in time and *P(t)* population at time *t*.

Procedure **Genetic Algorithm**:
  Let *t = 0*.
  Initialize *P(t)*.
  Evaluate *P(t)*.
  While (not Termination-Condition(*P(t)*)):
    Let *t = t + 1*.
    Select *P(t)* from *P(t-1)*.
    Recombine *P(t)*.
    Evaluate *P(t)*.
  Return *P(t)*.

Figure 10: The Core Of A Genetic Algorithm (Janikow, 1991).

crossover. Mutation introduces random variability into the population, and crossover exchanges random pieces of two chromosomes in the hope of propagating a partial solution. Since both of these operators are usually defined on syntactic pieces of the underlying representation, for example on bits in case of a binary representation, the search has domain-independent properties. However, the applicability of a GA to a particular problem depends on the representation emphasizing meaningful semantic pieces of information, called *building blocks*, to be used by the crossover operator, and on the evaluation function to properly guide the search.

The most praised characteristic of a classical GA, its domain-independent search, is at the same time its problem, because this may make the algorithm unmanageably complex in some domains. The search is unguided and does not make use of any knowledge about the domain or its structure that could make the search more efficient. To overcome this problem, GIL incorporates *task specific knowledge* into the algorithm to guide the search and provide for a faster convergence to a desired solution. The traditional domain-independent operators are replaced by special forms of mutation and crossover that implement the specific problem solving methodology of inductive learning.

The chromosomes are represented in VL_1, but GIL only considers VL_1 formulas using the equality relation, '=', and internal disjunctions of attribute values. If all attributes only take a finite number of symbolic or numeric values, this is no lack of generality, because selectors using other relational operators can be rewritten using the equality operator. For continuous numeric attributes, however, this imposes a significant restriction, since a finite number of intervals or discretized values has to be introduced to represent all possible values. For further simplification, it is assumed that only a single concept description has to be learned.

GIL's *genetic operators* are based on Michalski's description of various inductive operators that constitute the process of inductive inference (Michalski, 1983). According to the three syntactic levels of the rule-based framework (conditions, rules, rule sets), the operators can be divided into three corresponding groups. In addition, each operator is classified as having either generalizing, specializing, or unspecified or independent behavior. Figures 11 to 13 list the genetic operators for the three syntactic levels along with short descriptions.

Genetic Operators on Rule Set Level (VL_1 set of complexes), operators act on whole rule sets:

* *Rules exchange* (independent operator):
  This operator exchanges random rules between two parent rule sets.
* *Rules copy* (generalization operator):
  This operator requires two parent rule sets and copies a random rule from each of the sets to the other.
* *New event* (generalization operator):
  If there is a positive event not covered yet by the current rule set, this event's description is added to the set as a new rule.
* *Rules generalization* (generalization operator):
  This operator acts on a single rules set taking two random rules and replacing them by their most specific generalization, which is not necessarily consistent with respect to previously excluded negative events.
* *Rules drop* (specialization operator):
  This operator drops a random rule from the current set.
* *Rules specialization* (specialization operator):
  This operator replaces two random rules from the current rule set by their most general specialization.

Figure 11: Genetic Operators On Rule Set Level (Janikow, 1991).

Genetic Operators on Rule Level (VL_1 complex), operators act on rules:

* *Rule split* (independent operator):
  This operator splits a single rule into a number of rules, according to a partition of values of a condition, according to each value individually (nominal data types) or according to two disjoint subsets of values by cutting the ordered set of values at a random place (linear data types).
* *Condition drop* (generalization operator):
  This operator removes a present condition from a single rule.
* *Turning conjunction into disjunction* (generalization operator):
  This operator splits the complex of the current rule into a disjunction, where the complex's separation into *n* and *m* selectors is random and position independent.
* *Condition introduce* (specialization operator):
  This operator introduces a random condition with an unconditioned attribute to the current rule. The new selector is a random choice from among all of its possible internal disjunctions.
* *Rule directed split* (specialization operator):
  This operator takes a single rule. If this rule covers a negative event, it is split into a set of maximally general rules that are still consistent with that event, i.e. that do not cover that event anymore.

Figure 12: Genetic Operators On Rule Level (Janikow, 1991).

Genetic Operators on Condition Level (VL_1 selector), operators act on conditions:

* *Reference change* (independent operator):
  This operator acts on a single condition (selector) randomly removing or adding a single domain value to this condition.
* *Reference extension* (generalization operator):
  This operator extends the domain of a single condition by allowing a number of additional random values (nominal data type) or by selecting a single value or closing a range between two current values (linear data type).
* *Reference restriction* (specialization operator):
  This operator removes some domain values from a single condition and implements the opposite action of the reference extension operator.

Figure 13: Genetic Operators On Condition Level (Janikow, 1991).

The *evaluation function* must reflect the learning criteria, here *completeness*, *consistency*, and possibly *complexity*. Completeness and consistency are defined as:

$$Completeness = \frac{number\ of\ positive\ training\ events\ covered\ by\ a\ rule\ set}{total\ number\ of\ positive\ training\ events}$$

$$Consistency = 1 - \frac{number\ of\ negative\ training\ events\ covered\ by\ a\ rule\ set}{total\ number\ of\ negative\ training\ events}$$

The completeness and consistency measures are replaced with a single measure of *correctness* using two weights w_1 and w_2 related to the relative frequency of positive and negative examples in the training set:

$$Correctness = \frac{w_1 \cdot Completeness + w_2 \cdot Consistency}{w_1 + w_2}$$

The *cost* of a description is measured by its *complexity*, which combines the number of rules and conditions in the following way:

$$Complexity = 2 \cdot number\ of\ rules + number\ of\ conditions$$

Finally, correctness and cost are combined to the single evaluation function:

$$Evaluation = Correctness \cdot (1 + w_3 \cdot (1 - Complexity))^f$$

where w_3 determines the influence of complexity (which itself is normalized on [0,1]), and f grows very slowly on [0,1] as the population ages. For the experiments described later, a very low w_3 weight was used. A dynamic approach is used to adjust the effect of the cost as the population ages. The effect of a very slowly raising f is that initially the influence of the cost is very light in order to promote deeper space exploration, and only increases at later stages in order to minimize the descriptions' complexity.

The individuals in the *initial population* are sets of a random number of randomly generated complexes or random positive training events. Each operator is given some *initial probabilities* from two separate groups: application probabilities and selection probabilities. *Application probabilities* describe how likely it is that the individual operators are applied to their corresponding structures (rule sets, rules, conditions) and have a dynamic character with respect to the current context, to both the current coverage and the current, problem-dependent size of the average chromosome. The a priori probabilities of generalizing operators are increased for applications to structures that are incomplete, and decreased for those inconsistent. On the other hand, the a priori probabilities of specializing operators are increased for applications to structures that are inconsistent, and decreased for those that are incomplete. The measures of inconsistency /incompleteness serve as additional heuristic guiding the selection of appropriate operators (in addition to fitness). The application probabilities are adjusted linearly to those measures according to the following formulas

$$\textit{Generalizing operators}: \quad p' = p \cdot (\frac{3}{2} - \textit{Completeness}) \cdot (\frac{1}{2} + \textit{Consistency})$$

$$\textit{Specializing operators}: \quad p' = p \cdot (\frac{1}{2} + \textit{Completeness}) \cdot (\frac{3}{2} - \textit{Consistency})$$

where the new value p' is the adjusted probability, and p is the actual probability. The value of p' is computed for each chromosome separately and does not replace the a priori p. *Selection probabilities* serve as a mean of selecting one of a number of possible actions or substructures to participate in the operation to be applied. They are static.

The **GIL algorithm** uses the above components within the genetic algorithm framework as shown in Figure 10. At each iteration all rule sets of the population are evaluated, and a

new population is formed by drawing members form the original population in such a way that more fit individuals have a higher chance of being selected. Then the operators are applied to the new population in order to move these partial solutions closer to a desired solution. Each operator acts on structures from its level, applying itself to some randomly selected structures. The application depends on the initial probabilities, the consistency /completeness of such structures, and the size of the currently average chromosome. The cycle repeats until a desired description is found or some resources are exhausted.

RULE SET QUALITY MEASURES

This chapter proposes several metrics for comparing rule sets generated by inductive learning algorithms. These rule set quality measures include metrics for the accuracy, generalization, time and space complexity, and domain coverage of rule sets.

Predictive Accuracy

The most widely used rule set quality measure is the predictive accuracy of a rule set on a set of previously unseen instances. The *overall accuracy* of a rule set is the number of correctly classified test instances divided by the total number of test instances. Instances not covered by the rule set are not counted as correctly classified but are nevertheless counted for the total number of instances:

$$Overall\ accuracy = \frac{number\ of\ correctly\ classified\ test\ instances}{number\ of\ test\ instances}$$

Often not only the overall accuracy of a rule set is of interest, but the *accuracies for the prediction of the distinct classes* as well. Sometimes it is desirable to have a similar accuracy on instances of all classes, while in other cases misclassifications of instances of a certain class are more costly than misclassifications of instances of another class. An example for such a case is the classification of patients into those who need medication and those who do not. In case of a life threatening illness it is often more serious not to treat a patient that should be treated than treating a patient who is not sick. The accuracy of a rule set for a particular

class is the fraction of the instances of that class classified correctly among all instances of that class in the test set:

$$Accuracy\ for\ class\ C = \frac{number\ of\ correctly\ classified\ test\ instances\ of\ class\ C}{number\ of\ test\ instances\ of\ class\ C}$$

While ordered rules and decision trees allow to classify each covered instance unambiguously, several rules of an unordered rule set predicting distinct classes may cover an instance at the same time. The question of which class to predict for this instance has to be resolved. For the following experiments, the same instance classification procedure was applied to all rule sets. Let R denote the number of rules in an unordered set of rules. Each unordered rule I was associated with a weight w_i representing the number of training instances correctly classified by this rule. If a test instance X was matched by several unordered rules predicting different classes, the weights of these rules were added for each class and the class with the highest count was predicted:

$$Predicted\ class(X) = C_j \quad with \quad n_{C_j} = \max\{\, n_{C_k} \mid n_{C_k} = \sum_{i=1}^{R} P\,(C_k\,,\,X) \cdot w_i \,\}$$

$$where \quad P\,(C_k\,,\,X) = 1 \quad , if \ \ rule\ i\ covers\ X \ \wedge \ \ rule\ i\ predicts\ class\ C_j\ , \ and$$
$$P\,(C_k\,,\,X) = 0 \quad , otherwise .$$

The results of this classification heuristic do not always correspond to the heuristic of the learning algorithms, but the use of this uniform heuristic for the rule sets of all algorithms allowed a better comparison of the rule sets independent of their algorithms. The accuracies obtained by the learning algorithms using their own heuristics are given in the description of the experimental results as well, but only as additional information.

Let I denote the number of instances in the test set, let R denote the number of rules in an ordered or unordered rule set, and let S_R denote the number of selectors of the rule with the most selectors in that rule set, then the predictive accuracies of the rule set can be computed in $O(I * R * S_R)$ time. If S_P denotes the number of selectors in the longest path of a decision tree, $O(I * S_P)$ is an upper bound for the computation of the accuracies of this tree.

Generalization

The *generalization* measure was previously used by Gallion et al. (1993). The generalization of a rule set describes how well the rule set generalizes from the set of training instances and is defined as

$$Generalization = 1 - \frac{number\ of\ rules}{number\ of\ training\ instances}$$

Thinking of the original training data as a set of rules, the generalization of a rule set produced by a learning algorithm presents the reduction produced by the algorithm as a percentage. Using the same notation as above, this metric can be computed in $O(I + R)$ time.

Space Complexity

For many real world applications, the number of training instances available is very large and the space complexity of a rule set becomes a further relevant aspect as it is necessary to store the rule set and as it is desirable to have rule sets that are small enough to be intelligible for humans. The space complexity measures considered here are the *number of selectors* in a rule set, the *number of complexes* in a rule set, and the *number of rules* in a rule. The number of rules includes the default rule in case the rule set has such a rule.

In case of a decision tree, the number of leaf nodes corresponds to the number of rules and the attribute tests at the non-leaf nodes correspond to selectors, although for decision trees these selectors can be more complex than in the case of a rule set. Attribute tests with two outcomes directly correspond to selectors, while attribute tests with $n>2$ outcomes are transformed into a chain of $n-1$ selectors in such a way that the first selector represents the first outcome of the test, the second selector the second, and so on, so that each non-leaf node in the tree becomes a binary decision node. Figure 14 shows the ordered rule set obtained from the decision tree shown in Figure 4 after it has been transformed as described.

The attribute test A_2 in Figure 4, for example, is transformed into the two selector nodes $(A_2 = v_{2.1})$ and $(A_2 = v_{2.2})$. Going down the transformed tree from the root node A_1 along the highlighted path to the leaf node C_3 to classify an instance that has the value $v_{2.3}$ for attribute

```
If (A1 ≤ x0) then
        if (A2 = v2.1) then       predict class C1
        else if (A2 = v2.2) then  predict class C2
        else
                if (A4 = v4.1) then predict class C3
                else                predict class C4
else
        if (A3 = v3.1) then predict class C5
        else                predict class C6
```

Figure 14: Ordered Rule Set Obtained From The Decision Tree In Figure 4.

A_2, first the selector $(A_2 = v_{2.1})$ is tested and found to be not satisfied, hence the second selector $(A_2 = v_{2.2})$ is tested and found to be not satisfied either, so that the classification procedure then follows the remaining branch for $A_2 = v_{2.3}$. Each attribute tests with n outcomes counts as $n-1$ selectors. The number of selectors in a transformed decision tree is the sum of the number of selectors for each of its attribute tests. Since there is no real complex involved along a path in a tree, the number of complexes in a tree is set to the number of its paths, i.e. the number of its leaf nodes.

If R denotes the number of rules in the set, C the number of complexes, and S the number of selectors, then the space complexity measures of the rule set can be computed in $O(R + C + S)$ time. Sometimes, like for example for the internal rule set quality evaluation in GIL (Janikow, 1991), the number of complexes and the number of rules are combined into one complexity measure. Since the amount of memory or disk space needed to store a rule set and the correlation of this need to the space complexity measures introduced in this work depend on the particular implementation, combined measures like weighted sums of the space complexity measures have not been used for this study. For some implementations, the amount of memory or disk space needed may simply be proportional to the number of selectors and independent of the number of complexes. It is sensible to let the users define combined complexity measures according to their applications, if such measures are needed at all. Based on the metrics presented here, this can be done easily.

Time Complexity

For some applications, it is relevant how long it takes to classify previously unseen instances using a particular rule set. The time complexity measures proposed here count the *average number of selector tests* and the *average number of complex tests needed to classify a previously unseen instance.*

In an unordered rule set, all rules always have to be evaluated to determine which are satisfied by an instance to be classified and hence all complexes are counted for the number of complex tests. In an ordered rule set, all rules after the first rule that matches an instance do not have to be evaluated and hence are not counted for the number of complex tests and selector tests. If an instance satisfies a complex it is tested against, all selectors of this complex are counted, since they all have to be evaluated for the instance. If an instance does not satisfy such a complex, the selectors of this complex after the first unsatisfied selector are not counted, since they do not need to be evaluated.

In case of a decision tree, the non-leaf nodes with more than two outcomes are transformed into a chain of several binary decision nodes as described above. The number of selectors needed to classify a particular test instance is the number of selector tests along the path in tree followed to classify this instance. The computational complexity of the time complexity measures is equivalent to the computational complexity of the predictive accuracy, because, to compute these measures, all instances have to be classified and the number of tested selectors and complexes has to be counted.

Domain Coverage

The *domain coverage* of a rule set is defined as the percentage of the instances in the domain it covers, no matter whether the instances are correctly classified or not. This metric is computed as the number of test instances covered by the rule set divided by the total number of test instances. Like for the computation of the predictive accuracy and the time complexity measures, it is assumed that the test instances are representative for the particular domain, because otherwise the values computed for these measures would not be predictive for the performance of the rule set on previously unseen instances.

If a rule set has a low coverage, there are many instances whose class cannot be predicted by the rule set and hence the overall quality of this rule set cannot be very high. A low coverage automatically results in a low predictive accuracy, because uncovered instances are counted as not correctly classified. Therefore a high coverage is desirable, although it is no guarantee of a high accuracy. Rule sets including a default rule and decision trees always have a coverage of 100%, because they always cover all instances, but they do not necessarily have a high predictive accuracy. The computation of the domain coverage of a rule set requires the classification of all test instances and hence it has the same time complexity as the computation of the predictive accuracy.

EXPERIMENTS

This chapter describes the application of the proposed quality measures to compare the rule sets generated by AQ15c, C4.5, CN2, and GIL for six real world data sets. The experiments show how the variety of the proposed metrics can be used to analyze and compare the different types of rule sets generated by these learning algorithms.

Data Sets Used In The Experiments

For the comparison, six real world data sets were selected most of which are known as standard data sets in the machine learning literature and which are all available from the University of California at Irvine (UCI) Repository of Machine Learning Databases (Merz, Murphy, and Aha, 1994). The data sets for the comparison were chosen from a wide range of domains including three medical, one botanic, one economic, and one political domain. They are not free of noise, for some examples, attribute values may be missing, and the description of the instances by the given collection of attributes may be incomplete in that two instances of distinct classes may have identical values for all attributes. These are problems that inductive learning systems have to cope with in most real world applications. The domains have symbolic and continuous or discrete numeric attributes. Structured domains with partially ordered values are not considered in this comparison. Table 1 summarizes the characteristics of these data sets.

Table 1: Summary of the data sets used in the experiments.

Database name:	Br-cancer-loi	Br-cancer-wi	Lymph	Iris	Credit	Vote 1984
No. of attributes:	9	9	18	4	15	16
Symbolic:	5	0	15	0	9	16
Discrete numeric:	4	9	3	0	3	0
Continuous numeric:	0	0	0	4	3	0
Average no. of values per attribute:	4.56	9.89	3.28	30.75	77.93	2.00
No. of classes:	2	2	4	3	2	2
Class 1:	70.28 %	65.52 %	1.35 %	33.33 %	44.49 %	38.62 %
Class 2:	29.72 %	34.48 %	54.73 %	33.33 %	55.51 %	61.38 %
Class 3:	-----	-----	41.22 %	33.33 %	-----	-----
Class 4:	-----	-----	2.70 %	-----	-----	-----
No. of instances:	286	699	148	150	690	435
No. of attributes with unknown values:	2	1	0	0	7	16
Average percentage of instances with unknown values per attribute:	0.35 %	0.25 %	0.00 %	0.00 %	4.48 %	5.63 %

The first medical domain, the **Breast Cancer Database from the Ljubljana Oncology Institute** (Br-cancer-loi), was originally provided by M. Zwitter and M. Soklic from the University Medical Center, Institute of Oncology, Ljubljana, Yugoslavia. This database repeatedly appeared in the machine learning literature, e.g. in Michalski et. al. (1986b) and Clark and Niblett (1987). The goal is to predict whether a cancer recurrence is likely or not. The data set includes 201 instances classified as “no recurrence” and 85 instances classified as “recurrence”. The data set was chosen, because it consists of real medical data, contains noise and some unknown values, and is well known in the machine learning literature.

The **Wisconsin Breast Cancer Database** (Br-cancer-wi) is the second medical domain. This database was originally obtained from the University of Wisconsin Hospitals, Madison, Wisconsin from Dr. W. H. Wolberg (Wolberg and Mangasarian, 1990). Each of the 699 instances is represented by the values of 9 attributes and is classified as a “benign” or “malignant” cancer tumor. Like the first data set, this set was chosen for being real medical data with noise and some unknown attributes values.

The third medical domain, the **Lymphography Domain** (Lymph), was, like the first medical domain, originally provided by M. Zwitter and M. Soklic from the University Medical Center, Institute of Oncology, Ljubljana, Yugoslavia and similarly it has often appeared in the

machine learning literature. This real-world data set offers a good mix of 9 Boolean attributes, 6 nominal attributes, and 3 numeric attributes and an interesting non-symmetric distribution of the 148 instances among the four classes "normal" with 2 instances, "metastases" with 81 instances, "malign" with 61 instances, and "fibrosis" with 4 instances.

The **Iris data set**, developed by R. A. Fisher (1936), lists the measurements of four characteristics of Iris flowers: petal length, petal width, sepal length, and sepal width. The set includes the measurements of 50 Iris flowers of each species Virginica, Versicolor, and Setosa and the class to be predicted is the species of the Iris plant. The data set was selected, because it is real, contains noise, and is very well known in the pattern recognition literature. While one class is linearly separable from the other two, the latter are not linearly separable from each other.

The **Credit Approval Database** (Credit) concerns credit card applications and was submitted to the UCI Repository of Machine Learning Databases and previously used by J.R. Quinlan (1987; 1993). All attribute names and values have been changed to meaningless symbols to protect the confidentiality of the data. This real-world data set is interesting, because there is a good mix of 6 continuous numeric attributes and 9 discrete-valued nominal attributes having from 2 to 14 possible values. There are also a few missing values. The 690 cases are split 44 % to 56 % between the two classes.

The **1984 United States Congressional Voting Records Database** (Vote 1984) was compiled by Jeff Schlimmer from Washington State University. It includes attributes representing the votes for each of the 435 United States House of Representatives Congressmen on 16 key votes. The votes used in this data set were combined and simplified to either "yea" or "nay." The predicted class for this data set is the party affiliation, Democrat or Republican. This data set was chosen, because the two classes did not always vote along party lines and there are unknown attribute values for each attribute because of congressmen voting present or not voting all.

Experimental Setup

The ***v*-fold cross validation** technique was used to generate pairs of training and test data sets for each domain. This technique is preferable when the number of instances in the data set

is a few hundred or less in total (Breiman et al., 1984). In *v*-fold cross validation, the original data set is randomly partitioned in *v* subsets, each containing approximately the number of instances. In each experiment, one subset is used for testing and the other *v-1* subsets are used for training. In this study, *v* was chosen as four and the results were averaged for each set of tests.

For **AQ15c**, the maximum *star* size was set to 10 and the default criteria table was used, which focuses on maximizing the number of newly covered positive events, i.e. events that are not covered by previous complexes, and minimizing the number of extended selectors, i.e. selectors with single values or disjunctions of values. In addition, the *trim(mini)* option was invoked to specify the generality of the resulting rules, specifying that rules should be as simple as possible, involving the minimum number of extended selectors, each with a minimum number of values. Ambiguous examples were always taken as positive examples for the current class, and were therefore possibly covered by more than one classification rule. Unordered rules were induced in the "intersecting covers" mode, which produces rules which may intersect over areas in the learning space in which there are no events. Ordered rules were produced in the "variable-valued logic" mode, which assumes for the rules for class *n* that the rules for the classes *1* to *n-1* are not satisfied and hence uses a default rule for the last class. The evaluation of the rules by AQ15c itself as listed in the accuracy result tables was obtained by the evaluation method described in the section about AQ15 in the description of the learning algorithms and in (Michalski et al., 1986b).

For most data sets, the **C4.5** decision trees and ordered rules were generated using the default *confidence level* of *25 %*. Only for the Credit data set, the confidence level was set to *10 %* in order to use heavier pruning on trees induced for this domain. The decision trees were built with an option to avoid near-trivial tests in which almost all training cases have the same outcome, which can lead to odd trees with little predictive power. This option requires that any test used in the tree must have at least two outcomes with a minimum number of cases or, to be more precise, the sum of the weights of the cases for at least two of the subsets T_i must attain some minimum. The default minimum of *2* was used for all data sets but the credit data set, where the minimum was set to *15* in order to prevent an overspecialization of the induced trees.

The unordered and ordered **CN2** rules for all data sets were created with the same parameter setting. The maximum *star* size was set to *10*, the Laplacian error estimate was employed as ranking function for the complexes, and the *confidence treshold* was set to *10 %*. Similarly, all **GIL** rules were induced with the same parameters. The weights used for the evaluation function, the application probabilities of the genetic operators, and the other relevant parameters are listed in Appendix B.

The implementation of the GIL algorithm used for this study can only learn rules for *one class at a time* and it cannot handle *unknown attribute values*. In order to obtain rule sets able to predict several classes, one unordered rule set was induced for each class and the union of these rule sets was considered for further evaluation. The unknown attribute values for a particular attribute were replaced by the most common value of that attribute in the data set. This was only done for GIL, but not for the other learning algorithms, which can handle unknown attribute values.

For the Iris and the Credit data sets, the GIL system was not able to complete the induction of a single rule set on a SUN SPARC 20 workstation with 98 MB main memory within 12 hours. Both data sets contain several *continuous numeric attributes*. The GIL system does not explicitly handle continuous attribute values, but internally represents each encountered attribute value as a discrete value. This increases the size of the internal representation of each instance and each complex significantly, because for each possible value of each attribute one bit is reserved in this representation. The problem was solved by introducing intervals for each attribute with more than 25 possible numeric values. The intervals were based on the *5* % quantiles q_0, q_5, q_{10}, ..., q_{95}, q_{100} of the particular attribute, where each q_i is the smallest encountered value of the attribute greater than or equal to the attribute value of at least I % of the instances in the data set. Each value of the attribute was mapped to the smallest q_i greater or equal this value. As a result, the attributes with *5* % quantile-based intervals have at most 21 distinct values. This interval introduction means a potential loss of information, but it enabled the GIL system to handle the Iris and the Credit data sets and therefore was considered acceptable. The interval introduction reduced the average number of values per attribute for the Iris data set from 30.75 to 21.25 and for the Credit data set from 77.93 to 9.33.

All rule sets of all learning algorithms were transformed into a unified format and evaluated by the same *evaluation program*, which computed all measures proposed in the previous section. For this evaluation program, unknown attribute values were handled like for GIL, i.e. the unknown values were replaced by the most common value of the corresponding attribute.

Predictive Accuracy Results

The average predictive accuracy results of all combinations of learning algorithm and data set are summarized in Table 2. For each combination, the average accuracy of the corresponding learning algorithm according to its own evaluation method is presented in brackets ('[', ']'), but the following comparison only considers the results of the evaluation method described in the previous chapter, which is uniformly applied to all rule sets. Since the implementation of the GIL algorithm does not allow the evaluation of a separate test set but automatically picks a fraction of the training set for its computation of the accuracy, the values given in brackets for GIL, are not comparable with the other accuracy values.

Table 2 lists the average accuracy of each rule set on each class separately as well as the average overall accuracy on the test sets and its standard deviation. Since a good rule set should be better than a default rule always predicting the majority class in the training set, the average accuracy values for such a default rule are presented as well. The accuracy values in the default rule column are computed as the average of the four experiments. The default rule in a particular experiment always predicts the majority class of the training set in this experiment, which is not necessarily identical to the majority class of the whole data set. The number in brackets represents the percentage of the instances of the majority class in the whole data set. Table 2 only shows the average result of the $v=4$ experiments for each combination of learning algorithm and data set. The detailed results of each experiment are listed in Appendix C.

Unlike the instance classification procedure of AQ15c, the uniform evaluation procedure does not use analogical matching to cover otherwise uncovered instances. For some data sets, its accuracy values therefore differ significantly from those obtained by AQ15c. For the Credit and the Vote 1984 data sets, this leads to very low accuracy results, especially for the

Table 2: Average predictive accuracy results [in %].

Algorithm:	AQ15c ordered rules	AQ15c unordered rules	C4.5 decision tree	C4.5 ordered rules	CN2 ordered rules	CN2 unordered rules	GIL unordered rules	Default rule
Br-cancer-loi:	[37.76]	[70.28]	[74.10]	[71.68]	[66.09]	[71.65]	[66.08]	[70.28]
No recurr.:	75.36	74.33	94.89	84.63	90.46	92.98	81.15	100.00
Recurrence:	44.05	28.07	24.81	44.59	13.24	21.43	25.33	0.00
Overall mean:	**66.10**	**60.50**	**74.10**	**71.68**	**66.09**	**72.02**	**64.32**	**70.26**
Std. deviation:	4.20	2.08	5.23	3.04	4.04	2.66	4.95	3.92
Br-cancer-wi:	[95.42]	[95.99]	[93.85]	[94.14]	[94.14]	[94.00]	[98.30]	[65.52]
Begnin:	95.23	95.23	95.42	96.30	91.90	98.03	94.73	100.00
Malignant:	96.24	87.86	90.94	90.11	98.30	86.41	93.38	0.00
Overall mean:	**95.57**	**91.28**	**93.85**	**94.14**	**94.14**	**93.99**	**94.42**	**65.52**
Std. deviation:	1.36	1.77	0.86	0.64	1.04	1.43	1.12	1.50
Lymph:	[58.11]	[81.08]	[79.05]	[76.35]	[72.30]	[79.05]	[84.55]	[54.73]
Normal:	100.00	0.00	0.00	0.00	0.00	50.00	0.00	0.00
Metastases:	79.03	80.08	83.65	80.18	88.35	90.98	72.55	100.00
Malign:	69.14	65.43	75.08	74.37	71.07	69.99	66.11	0.00
Fibrosis:	83.33	50.00	83.33	83.33	50.00	33.33	0.00	0.00
Overall mean:	**76.35**	**71.62**	**79.05**	**76.35**	**79.05**	**79.73**	**66.89**	**54.78**
Std. deviation:	8.41	6.76	5.19	4.00	11.21	7.99	9.81	10.20
Iris:	[50.13]	[95.34]	[94.03]	[94.03]	[96.68]	[96.68]	[94.54]	[33.33]
Setosa:	92.50	92.50	100.00	100.00	100.00	100.00	97.22	50.00
Versicolor:	81.16	81.16	93.30	93.30	91.38	91.38	80.34	25.00
Virginica:	98.21	81.25	87.90	87.90	98.08	98.08	86.46	25.00
Overall mean:	**89.94**	**84.62**	**94.03**	**94.03**	**96.68**	**96.68**	**89.37**	**25.34**
Std. deviation:	7.02	5.29	3.92	3.92	2.21	2.21	7.22	2.97
Credit:	[55.51]	[56.81]	[85.07]	[85.38]	[80.14]	[81.60]	[81.34]	[55.51]
+ :	12.71	12.71	84.28	80.89	78.68	70.81	86.03	0.00
- :	98.96	20.21	85.93	88.74	81.17	89.97	82.44	100.00
Overall mean:	**60.59**	**16.83**	**85.07**	**85.21**	**80.14**	**81.59**	**84.06**	**55.10**
Std. deviation:	4.21	7.05	2.95	3.02	1.14	0.62	1.91	1.63
Vote 1984:	[61.38]	[88.51]	[95.86]	[95.63]	[93.79]	[93.79]	[94.92]	[61.38]
Republican:	0.00	0.00	94.04	94.04	92.56	87.04	91.33	0.00
Democrat:	100.00	0.00	97.30	97.30	93.76	96.21	93.99	100.00
Overall mean:	**61.40**	**0.00**	**95.86**	**95.86**	**93.10**	**93.56**	**92.88**	**61.40**
Std. deviation:	4.95	0.00	2.79	2.79	1.04	2.97	1.76	4.95

unordered rules, where no default rule keeps the accuracy from decreasing below a certain level as it happens for the ordered rule sets. The Credit data set contains continuous numeric attribute values, which AQ15c interprets as discrete values. This could result in a very low coverage on the test set, because the attribute values of the test instances may not have occurred in the training data, and hence this could be an alternative explanation for the low accuracy results of the AQ15c rules for this data set. This way of handling continuous attribute values, however, does not explain the poor performance of the unordered AQ15c rules on the Vote 1984 database, which has only Boolean attributes. Not a single test instance is matched by a rule. This is obviously a case where the analogical matching method

of the AQ15c instance classification procedure would have been necessary to use the rules. The accuracy of the ordered rules for this data set looks similarly poor, but through the default rule these rule sets still achieve the default accuracy.

While on most data sets the accuracy values for the rule sets of all algorithms exceed the default accuracy, four out of the seven rule sets for the breast cancer database from the Ljubljana Oncology Institute perform worse than a default rule, which indicates that this domain is very noisy and difficult to learn. The lymphography database from the same institute shows that the algorithms have problems handling very unsymmetric class distributions and hence have a very low accuracy on the minority classes.

Complexity, Generalization, And Coverage Results

The average results for the space and time complexity of the rule sets are shown in Table 3 and Table 4, respectively. The detailed space and time complexity results of the *v=4* test runs for each algorithm and database combination are listed in Appendix D. Table 3 lists the space complexity measures, i.e. the average numbers of rules, complexes, and selectors for each combination of learning algorithm and data sets. The number of rules includes the default rule if a rule set has such a rule. Hence the number of rules equals the number of complexes for the unordered AQ15c rule sets and the unordered GIL rule sets, which do not have a default rule, and is one higher than the number of complexes for all ordered rule sets and the unordered CN2 rule sets, which have default rules. The C4.5 decision trees always cover the full domain without any need for a default rule. Table 4 contains the time complexity results, i.e. the average numbers of complexes and selectors that needed to be tested to classify a test instance. The most significant result visible in these two tables is the high space and time complexity of the GIL rules compared to all other rule sets.

Table 5 shows the average generalization and coverage results. The detailed results of the *v=4* test runs for each algorithm and database combination are listed in Appendix D. Usually a high *generalization* rate is desirable, but it does not automatically imply that the rule set has a high quality. For example, a rule set consisting of a default rule only has a very high generalization rate, but of course it cannot be considered to be a good rule set.

Table 3: Average space complexity results [in number of rules, complexes, and selectors respectively].

Algorithm:	AQ15c ordered rules	AQ15c unordered rules	C4.5 decision tree	C4.5 ordered rules	CN2 ordered rules	CN2 unordered rules	GIL unordered rules
Br-cancer-loi:							
No. of rules:	15.75	31.00	9.00	7.25	3.00	4.75	56.75
No. of complexes:	14.75	31.00	9.00	6.25	2.00	3.75	56.75
No. of selectors:	63.50	134.00	11.25	12.25	5.00	11.25	294.25
Br-cancer-wi:							
No. of rules:	7.50	16.25	8.75	8.50	5.50	7.75	36.50
No. of complexes:	6.50	16.25	8.75	7.50	4.50	6.75	36.50
No. of selectors:	30.50	64.25	15.50	16.25	12.25	18.75	177.25
Lymph:							
No. of rules:	10.00	15.00	13.00	10.00	7.00	9.25	19.25
No. of complexes:	9.00	15.00	13.00	9.00	6.00	8.25	19.25
No. of selectors:	27.75	47.75	24.00	24.50	17.75	21.00	100.00
Iris:							
No. of rules:	4.50	6.75	3.50	4.75	3.00	4.75	9.00
No. of complexes:	3.50	6.75	3.50	3.75	2.00	3.75	9.00
No. of selectors:	5.75	11.50	5.00	5.00	3.25	6.25	17.00
Credit:							
No. of rules:	18.75	35.50	7.00	6.50	11.75	11.50	116.75
No. of complexes:	17.75	35.50	7.00	5.50	10.75	10.25	116.75
No. of selectors:	93.50	195.75	10.25	11.75	36.75	36.25	965.50
Vote 1984:							
No. of rules:	2.75	12.25	5.75	6.00	5.00	6.50	16.25
No. of complexes:	1.75	12.25	5.75	5.00	4.00	5.50	16.25
No. of selectors:	6.25	37.75	9.50	11.00	13.00	18.00	64.75

The unordered AQ15c rule sets for the Vote 1984 data set show, that a high generalization, 96.25 %, can go along with a very low coverage, 0.00 %, and a very low accuracy, 0.00 %. Used in addition to other metrics, the generalization measure describes a further characteristic of a rule set. A reduction of the complexity of the knowledge representation is desirable in many domains and one of the motivations to use inductive learning techniques. Furthermore a low generalization may indicate an over-specialization of a rule set and hence point out a potential reason for a low predictive accuracy.

Like the generalization metric, the *domain coverage* measure should only be considered in conjunction with other measures. A low coverage automatically implies a low accuracy and the corresponding rule set can be viewed as not correctly capturing the regularities of the domain. While a low coverage may indicate an over-specialization of a rule set, a high

Table 4: Average time complexity results [as average number of tests per instance].

Algorithm:	AQ15c ordered rules	AQ15c unordered rules	C4.5 decision tree	C4.5 ordered rules	CN2 ordered rules	CN2 unordered rules	GIL unordered rules
Br-cancer-loi:							
Complex tests:	7.19	31.00	1.00	5.03	1.86	3.75	56.75
Selector tests:	15.43	56.52	2.23	6.41	2.64	5.19	117.09
Br-cancer-wi:							
Complex tests:	3.04	16.25	1.00	2.44	2.03	6.75	36.50
Selector tests:	6.57	26.90	3.35	3.90	4.57	11.66	76.85
Lymph:							
Complex tests:	5.61	15.00	1.00	5.10	3.16	8.25	19.25
Selector tests:	10.48	26.46	7.53	8.45	6.78	12.74	43.55
Iris:							
Complex tests:	2.30	6.75	1.00	2.28	1.73	3.75	9.00
Selector tests:	2.66	7.45	2.89	2.63	2.36	5.08	12.40
Credit:							
Complex tests:	16.81	35.50	1.00	3.02	5.58	10.25	116.75
Selector tests:	20.10	42.86	2.93	4.47	12.57	21.80	359.70
Vote 1984:							
Complex tests:	1.75	12.25	1.00	2.18	1.89	5.50	16.25
Selector tests:	1.75	12.25	2.18	2.96	4.40	10.24	29.45

Table 5: Average generalization and coverage results [in %].

Algorithm:	AQ15c ordered rules	AQ15c unordered rules	C4.5 decision tree	C4.5 ordered rules	CN2 ordered rules	CN2 unordered rules	GIL unordered rules
Br-cancer-loi:							
Generalization:	92.66	85.55	95.80	96.62	98.60	97.79	73.54
Coverage:	100.00	84.63	100.00	100.00	100.00	100.00	90.89
Br-cancer-wi:							
Generalization:	98.57	96.90	98.33	98.76	98.70	98.52	93.04
Coverage:	100.00	93.71	100.00	100.00	100.00	100.00	98.14
Lymph:							
Generalization:	90.99	86.49	88.29	90.99	93.69	91.67	82.66
Coverage:	100.00	87.16	100.00	100.00	100.00	100.00	81.76
Iris:							
Generalization:	96.60	94.00	96.89	95.78	97.33	95.78	92.00
Coverage:	100.00	86.63	100.00	100.00	100.00	100.00	96.68
Credit:							
Generalization:	96.38	93.14	98.65	98.74	97.73	97.83	77.45
Coverage:	100.00	18.13	100.00	100.00	100.00	100.00	96.82
Vote 1984:							
Generalization:	99.16	96.25	98.24	98.16	98.47	98.01	95.02
Coverage:	100.00	0.00	100.00	100.00	100.00	100.00	98.16

coverage does not guarantee a high accuracy. Decision trees and rule sets with a default rule automatically cover the full domain without necessarily having a high accuracy.

The low accuracy results of the unordered AQ15c rules on the Credit and Vote 1984 data sets shown in Table 2 were obviously due to the low coverage of the rule sets, 18.13 % and 0.00 % respectively, as listed in Table 5. Rules may have a high accuracy on the instances they cover, if their coverage is low, their overall accuracy will be low, too. Hence a look at the coverage value my help to find a possible reason for the low accuracy of a rule set.

Comparison Of The Learning Algorithms

The experiments described above only allow a limited comparison of the learning algorithms used in this work. A really representative comparison should be based on more than six data sets. Furthermore one should take into consideration, that some rules, for example the ones generated by AQ15c, were developed for a different instance classification method. But the experiments can nevertheless serve for a demonstration of how the rule set quality measures can be applied to distinct types of rule sets with distinct rules.

Table 6: Rule set quality metrics averaged over all data sets (with standard deviations).

Algorithm:	AQ15c ordered rules	AQ15c unordered rules	C4.5 decision tree	C4.5 ordered rules	CN2 ordered rules	CN2 unordered rules	GIL unordered rules	Default rule
Predictive accuracy:	74.99 %	54.14 %	86.99 %	86.21 %	84.87 %	86.26 %	81.99 %	55.40 %
Std. deviation:	(13.66 %)	(34.10 %)	(8.25 %)	(9.37 %)	(10.82 %)	(9.03 %)	(12.05 %)	(14.51 %)
No. of rules:	9.88	19.46	7.83	7.17	5.88	7.42	42.42	-----
Std. deviation:	(5.75)	(10.28)	(2.96)	(1.71)	(2.98)	(2.42)	(36.75)	-----
No. of complexes:	8.88	19.46	7.83	6.17	4.88	6.42	42.42	-----
Std. deviation:	(5.75)	(10.28)	(2.96)	(1.71)	(2.98)	(2.42)	(36.75)	-----
No. of selectors:	37.88	81.83	12.58	12.17	14.67	18.58	269.79	-----
Std. deviation:	(31.50)	(63.32)	(5.96)	(6.03)	(11.03)	(9.35)	(323.51)	-----
Complex tests:	6.12	19.46	1.00	3.34	2.71	6.38	42.42	-----
Std. deviation:	(5.15)	(10.28)	(0.00)	(1.25)	(1.37)	(2.35)	(36.75)	-----
Selector tests:	9.50	28.74	3.52	4.80	5.55	11.12	106.51	-----
Std. deviation:	(6.63)	(16.86)	(1.84)	(2.04)	(3.46)	(5.62)	(118.22)	-----
Generalization:	95.63 %	92.06 %	96.03 %	96.51 %	97.42 %	96.60 %	85.62 %	-----
Std. deviation:	(2.95 %)	(4.46 %)	(3.60 %)	(2.70 %)	(1.74 %)	(2.37 %)	(8.22 %)	-----
Coverage:	100.00 %	61.71 %	100.00 %	100.00 %	100.00 %	100.00 %	93.74 %	100.00 %
Std. deviation:	(0.00 %)	(37.69 %)	(0.00 %)	(0.00 %)	(0.00 %)	(0.00 %)	(5.90 %)	(0.00 %)

Table 6 shows the values of all rule set quality metrics for the rule sets of each algorithm averaged over all data sets. In general, the decision trees generated by the C4.5 system had a slightly higher predictive accuracy than the other rule sets, but, as shown in Appendix F, this finding is not statistically significant (at the 95 % significance level). The comparatively low accuracy of the AQ15c rule sets is mainly due to the rule sets for the Credit and Vote 1984 data sets. On these two data sets, all other algorithms induced significantly better rule sets.

The space complexity and generalization measures show that AQ15c and especially GIL tend to produce more complex rule sets than C4.5 and CN2. For most data sets the size of the GIL rule sets is a multiple of the size of the other rule sets, but obviously this has no significant impact on the accuracy of these rule sets. For humans, however, these rule sets are less understandable due to their size. Furthermore the average numbers of complex and selector tests indicate that ordered rules and especially decision trees have a better time complexity than unordered rules. The results for the space complexity metrics show that the ordered rules induced by an algorithm producing ordered and unordered rules alternatively are less complex than the unordered rules generated by the same algorithm.

Comments On Some Additional Observations

As described in the section about the experimental setup, the GIL system had problems handling the Credit and the Iris data sets due to the large number of distinct values for some of their attributes. For the comparison of the learning algorithms presented in this work, the problem was solved for GIL by introducing 5 % quantiles for numeric attributes with more than 25 distinct values and thereby artificially reducing the complexity of these data sets. For a learning algorithm to be usable for applications with data sets of a reasonable size, it is important to be scalable, i.e. to be able to handle large numbers of attributes, of values per attributes, and of instances. Algorithms employed for *Knowledge Discovery in Databases (KDD)* (Simoudis, Han, and Fayyad, 1996), for example, have to cope this **scalability problem**, in order to be able to find regularities in huge databases within a reasonable amount of time.

Obviously there are two types of complexity to be considered, one associated with the generation of rules and the other associated with the use of rules. While the latter type is

captured by the space and time complexity metrics for rule sets proposed in this work, the first type, the space and time complexity of the learning algorithms, is not captured by these metrics, whose objective it is to describe properties of rule sets and not of learning algorithms. Hence the scalability problem, which is not related to the produced rule sets but to their generation only, is not addressed by the proposed metrics. If not only rule sets have to be compared, but a learning algorithm has to be selected for a particular application, the computational complexity of the available algorithms has to be considered as well, because some algorithms may not be able to handle the complexity of the data set of this application.

A further data set, which is larger and more complex than the six data sets used in the comparison, was taken to examine the scalability of the four learning algorithms used for this study. The **Employment 1992 (Employ92)** data set for females is a subset of the National Longitudinal Survey of Youth (NLSY) data set for 1992, which was obtained from the National Longitudinal Survey (NLS), Youth Cohort, and distributed by the Center of Human Resource Research at Ohio State University, Columbus, Ohio. The objective is to predict whether a female person was employed in 1992 or not based on values for a set of 49 socio-economic variables including information about previous employment, age, education, test scores, geographical location, consumed amounts of drugs and alcohol, marital status, number of young children. The 2882 instances of this data set are described by 22 Boolean, 14 discrete numeric, and 13 continuous numeric attributes. Each of these attributes has an average of 336.92 distinct values. There are no unknown attribute values in this subset of the NLSY data set for 1992. Out of the 2882 instances in the data set, 2078 (72.10 %) belong to one class and 804 (27.90 %) to the other class.

Neither the GIL nor the AQ15c system were able to handle this data set in a reasonable amount of time (if at all), i.e. the programs did not terminate within 12 hours for a single test run on a SUN SPARC 20 with 98 MB main memory when using v-fold cross validation with $v=4$. The C4.5 and CN2 learning systems could both handle the data set. After introducing 5% quantiles to the numeric attributes with more 25 values in this data set, the average number of distinct values per attribute was reduced to 8.35. While AQ15c could process the data set in this less complex form, the GIL system was still not able to learn from this data in

the given time frame of 12 hours. Even the use of 10% and 20% quantiles did not enable GIL to handle this data within the given time frame.

Sometimes preprocessing the data by introducing quantiles and /or reducing the size of the data set by reducing the number of instances and /or the number of attributes may enable a particular algorithm to handle an otherwise unmanageably complex data set, but of course it is preferable to have an algorithm able to handle the data without any need for a user to reduce the complexity of the data. From this point of view, the GIL and the AQ15c systems cannot be recommend for use in large scale applications.

CONCLUSIONS

This work proposed a set of measures to evaluate the quality of a rule set and to compare several rule sets generated by inductive learning algorithms. These measures include the widely used predictive accuracy metric, space and time complexity metrics, and two metrics called generalization and coverage. The application of these measures to the rule sets generated by four different learning algorithms for six real world data sets showed that these measures allow the comparison of rule sets of distinct types, i.e. decision trees, ordered rule sets, and unordered rule sets.

Some applications impose constraints on the rules that can be applied for predictions. In some domains, decisions have to be made fast and hence the average amount of time needed to classify a previously unseen instance should be small. The time complexity measures presented in this work can be employed to take care of that need. Other applications require their knowledge representation not to be too complex, which motivated the space complexity metrics proposed in this work.

The variety of measures allows the users to pick the metrics relevant for their particular applications. The additional measures do not only reveal further characteristics of a rule set, but in some cases allow to find an explanation for the low predictive accuracy of a rule set. A low coverage, for example, indicates that the rule set is not representative for the particular domain, and a low generalization indicates over-specialization.

REFERENCES

Breiman, L., Friedman, J.H., Ohlsen, R.A., and Stone, C.J. (1984). **Classification And Regression Trees**. Bedmont, CA, Wadsworth International.

Documenta Geigy Scientific Tables, 6th Edition, p.185 (with modifications).

Clark, P., and Boswell, R. (1991). *Rule Induction With CN2: Some Recent Improvements.* In Y. Kodratoff, editor, **Proceedings of the Fifth European Machine Learning Conference (EWSL-91)**, pp. 151-163, Berlin, Germany, Springer-Verlag. [http://www.cs.utexas.edu/users/pclark/papers/newcn.ps]

Clark, P., and Niblett, T. (1987). *Induction In Noisy Domains.* In I. Bratko and N. Lavrac, editors, **Progress in Machine Learning: Proc. 2nd European ML Conference (EWSL-87)**, pp. 11-30, Sigma, Wilmslow, UK. [http://www.cs.utexas.edu/users/pclark/papers/ewsl87.ps]

Clark, P., and Niblett, T. (1989). *The CN2 Induction Algorithm*. **Machine Learning**, Vol. 3, No. 4, pp. 261-283, Kluwer Academic Publishers.

Fisher, R.A. (1936). *The Use Of Multiple Measurements In Taxonomic Problems*. **Annual Eugenics**, Vol. 7, Part II, pp. 179-188.

Gallion, R., St. Clair, D.C., Sabharwal, C., and Bond, W.E. (1993). *Dynamic ID3: A Symbolic Learning Algorithm For Many-Valued Attribute Domains*. **Applied Computing: States of the Art and Practice, Proceedings of the 1993 Symposium on Applied Computing (SAC '93)**, pp. 14-20, ACM Press, Indianapolis, IN.

Hahn, G.J. and Meeker, W.Q. (1991). **Statistical Intervals -- A Guide For Practitioners**, pp. 100-108, John Wiley & Sons, New York, NY.

Hoel, P.G. (1984). **Introduction To Mathematical Statistics**. Fifth edition, Wiley Series in Probability and Mathematical Statistics, John Wiley & Sons, New York, N.Y.

Hong, J., Mozetic, I., and Michalski, R.S. (1986). *AQ15: Incremental Learning Of Attribute-Based Descriptions From Examples -- The Method And User's Guide*. File No. UIUCDCS-F-86-949, Intelligent Systems Group (ISG) Report 86-5, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL.

Janikow, C.Z. (1991). *Inductive Learning Of Decision Rules From Attribute-Based Examples: A Knowledge-Intensive Genetic Algorithm Approach*. Ph. D. dissertation, University of North Carolina at Chapel Hill, Chapel Hill, NC.

Janikow, C.Z. (1993). *A Knowledge-Intensive Genetic Algorithm For Supervised Learning*. **Machine Learning**, Vol. 13, pp. 189-228, Kluwer Academic Publishers, Boston, MA.

Kalbfleish, J. (1979). **Probability And Statistical Inference** (Vol. 2). Springer Verlag, New York, NY.

Klinkenberg, R.H., and St. Clair, D.C. (1996). *Rule Set Quality Measures For Inductive Learning Algorithms*. In C.H. Dagli, M. Akay, C.L.P. Chen, B.R. Fernández, and J. Gosh, editors, **Intelligent Engineering Systems Through Artificial Neural Networks**, Vol. 6, **Proceedings of the Conference Artificial Neural Networks In Engineering 1996 (ANNIE' 96)**, pp. 161-168, ASME Press, New York, NY.

Merz, C.J., Murphy, P. M., and Aha, D. W. (1994). *UCI Repository Of Machine Learning Databases*. Department of Information and Computer Science, University of California at Irvine, Irvine, CA. [http://www.ics.uci.edu/~mlearn/MLRepository.html]

Michalski, R.S. (1969). *On The Quasi-Minimal Solution Of The General Covering Problem*. **Proceedings of the Fifth International Symposium on Information Processing (FCIP 69)**, Vol. A3 (Switching Circuits), Bled, Yugoslavia, pp. 125-128.

Michalski, R.S. (1983). *Theory And Methdodology Of Inductive Learning*. In R.S. Michalski, J.G. Carbonell, and T.M. Mitchell, editors, **Machine Learning -- An Artificial Intelligence Approach,** Vol. 1, pp. 83-134, Tioga Publishing, Palo Alto, CA.

Michalski, R.S., and Larson, J. (1975). *AQVAL / 1 (AQ7) User's Guide And Program Description.* Report No. 731, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL.

Michalski, R.S., and Larson, J. (1983). *Incremental Generation Of VL_1 Hypotheses: The Underlying Methodology And The Description Of The Program AQ11*. File No. UIUCDCS-F-83-905, Intelligent Systems Group (ISG) Report 83-5, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL.

Michalski, R.S., and McCormick, B.H. (1971). *Interval Generalization Of Switching Theory*. Report No. 442, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL.

Michalski, R.S., Mozetic, I., Hong, J., and Lavrac, N. (1986a). *The AQ15 Inductive Learning System: An Overview And Experiments*. Report No. UIUCDCS-R-86-1260, Intelligent Systems Group (ISG) Report 86-20, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL.

Michalski, R.S., Mozetic, I., Hong, J., and Lavrac, N. (1986b). *The Multipurpose Incremental Learning System AQ15 And Its Testing Application To Three Medical Domains*. **Proceedings of the Fifth National Conference on Artificial Intelligence (AAAI-86)**, Vol. 2, pp. 1041-1045, Morgan Kaufmann.

Quinlan, J.R. (1986). *Induction Of Decision Trees*. **Machine Learning**, Vol. 1, No. 1, pp. 81-106, Kluwer Academic Publishers, Boston, MA.

Quinlan, J.R. (1987). *Simplifying Decision Trees.* **International Journal of Man-Machine Studies**, Vol. 27, No. 3, pp. 221-234.

Quinlan, J.R. (1993). **C4.5 : Programs For Machine Learning**. Morgan Kaufmann, San Mateo, CA.

Rissanen, J. (1983). *A Universal Prior For Integers And Estimation By Minimum Description Length.* **Annals of Statistics**, Vol. 11, No. 2, pp. 416-431.

Rivest, R.L. (1987). *Learning Decision Lists*. **Machine Learning**, Vol. 2, pp. 229-246, Kluwer Academic Publishers, Boston, MA..

Spiegel, M.R. (1961). **Theory And Problems Of Statistics**. Schaum's Outline Series, McGraw-Hill, New York, NY.

Simoudis, E., Han, J.W., and Fayyad, U., editors (1996). **Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)**, AAAI Press, Menlo Park, CA.

Wnek, J., Kaufman, K., Bloedorn, E., and Michalski, R.S. (1995). *Selective Inductive Learning System AQ15c: The Method And User's Guide*. Center for Machine Learning and Inference, George Mason University, Fairfax, VA.

Wolberg, W.H. and Mangasarian, O.L. (1990). *Multisurface Method Of Pattern Separation For Medical Diagnosis Applied To Breast Cytology*. **Proceedings of the National Academy of Sciences**, Vol. 87, pp. 9193-9196, USA.

APPENDIX A

Computation Of The Upper Boundary Of A Confidence Interval For The Probability In A Binomial Probability Distribution

APPENDIX A

Computation Of The Upper Boundary Of A Confidence Interval For The Probability In A Binomial Probability Distribution

The probability of *k* succesful trials in *n* independent trials with an equal probability *p* of a success for each trial is obtained from by the *binomial distribution*:

$$P(X=k) = \frac{n!}{k!\ (n-k)!}\ p^{k}\ (1-p)^{n-k}$$

The *cumulative binomial probability distribution function* describes the probability of at least *a* succesful trials in *n* trials:

$$P(X \geq a) = \sum_{k=a}^{n} \frac{n!}{k!\ (n-k)!}\ p^{k}\ (1-p)^{n-k}$$

For the problem considered here, the parameter *p* is unknown and the upper bound $U_{CF}(E,N)$ of a confidence interval for the probability *p* at confidence level *CF* has to be computed given the number of successful trials (errors), *E*, and the total number of trials, *N*. In order to compute the confidence interval, the inverse of the cumulative binomial probability distribution function has to be computed or at least approximated.

In the implementation of his C4.5 system, Quinlan (1993) uses a modified version of the approximation for $U_{CF}(E,N)$ described in (Documenta Geigy Scientific Tables). Hahn and Mekker (1980) describe an alternative way of computing this upper bound of a binomial confidence interval, which, for small values of *n*, is based on the *F* distribution and, for large values of *n*, is based on the normal distribution, respectively.

For *E* successful trials in a sample of size *N*, the upper boundary of a conservative one-sided $100(1-\alpha)\%$ confidence interval for *p*, the true proportion of successful trials, is given by :

$$U_{(1-\alpha)}(E,N) = \frac{1}{1 + \dfrac{(N-E)}{(E+1)\ F_{(1-\alpha;\ 2E+2,\ 2N-2E)}}}$$

where $F_{(\gamma;r1,r2)}$ is the $100\gamma th$ percentile of the F distribution with r_1 and r_2 degrees of freedom. For samples of small size, the critical values of the F distribution can be obtained by interpolating from tabulated values of that distribution. For large samples where $n(E/N)$ and $n(1-E/N)$ exceed 10, the upper bound of the confidence interval can be approximated uisng tabulations of the normal distribution percentiles:

$$U_{(1-\alpha)}(E,N) = p_0 + z_{(1-\alpha)}\sqrt{\frac{p_0(1-p_0)}{N}} \qquad where \qquad p_0 = \frac{E}{N}$$

and where $z(\gamma)$ is the $100\gamma th$ percentile of the standard normal distribution.

APPENDIX B

Parameter Setting For The GIL Algorithm

APPENDIX B

Parameter Setting For The GIL Algorithm

General parameters of the genetic algorithm:

Population size:	50 chromosomes;
Main iteration loop for rule induction:	300 iterations or until best chromosome is complete and consistent;
Final iteration loop for cost reduction:	100 iterations;

Elitism was used, i.e. the best chromosome of each generation was always kept and left unchanged.

Weights and exponent used for the evaluation function:

$w_1 = 1.0$

$w_2 = 1.0$

$w_3 = 0.002$ in the main loop and $w_3 = 0.03$ in the final loop

$f = 1.4$

Initial application probabilities of the genetic operators:

Note: The values specified here only reflect the ratio of the initial application probabilities. The actual probabilities were scaled in such a way that 80% of the selected chromosomes of a new generation were updated by the application the genetic operators.

Genetic operators on rule set level:

Rules exchange:	20 (operand selection probability: 20%)
Rules copy:	10
New event:	40
Rules generalization:	40
Rules drop:	40

Rules specialization: 40

Rules multi generalization: 10 (operand selection probability: 10%)

Rules multi drop: 10 (operand selection probability: 10%)

Rules multi specialization: 10 (operand selection probability: 10%)

Rules dir. reference extension: 20

Rules dir. reference restriction: 20

Genetic operators on rule level:

Rule split: 4

Condition drop: 8

Turning conjunction into disjunction: 0

Condition introduce: 8

Rule directed split: 12

Genetic operators on condition level:

Reference change: 2

Reference extension: 3 (50% probability for whole range fill)

Reference restriction: 3 (50% probability for whole range removal)

Split of the data set and initial population:

The GIL system used 90% of the data set for training and 10% for testing to compute the accuracies of its rules. Hence only 90% of the training data presented to GIL in each fold were really used for the rule set induction. The initial populations were composed of positive training events (50%) and random chromosomes (50%). Each random chromosome contained at least one complex. The probability for each further complex was 60%, i.e. ca. 0.4*100%=40% of the randomly generated chromosomes had one complex, ca. 0.4*60%=24% had two complexes, ca. 0.4*36%=14.4% had three complexes, etc..

APPENDIX C

Detailed Experimental Results: Predictive Accuracies

APPENDIX C

Detailed Experimental Results: Predictive Accuracies

Tables 7 to 10 list the predictive accuracy results for each of the *v=4* folds separately. The values in brackets ('[', ']') represent the average accuracy values according to the evaluation procedures of the learning algorithms. All other values were computed by the uniform evaluation program used for the experiments described in this study.

Table 7: Predictive accuracy results for fold 1 [in %].

Algorithm:	AQ15c ordered rules	AQ15c unordered rules	C4.5 decision tree	C4.5 ordered rules	CN2 ordered rules	CN2 unordered rules	GIL unordered rules	Default rule
Br-cancer-loi:	[34.72]	[69.44]	[75.00]	[68.06]	[70.83]	[70.83]	[78.60]	
No recurr.:	68.63	66.67	92.16	74.51	100.00	86.27	70.59	100.00
Recurrence:	47.62	33.33	33.33	52.38	0.00	28.57	33.33	0.00
Overall:	**62.50**	**56.94**	**75.00**	**68.06**	**70.83**	**69.44**	**59.72**	**70.83**
Br-cancer-wi:	[94.86]	[96.57]	[94.29]	[94.29]	[95.43]	[91.43]	[100.00]	
Begnin:	94.74	94.74	93.86	97.37	93.86	99.12	95.61	100.00
Malignant:	95.08	77.05	95.08	88.52	98.36	78.69	93.44	0.00
Overall:	**94.86**	**88.57**	**94.29**	**94.29**	**95.43**	**92.00**	**94.86**	**65.14**
Lymph:	[43.24]	[81.08]	[78.68]	[70.72]	[72.97]	[67.57]	[90.45]	
Normal:	100.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Metastases:	57.14	64.29	78.57	78.57	85.71	85.71	71.43	100.00
Malign:	65.00	65.00	80.00	70.00	70.00	65.00	65.00	0.00
Fibrosis:	50.00	50.00	50.00	50.00	50.00	0.00	0.00	0.00
Overall:	**61.62**	**61.62**	**75.68**	**70.27**	**72.97**	**67.57**	**62.16**	**37.84**
Iris:	[52.63]	[94.74]	[89.47]	[89.47]	[94.74]	[94.74]	[96.97]	
Setosa:	88.24	88.24	100.00	100.00	100.00	100.00	100.00	0.00
Versicolor:	87.50	87.50	87.50	87.50	87.50	87.50	50.00	100.00
Virginica:	100.00	92.31	76.92	76.92	92.31	92.31	69.23	0.00
Overall:	**92.11**	**89.47**	**89.47**	**89.47**	**94.74**	**94.74**	**78.95**	**21.05**
Credit:	[53.76]	[56.65]	[83.82]	[83.82]	[81.50]	[81.50]	[82.35]	
+ :	0.00	0.00	80.00	80.00	92.50	65.00	82.50	0.00
- :	100.00	18.28	87.10	87.10	72.04	95.70	79.57	100.00
Overall:	**53.76**	**9.83**	**83.82**	**83.82**	**81.50**	**81.50**	**80.92**	**53.76**
Vote 1984:	[58.00]	[90.83]	[93.58]	[93.58]	[94.50]	[93.58]	[95.85]	
Republican:	0.00	0.00	91.30	91.30	95.65	86.96	91.30	0.00
Democrat:	100.00	0.00	95.24	95.24	93.65	96.83	93.65	100.00
Overall:	**57.80**	**0.00**	**93.58**	**93.58**	**94.50**	**92.66**	**92.66**	**57.80**

Table 8: Predictive accuracy results for fold 2 [in %].

Algorithm:	AQ15c ordered rules	AQ15c unordered rules	C4.5 decision tree	C4.5 ordered rules	CN2 ordered rules	CN2 unordered rules	GIL unordered rules	Default rule
Br-cancer-loi:	[40.27]	[75.00]	[81.94]	[76.39]	[59.72]	[75.00]	[50.00]	
No recurr.:	72.73	72.73	100.00	89.09	61.82	98.18	85.45	100.00
Recurrence:	35.29	23.53	23.53	35.29	52.94	5.88	29.41	0.00
Overall:	**63.89**	**61.11**	**81.94**	**76.39**	**59.72**	**76.39**	**72.22**	**76.39**
Br-cancer-wi:	[93.71]	[94.29]	[94.86]	[94.86]	[94.29]	[96.00]	[99.00]	
Begnin:	93.28	93.28	95.80	95.80	93.28	98.32	94.96	100.00
Malignant:	94.64	85.71	92.86	92.86	96.43	91.07	94.64	0.00
Overall:	**93.71**	**90.86**	**94.86**	**94.86**	**94.29**	**96.00**	**94.86**	**68.00**
Lymph:	[67.57]	[78.38]	[81.08]	[81.08]	[97.30]	[89.19]	[75.00]	
Normal:	---	---	---	---	---	---	---	---
Metastases:	91.67	79.17	79.17	79.17	95.83	87.50	79.17	100.00
Malign:	61.54	53.85	84.62	84.62	100.00	92.31	92.13	0.00
Fibrosis:	---	---	---	---	---	---	---	---
Overall:	**81.08**	**70.27**	**81.08**	**81.08**	**97.03**	**89.19**	**83.78**	**64.86**
Iris:	[28.94]	[94.74]	[94.74]	[94.74]	[97.37]	[97.37]	[96.67]	
Setosa:	92.86	92.86	100.00	100.00	100.00	100.00	100.00	0.00
Versicolor:	100.00	100.00	100.00	100.00	92.31	92.31	32.31	0.00
Virginica:	100.00	63.64	81.82	81.82	100.00	100.00	90.91	100.00
Overall:	**97.37**	**86.84**	**94.74**	**94.74**	**97.37**	**97.37**	**94.74**	**28.95**
Credit:	[56.65]	[56.65]	[87.86]	[89.02]	[80.35]	[81.50]	[75.96]	
+ :	14.67	14.67	94.67	89.33	96.00	64.00	85.33	0.00
- :	98.97	12.24	82.65	89.80	67.35	94.90	84.69	100.00
Overall:	**62.43**	**13.29**	**87.86**	**89.60**	**79.77**	**81.50**	**84.97**	**56.65**
Vote 1984:	[61.47]	[88.07]	[98.17]	[97.25]	[94.50]	[98.17]	[93.80]	
Republican:	0.00	0.00	97.62	97.62	80.95	97.62	92.86	0.00
Democrat:	100.00	0.00	98.51	98.51	98.51	98.51	89.55	100.00
Overall:	**61.47**	**0.00**	**98.17**	**98.17**	**91.74**	**98.17**	**90.83**	**61.47**

Table 9: Predictive accuracy results for fold 3 [in %].

Algorithm:	AQ15c ordered rules	AQ15c unordered rules	C4.5 decision tree	C4.5 ordered rules	CN2 ordered rules	CN2 unordered rules	GIL unordered rules	Default rule
Br-cancer-loi:	[40.85]	[60.91]	[71.83]	[71.83]	[66.20]	[71.83]	[69.05]	
No recurr.:	82.98	80.85	95.74	95.74	100.00	97.87	91.49	100.00
Recurrence:	54.17	25.00	25.00	25.00	0.00	20.83	12.50	0.00
Overall:	**73.23**	**61.97**	**71.83**	**71.83**	**66.20**	**71.83**	**64.79**	**66.20**
Br-cancer-wi:	[96.57]	[96.00]	[93.71]	[94.29]	[94.29]	[94.29]	[97.05]	
Begnin:	98.21	98.21	97.32	97.32	91.96	98.21	95.43	100.00
Malignant:	95.24	82.54	87.30	88.89	98.41	85.71	93.65	0.00
Overall:	**97.14**	**92.57**	**93.71**	**94.29**	**94.29**	**93.71**	**95.43**	**64.00**
Lymph:	[59.46]	[78.38]	[72.97]	[78.38]	[78.38]	[78.38]	[82.50]	
Normal:	---	---	---	---	---	---	---	0.00
Metastases:	86.36	86.36	83.65	77.27	90.91	95.45	68.18	100.00
Malign:	64.29	57.14	50.00	78.57	57.14	50.00	50.00	0.00
Fibrosis:	100.00	0.00	100.00	100.00	100.00	100.00	0.00	0.00
Overall:	**78.38**	**72.97**	**72.97**	**78.38**	**78.38**	**78.38**	**59.46**	**59.46**
Iris:	[59.46]	[100.00]	[100.00]	[100.00]	[100.00]	[100.00]	[87.57]	
Setosa:	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
Versicolor:	80.00	80.00	100.00	100.00	100.00	100.00	93.33	0.00
Virginica:	100.00	83.33	100.00	100.00	100.00	100.00	100.00	0.00
Overall:	**91.89**	**86.49**	**100.00**	**100.00**	**100.00**	**100.00**	**97.30**	**27.03**
Credit:	[57.56]	[52.33]	[87.79]	[87.21]	[80.23]	[80.81]	[81.62]	
+ :	9.59	9.59	89.04	80.82	61.64	64.38	87.67	0.00
- :	98.99	20.20	86.87	89.90	94.95	92.93	84.85	100.00
Overall:	**61.05**	**15.70**	**87.79**	**86.05**	**80.81**	**80.81**	**86.05**	**54.65**
Vote 1984:	[56.88]	[89.91]	[92.66]	[92.66]	[93.58]	[89.91]	[96.00]	
Republican:	0.00	0.00	87.23	87.23	93.62	78.72	87.23	0.00
Democrat:	100.00	0.00	96.77	96.77	93.55	98.39	96.77	100.00
Overall:	**56.88**	**0.00**	**92.66**	**92.66**	**93.58**	**89.91**	**92.66**	**56.88**

Table 10: Predictive accuracy results for fold 4 [in %].

Algorithm:	AQ15c ordered rules	AQ15c unordered rules	C4.5 decision tree	C4.5 ordered rules	CN2 ordered rules	CN2 unordered rules	GIL unordered rules	Default rule
Br-cancer-loi:	[35.21]	[67.61]	[67.61]	[70.42]	[67.61]	[69.01]	[66.65]	
No recurr.:	77.08	77.08	91.67	79.17	100.00	89.58	77.08	100.00
Recurrence:	39.13	30.43	17.39	65.67	0.00	30.43	26.09	0.00
Overall:	**64.79**	**61.97**	**67.61**	**70.42**	**67.61**	**70.42**	**60.56**	**67.61**
Br-cancer-wi:	[96.55]	[97.13]	[92.53]	[93.10]	[92.53]	[94.25]	[97.15]	
Begnin:	94.69	94.69	94.69	94.69	88.50	96.46	92.92	100.00
Malignant:	100.00	90.16	88.52	90.16	100.00	90.16	91.80	0.00
Overall:	**96.55**	**93.10**	**92.53**	**93.10**	**92.53**	**94.25**	**92.53**	**64.94**
Lymph:	[62.16]	[86.49]	[86.49]	[75.68]	[67.57]	[81.08]	[90.23]	
Normal:	100.00	0.00	0.00	0.00	0.00	100.00	0.00	0.00
Metastases:	80.95	90.48	90.48	85.71	80.95	95.24	71.43	100.00
Malign:	85.71	85.71	85.71	64.29	57.14	71.43	57.14	0.00
Fibrosis:	100.00	100.00	100.00	100.00	0.00	0.00	0.00	0.00
Overall:	**83.78**	**86.49**	**86.49**	**75.68**	**67.57**	**83.78**	**62.16**	**56.76**
Iris:	[59.46]	[91.89]	[91.89]	[91.89]	[94.59]	[94.59]	[96.97]	
Setosa:	88.89	88.89	100.00	100.00	100.00	100.00	88.89	100.00
Versicolor:	57.14	57.14	85.71	85.71	85.71	85.71	85.71	0.00
Virginica:	92.86	85.71	92.86	92.86	100.00	100.00	85.71	0.00
Overall:	**78.38**	**75.68**	**91.89**	**91.89**	**94.59**	**94.59**	**86.49**	**24.32**
Credit:	[54.07]	[61.63]	[80.81]	[81.40]	[78.49]	[82.56]	[85.45]	
+ :	26.58	26.58	73.42	73.42	64.56	89.87	88.61	0.00
- :	97.85	30.11	87.10	88.17	90.32	76.34	80.64	100.00
Overall:	**65.12**	**28.49**	**80.81**	**81.40**	**78.49**	**82.56**	**84.30**	**55.81**
Vote 1984:	[69.44]	[85.19]	[99.07]	[99.07]	[92.59]	[93.52]	[93.80]	
Republican:	0.00	0.00	100.00	100.00	100.00	84.85	93.94	0.00
Democrat:	100.00	0.00	98.67	98.67	89.33	91.09	96.00	100.00
Overall:	**69.44**	**0.00**	**99.07**	**99.07**	**92.59**	**93.52**	**95.73**	**69.44**

APPENDIX D

Detailed Experimental Results: Space And Time Complexity

APPENDIX D

Detailed Experimental Results: Space And Time Complexity

Tables 11 to 14 list the space complexity and Tables 15 to 18 the time complexity results for each of the *v=4* folds separately.

Table 11: Space complexity results for fold 1 [in number of rules, complexes, and selectors respectively].

Algorithm:	AQ15c ordered rules	AQ15c unordered rules	C4.5 decision tree	C4.5 ordered rules	CN2 ordered rules	CN2 unordered rules	GIL unordered rules
Br-cancer-loi:							
No. of rules:	16	31	10	6	4	6	67
No. of complexes:	15	31	10	5	3	5	67
No. of selectors:	68	133	13	8	8	14	364
Br-cancer-wi:							
No. of rules:	8	16	9	8	6	8	52
No. of complexes:	7	16	9	7	5	7	52
No. of selectors:	34	67	16	18	13	17	294
Lymph:							
No. of rules:	10	15	15	10	6	8	18
No. of complexes:	9	15	15	9	5	7	18
No. of selectors:	25	43	28	24	16	17	91
Iris:							
No. of rules:	4	6	3	5	3	4	8
No. of complexes:	3	6	3	4	2	3	8
No. of selectors:	5	9	4	6	3	4	13
Credit:							
No. of rules:	20	37	13	7	13	11	74
No. of complexes:	19	37	13	6	12	10	74
No. of selectors:	103	208	17	13	41	35	571
Vote 1984:							
No. of rules:	8	19	5	6	5	7	16
No. of complexes:	7	19	5	5	4	6	16
No. of selectors:	25	61	8	10	12	19	57

Table 12: Space complexity results for fold 2 [in number of rules, complexes, and selectors respectively].

Algorithm:	AQ15c ordered rules	AQ15c unordered rules	C4.5 decision tree	C4.5 ordered rules	CN2 ordered rules	CN2 unordered rules	GIL unordered rules
Br-cancer-loi:							
No. of rules:	16	33	9	9	3	4	50
No. of complexes:	15	33	9	8	2	3	50
No. of selectors:	65	144	11	18	5	11	250
Br-cancer-wi:							
No. of rules:	7	15	9	10	5	7	54
No. of complexes:	6	15	9	9	4	6	54
No. of selectors:	27	58	16	20	14	19	257
Lymph:							
No. of rules:	10	15	13	9	7	11	20
No. of complexes:	9	15	13	8	6	10	20
No. of selectors:	28	52	24	22	17	24	104
Iris:							
No. of rules:	5	8	3	4	3	5	10
No. of complexes:	4	8	3	3	2	4	10
No. of selectors:	7	15	4	4	3	8	18
Credit:							
No. of rules:	19	38	2	6	9	12	82
No. of complexes:	18	38	2	5	8	11	82
No. of selectors:	97	213	2	10	25	42	653
Vote 1984:							
No. of rules:	1	11	5	6	5	7	18
No. of complexes:	0	11	5	5	4	6	18
No. of selectors:	0	33	8	10	14	22	77

Table 13: Space complexity results for fold 3 [in number of rules, complexes, and selectors respectively].

Algorithm:	AQ15c ordered rules	AQ15c unordered rules	C4.5 decision tree	C4.5 ordered rules	CN2 ordered rules	CN2 unordered rules	GIL unordered rules
Br-cancer-loi:							
No. of rules:	15	29	3	4	2	5	51
No. of complexes:	14	29	3	3	1	4	51
No. of selectors:	60	130	4	5	2	12	262
Br-cancer-wi:							
No. of rules:	7	17	8	9	6	8	24
No. of complexes:	6	17	8	8	5	7	24
No. of selectors:	26	63	14	14	11	19	106
Lymph:							
No. of rules:	10	15	12	10	8	9	22
No. of complexes:	9	15	12	9	7	8	22
No. of selectors:	31	49	22	23	20	22	130
Iris:							
No. of rules:	5	8	4	5	3	5	12
No. of complexes:	4	8	4	4	2	4	12
No. of selectors:	7	15	6	5	4	7	29
Credit:							
No. of rules:	18	33	6	7	12	12	63
No. of complexes:	17	33	6	6	11	11	63
No. of selectors:	92	186	10	14	37	40	489
Vote 1984:							
No. of rules:	1	8	7	6	4	6	15
No. of complexes:	0	8	7	5	3	5	15
No. of selectors:	0	25	12	14	8	14	48

Table 14: Space complexity results for fold 4 [in number of rules, complexes, and selectors respectively].

Algorithm:	AQ15c ordered rules	AQ15c unordered rules	C4.5 decision tree	C4.5 ordered rules	CN2 ordered rules	CN2 unordered rules	GIL unordered rules
Br-cancer-loi:							
No. of rules:	16	31	14	10	3	4	59
No. of complexes:	15	31	14	9	2	3	59
No. of selectors:	61	129	17	18	5	8	301
Br-cancer-wi:							
No. of rules:	8	17	9	7	5	8	16
No. of complexes:	7	17	9	6	4	7	16
No. of selectors:	35	69	16	13	11	20	52
Lymph:							
No. of rules:	10	15	12	11	7	9	17
No. of complexes:	9	15	12	10	6	8	17
No. of selectors:	27	47	22	29	18	21	75
Iris:							
No. of rules:	4	5	4	5	3	5	6
No. of complexes:	3	5	4	4	2	4	6
No. of selectors:	4	7	6	5	3	6	8
Credit:							
No. of rules:	18	34	7	6	13	10	248
No. of complexes:	17	34	7	5	12	9	248
No. of selectors:	82	176	12	10	40	28	2149
Vote 1984:							
No. of rules:	1	11	6	6	6	6	16
No. of complexes:	0	11	6	5	5	5	16
No. of selectors:	0	32	10	10	18	17	77

Table 15: Time complexity results for fold 1 [as average number of tests per instance].

Algorithm:	AQ15c ordered rules	AQ15c unordered rules	C4.5 decision tree	C4.5 ordered rules	CN2 ordered rules	CN2 unordered rules	GIL unordered rules
Br-cancer-loi:							
Complex tests:	7.93	31.00	1.00	3.22	2.61	5.00	67.00
Selector tests:	16.58	54.92	2.89	4.44	3.78	6.99	161.75
Br-cancer-wi:							
Complex tests:	3.23	16.00	1.00	2.51	2.10	7.00	52.00
Selector tests:	6.81	27.00	3.34	4.61	4.32	10.71	115.74
Lymph:							
Complex tests:	4.86	15.00	1.00	4.81	3.16	7.00	18.00
Selector tests:	8.81	25.11	7.49	8.38	6.92	10.19	43.65
Iris:							
Complex tests:	2.03	6.00	1.00	2.13	1.79	3.00	8.00
Selector tests:	2.39	6.45	2.39	2.39	2.55	3.32	8.97
Credit:							
Complex tests:	19.00	37.00	1.00	2.35	5.52	10.00	74.00
Selector tests:	22.19	44.58	3.23	3.66	12.56	18.73	190.36
Vote 1984:							
Complex tests:	7.00	19.00	1.00	2.25	1.72	6.00	16.00
Selector tests:	7.00	19.00	2.11	2.81	4.00	11.07	28.59

Table 16: Time complexity results for fold 2 [as average number of tests per instance].

Algorithm:	AQ15c ordered rules	AQ15c unordered rules	C4.5 decision tree	C4.5 ordered rules	CN2 ordered rules	CN2 unordered rules	GIL unordered rules
Br-cancer-loi:							
Complex tests:	7.11	33.00	1.00	7.26	1.90	3.00	50.00
Selector tests:	16.56	59.40	1.93	8.88	3.10	4.74	99.22
Br-cancer-wi:							
Complex tests:	2.75	15.00	1.00	2.70	1.81	6.00	54.00
Selector tests:	6.07	24.81	3.32	3.99	5.26	11.69	119.83
Lymph:							
Complex tests:	5.81	15.00	1.00	4.41	2.49	10.00	20.00
Selector tests:	11.84	28.19	7.97	7.32	6.11	14.97	41.95
Iris:							
Complex tests:	2.29	8.00	1.00	2.03	1.63	4.00	10.00
Selector tests:	2.66	8.92	2.50	2.42	1.95	5.87	13.00
Credit:							
Complex tests:	16.82	38.00	1.00	2.54	4.66	11.00	82.00
Selector tests:	19.52	44.20	1.49	3.96	10.31	24.54	256.46
Vote 1984:							
Complex tests:	0.00	11.00	1.00	2.25	2.06	6.00	18.00
Selector tests:	0.00	11.00	2.16	2.83	4.98	12.04	33.12

Table 17: Time complexity results for fold 3 [as average number of tests per instance].

Algorithm:	AQ15c ordered rules	AQ15c unordered rules	C4.5 decision tree	C4.5 ordered rules	CN2 ordered rules	CN2 unordered rules	GIL unordered rules
Br-cancer-loi:							
Complex tests:	6.79	29.00	1.00	2.10	1.00	4.00	51.00
Selector tests:	13.85	54.18	1.76	2.42	1.10	5.04	91.58
Br-cancer-wi:							
Complex tests:	2.83	17.00	1.00	2.58	2.24	7.00	24.00
Selector tests:	6.54	27.38	3.24	3.54	4.28	12.28	43.02
Lymph:							
Complex tests:	6.08	15.00	1.00	6.62	3.97	8.00	22.00
Selector tests:	11.32	26.16	8.41	10.30	7.43	13.84	49.54
Iris:							
Complex tests:	2.54	8.00	1.00	2.68	1.73	4.00	12.00
Selector tests:	2.97	8.78	3.27	3.08	2.81	5.78	20.05
Credit:							
Complex tests:	16.40	33.00	1.00	3.92	6.17	11.00	63.00
Selector tests:	21.13	41.74	3.72	5.62	13.60	24.36	143.29
Vote 1984:							
Complex tests:	0.00	8.00	1.00	2.17	1.77	5.00	15.00
Selector tests:	0.00	8.00	2.28	3.66	4.08	8.62	27.27

Table 18: Time complexity results for fold 4 [as average number of tests per instance].

Algorithm:	AQ15c ordered rules	AQ15c unordered rules	C4.5 decision tree	C4.5 ordered rules	CN2 ordered rules	CN2 unordered rules	GIL unordered rules
Br-cancer-loi:							
Complex tests:	6.91	31.00	1.00	7.54	1.93	3.00	59.00
Selector tests:	14.75	57.59	2.34	9.90	2.59	4.00	117.09
Br-cancer-wi:							
Complex tests:	3.35	17.00	1.00	1.95	1.95	7.00	16.00
Selector tests:	6.87	28.41	3.51	3.49	4.40	11.97	28.79
Lymph:							
Complex tests:	5.68	15.00	1.00	4.54	3.00	8.00	17.00
Selector tests:	9.95	26.38	6.27	7.81	6.68	11.97	39.08
Iris:							
Complex tests:	2.35	5.00	1.00	2.27	1.76	4.00	6.00
Selector tests:	2.59	5.65	3.38	2.62	2.11	5.35	7.57
Credit:							
Complex tests:	15.00	34.00	1.00	3.26	5.95	9.00	248.00
Selector tests:	17.56	40.94	3.27	4.62	13.81	19.56	848.67
Vote 1984:							
Complex tests:	0.00	11.00	1.00	2.04	1.98	5.00	16.00
Selector tests:	0.00	11.00	2.19	2.54	4.55	9.24	28.85

APPENDIX E

Detailed Experimental Results: Generalization And Coverage

APPENDIX E

Detailed Experimental Results: Generalization And Coverage

Tables 19 to 22 list the generalization and coverage results for each of the *v=4* folds separately.

Table 19: Generalization and coverage results for fold 1 [in %].

Algorithm:	AQ15c ordered rules	AQ15c unordered rules	C4.5 decision tree	C4.5 ordered rules	CN2 ordered rules	CN2 unordered rules	GIL unordered rules
Br-cancer-loi:							
Generalization:	92.52	85.51	95.33	97.20	98.13	97.20	68.69
Coverage:	100.00	83.33	100.00	100.00	100.00	100.00	88.89
Br-cancer-wi:							
Generalization:	98.47	96.95	98.28	98.47	98.86	98.47	90.08
Coverage:	100.00	91.43	100.00	100.00	100.00	100.00	97.71
Lymph:							
Generalization:	90.99	86.49	86.49	90.99	94.59	92.79	83.78
Coverage:	100.00	75.68	100.00	100.00	100.00	100.00	78.38
Iris:							
Generalization:	96.43	94.64	97.32	95.54	97.32	96.43	92.86
Coverage:	100.00	92.11	100.00	100.00	100.00	100.00	92.11
Credit:							
Generalization:	96.13	92.84	97.49	98.65	97.49	97.87	85.69
Coverage:	100.00	10.98	100.00	100.00	100.00	100.00	95.38
Vote 1984:							
Generalization:	97.55	94.17	98.47	98.16	98.47	97.85	95.09
Coverage:	100.00	0.00	100.00	100.00	100.00	100.00	100.00

Table 20: Generalization and coverage results for fold 2 [in %].

Algorithm:	AQ15c ordered rules	AQ15c unordered rules	C4.5 decision tree	C4.5 ordered rules	CN2 ordered rules	CN2 unordered rules	GIL unordered rules
Br-cancer-loi:							
Generalization:	92.52	84.58	95.79	95.79	98.60	98.13	76.64
Coverage:	100.00	80.56	100.00	100.00	100.00	100.00	97.22
Br-cancer-wi:							
Generalization:	98.66	97.14	98.28	98.09	99.05	98.66	89.69
Coverage:	100.00	94.86	100.00	100.00	100.00	100.00	98.29
Lymph:							
Generalization:	90.99	86.49	88.29	91.89	93.69	90.09	81.98
Coverage:	100.00	83.78	100.00	100.00	100.00	100.00	94.59
Iris:							
Generalization:	95.54	92.86	97.32	96.43	97.32	95.54	91.07
Coverage:	100.00	86.84	100.00	100.00	100.00	100.00	100.00
Credit:							
Generalization:	96.33	92.65	99.61	98.84	98.26	97.68	84.14
Coverage:	100.00	15.03	100.00	100.00	100.00	100.00	95.95
Vote 1984:							
Generalization:	99.69	96.63	98.47	98.16	98.47	97.85	94.48
Coverage:	100.00	0.00	100.00	100.00	100.00	100.00	96.33

Table 21: Generalization and coverage results for fold 3 [in %].

Algorithm:	AQ15c ordered rules	AQ15c unordered rules	C4.5 decision tree	C4.5 ordered rules	CN2 ordered rules	CN2 unordered rules	GIL unordered rules
Br-cancer-loi:							
Generalization:	93.02	86.51	98.60	98.14	99.07	97.67	76.28
Coverage:	100.00	84.51	100.00	100.00	100.00	100.00	90.14
Br-cancer-wi:							
Generalization:	98.66	96.76	98.47	98.28	98.86	98.47	95.42
Coverage:	100.00	94.86	100.00	100.00	100.00	100.00	99.43
Lymph:							
Generalization:	90.99	86.49	89.19	90.99	92.79	91.89	80.18
Coverage:	100.00	91.89	100.00	100.00	100.00	100.00	72.97
Iris:							
Generalization:	95.58	92.92	96.46	95.58	97.35	95.58	89.38
Coverage:	100.00	86.49	100.00	100.00	100.00	100.00	97.30
Credit:							
Generalization:	96.53	93.63	98.84	98.65	97.68	97.68	87.84
Coverage:	100.00	16.28	100.00	100.00	100.00	100.00	97.09
Vote 1984:							
Generalization:	99.69	97.55	97.85	98.16	98.77	98.16	95.40
Coverage:	100.00	0.00	100.00	100.00	100.00	100.00	97.25

Table 22: Generalization and coverage results for fold 4 [in %].

Algorithm:	AQ15c ordered rules	AQ15c unordered rules	C4.5 decision tree	C4.5 ordered rules	CN2 ordered rules	CN2 unordered rules	GIL unordered rules
Br-cancer-loi:							
Generalization:	92.56	85.58	93.49	95.35	98.60	98.14	72.56
Coverage:	100.00	90.14	100.00	100.00	100.00	100.00	87.32
Br-cancer-wi:							
Generalization:	98.48	96.76	98.29	98.67	99.05	98.48	96.95
Coverage:	100.00	93.68	100.00	100.00	100.00	100.00	97.13
Lymph:							
Generalization:	90.99	86.49	89.19	90.09	93.69	91.89	84.68
Coverage:	100.00	75.68	100.00	100.00	100.00	100.00	81.08
Iris:							
Generalization:	96.46	95.58	96.46	95.58	97.35	95.58	94.69
Coverage:	100.00	81.08	100.00	100.00	100.00	100.00	97.30
Credit:							
Generalization:	96.53	93.44	98.65	98.84	97.49	98.07	52.12
Coverage:	100.00	30.23	100.00	100.00	100.00	100.00	98.84
Vote 1984:							
Generalization:	99.69	96.64	98.17	98.17	98.17	98.17	95.11
Coverage:	100.00	0.00	100.00	100.00	100.00	100.00	99.07

APPENDIX F

Significance Tests: Is The Most Accurate Learning Algorithm Significantly More Accurate Than The Other Algorithms ?

APPENDIX F

Significance Tests: Is The Most Accurate Learning Algorithm Significantly More Accurate Than The Other Algorithms ?

In the experiments described in this study, the average predictive accuracy of the decision trees generated by C4.5 was higher than the average predictive accuracy of any other learning algorithm used in the comparison. Therefore this appendix addresses the question, whether the C4.5 decision trees are *significantly* more accurate than the other rule set types (at the 95% confidence level). The following sections describe the statistical tests performed and their results.

First, the one-factor and two-factor models of the **ANalysis Of VAriance (ANOVA)** are employed to determine whether the choice of the algorithm significantly influences the predictive accuracy of the resulting rule set (Hoel, 1984). Then, since the fact of a relationship alone is not specific enough to answer the question of interest, a **series of one-tailed hypothesis tests based on Student's *t* distribution** is used to answer this question (Spiegel, 1961; Hoel, 1984). Each such hypothesis test determines whether the difference of the average accuracies of two algorithms is significant.

Both, the ANOVA tests and the test based on Student's t distribution assume that the underlying distributions are normal and that all distributions have the same variance. Since there is no way to proof this assumptions here, it is sensible to use another test in addition that does not make these assumptions about the distributions. The **sign test** is such a test (Hoel, 1984). When comparing two accuracy value samples, this test only looks which sample has the higher value, i.e. only the sign of the difference of the two accuracy values is considered, but not the amount of the difference.

One-factor ANalysis Of Variance (ANOVA)

The objective of the linear hypothesis model the ANOVA is based on is to separate the total variance of the variable being studied into components that are of experimental interest. In this case the variable being studied is the predictive accuracy of a rule set. When the total

Table 23: Data table for the one-factor ANOVA [rows = rule set type, columns = data set].

Data set: / Algorithm:	1	2	...	j	...	b	Row Mean
1	X_{11}	X_{12}	...	X_{1j}	...	X_{1b}	$\overline{X}_{1.}$
2	X_{21}	X_{22}	...	X_{2j}	...	X_{2b}	$\overline{X}_{2.}$
...	...	...	...	...	...	...	...
I	X_{i1}	X_{i2}	...	X_{ij}	...	X_{ib}	$\overline{X}_{i.}$
...	...	...	...	...	...	...	...
a	X_{a1}	X_{a2}	...	X_{aj}	...	X_{ab}	$\overline{X}_{a.}$
Overall Mean							$\overline{X}$

variation is split up this way, statistical methods can be utilized to eliminate the effects of certain interfering variables. This increases the sensitivity of the experiment.

For the experiments described in this work, seven different types of rule sets were generated for six different data sets. The problem of interest then is to determine whether varying the rule set type in this manner had any effect on the predictive accuracy. For the one-factor ANOVA, the use of different data sets is regarded as replications of the same experiment. For this experiment, the accuracy values of all combinations of rule set type and data set can be arranged in a rectangular array with seven rows and six columns, or more general with *a* rows and *b* columns, as shown in Table 23. The entries in the right margin of the table represent the means of the corresponding row entries. The dot in the subscript indicates that the summation is with respect to the second of the two subscripts, that is with respect to columns.

The linear hypothesis model underlying the one-factor ANOVA assumes that the random variable X_{ij} has a mean μ_{ij} that can be written in the form

$$\mu_{ij} = \mu + \alpha_i$$

where μ denotes the expected value of $\overline{X}$ and α_i denotes the expected value of $\overline{X}_{i.} - \overline{X}$. This assumption states that the mean of the variable X_{ij} is the sum of a general mean μ and a

row effect α_i. In addition to this assumption, the linear hypothesis model assumes that the variables X_{ij} are independently normally distributed with the same variances σ^2.

For the problem of testing the hypothesis, that the theoretical row means μ_i , which are the expected values of the row means $\overline{X}_{i.}$ of Table F.1, are equal, the hypothesis can be written in the form

H_0: $\mu_1 = \mu_2 =. ... = \mu_a$,i.e. the differences of the samples are not significant.

or equivalently in the form

H_0: $\alpha_1 = \alpha_2 =. ... = \alpha_a = 0$, i.e. the differences of the samples are not significant.

The alternative hypothesis of this test is

H_1: at least two of the means are not equal, i.e. the differences are significant.

Hypothesis H_0 may be tested by using the right tail of the F distribution as critical region, where

$$F = \frac{a(b-1)\sum_{i=1}^{a}\sum_{j=1}^{b}(\overline{X}_{i.} - \overline{X})}{(a-1)\sum_{i=1}^{a}\sum_{j=1}^{b}(\overline{X}_{ij} - \overline{X}_{i.})}$$

and where $v_1 = a - 1$ and $v_2 = a\,(b - 1)$ are the degrees of freedom for the variable of interest, in this case the choice of the learning algorithm, and for the error, respectively.

Two-factor ANalysis Of Variance (ANOVA)

The preceding analysis of variance model was relatively simple in the sense that there was only one classified variable of interest. Incorporating additional variables into the analysis leads to a more general linear hypothesis model. In the two-factor model one variable (A) is classified into *a* categories and the other variable (B) into *b* categories and the results of the experiments are displayed in a matrix format similar to Table 23. The difference here is that

Table 24: Data table for the two-factor ANOVA [rows = rule set type, columns = data set].

Algorithm: Data set:	1	2	...	j	...	b	Row Mean
1	X_{11}	X_{12}	...	X_{1j}	...	X_{1b}	$\overline{X}_{1.}$
2	X_{21}	X_{22}	...	X_{2j}	...	X_{2b}	$\overline{X}_{2.}$
...	...	...	...	...	...	...	...
I	X_{i1}	X_{i2}	...	X_{ij}	...	X_{ib}	$\overline{X}_{i.}$
...	...	...	...	...	...	...	...
a	X_{a1}	X_{a2}	...	X_{aj}	...	X_{ab}	$\overline{X}_{a.}$
Column Mean	$\overline{X}_{.1}$	$\overline{X}_{.2}$	...	$\overline{X}_{.j}$	...	$\overline{X}_{.b}$	$\overline{X}$

the various columns in the matrix now represent various classification categories of the second variable instead of repetitions of an experiment as in the earlier model.

Since the second variable (B) is to be analyzed in the same manner as the first variable, it is necessary to include the column means as well as the row means in the analysis. For this model, therefore, Table 23 is replaced by Table 24. The two-factor ANOVA model assumes that the random variable X_{ij} has a mean μ_{ij} that can be written in the form

$$\mu_{ij} = \mu + \alpha_i + \beta_j$$

where μ denotes the expected value of $\overline{X}$, α_i denotes the expected value of $\overline{X}_{i.} - \overline{X}$, and β_j denotes the expected value of $\overline{X}_{.j} - \overline{X}$. This assumption states that the mean of the variable X_{ij} is the sum of a general mean μ, a row effect α_i, and a column effect β_j. In addition to this assumption, the linear hypothesis model assumes that the variables X_{ij} are independently normally distributed with the same variances σ^2.

Now that there are two variables to be classified, it is possible to test two hypotheses. Just as in the one-factor model, it is possible to test for equality of the row means (H_0'), but now it will also be possible to test for equality of column means (H_0''):

1. H_0': $\alpha_1 = \alpha_2 = \ldots = \alpha_a = 0$

   H_1': at least one of the α_i's is not equal to zero.

2. H_0'': $\beta_1 = \beta_2 = \ldots = \beta_b = 0$

   H_1'': at least one of the β_j's is not equal to zero.

The hypotheses H_0' and H_0'' may be tested by using the right tail of the F distribution as critical region, where

$$F' = \frac{(b-1)\sum_{i=1}^{a}\sum_{j=1}^{b}(\overline{X}_{i.} - \overline{X})^2}{\sum_{i=1}^{a}\sum_{j=1}^{b}(\overline{X}_{ij} - \overline{X}_{i.} - \overline{X}_{.j} + \overline{X})^2} \quad \text{and} \quad F'' = \frac{(a-1)\sum_{i=1}^{a}\sum_{j=1}^{b}(\overline{X}_{.j} - \overline{X})^2}{\sum_{i=1}^{a}\sum_{j=1}^{b}(\overline{X}_{ij} - \overline{X}_{i.} - \overline{X}_{.j} + \overline{X})^2}$$

respectively, and where $v_1' = a - 1$, $v_2' = (a - 1)(b - 1)$, and $v_1'' = a - 1$, $v_2'' = (a - 1)(b - 1)$ respectively are the degrees of freedom for the variable of interest, in this case the choice of the learning algorithm and the choice of the data set respectively, and for the error.

Results of the one-factor ANOVA test

The one-factor classification model is used to determine if a relationship exists between the variables of interest: the predictive accuracy and the inductive learning method. Table 25

Table 25: Data table for the one-factor ANOVA [rows = rule set type, columns = data set]: Average predictive accuracy [in %].

Algorithm: \ Data set:	Br-cancer-loi	Br-cancer-wi	Lymph	Iris	Credit	Vote 1984	Row Mean
AQ15c ordered rules	66.10	95.57	76.35	89.94	60.59	61.40	74.99
AQ15c unordered rules	60.50	91.28	71.62	84.62	16.83	0.00	54.14
C4.5 decision tree	74.10	93.85	79.05	94.03	85.07	95.86	86.99
C4.5 ordered rules	71.68	94.14	76.35	94.03	85.21	95.86	86.21
CN2 ordered rules	66.09	94.14	79.05	96.68	80.14	93.10	84.87
CN2 unordered rules	72.02	93.99	79.73	96.68	81.59	93.56	86.26
GIL unordered rules	64.32	94.42	66.89	89.37	84.06	92.88	81.99
Overall Mean							79.35

Table 26: One-factor ANOVA result table (for the 95% confidence level).

Source of Variation	Sum of Squares	Degrees of Freedom	Mean Squares	Computed F Value	Critical F Value
Learning Algorithm:	5070.79	6	845.13	2.67	2.37
Error:	11095.24	35	317.01		
Totals:	16166.04	41			

shows the data the one-factor ANOVA is based on. The entries in the table are the predictive accuracy values for each rule set type on each data set, averaged over the *v=4* folds of the cross-validation experiment described in Appendix C. The results shown in Table 26 indicate that the hypothesis is rejected, because the computed F value is greater than the critical F value at the 95% confidence level (= 5% significance level). Hence the predictive accuracy of a rule set depends on the rule set type, i.e. there exists a relation *accuracy = f (learning algorithm).*

Results of the two-factor ANOVA test

A more sensitive analysis is based on the two-factor classification model. It tests in parallel, whether the different learning algorithms and the different data sets relate to the predictive accuracy of a rule set. The data in Table 27 corresponds to the data in Table 25. Only the column means have been added. As Table 28 shows, both, the choice of the learning algorithm as well as the choice of the data set, have a significant effect on the predictive accuracy of a rule set at the 95% confidence level (= 5% significance level), because the two hypotheses, H_0' and H_0'', can both be rejected. Hence there exists a relation of the form *accuracy = f (learning algorithm, data set).*

The one- and two-factor analysis of variance showed a significant relationship between the accuracy of a rule set and the inductive learning method used to generate this rule set, but it did not allow to answer the question, whether the most accurate inductive learning method is significantly more accurate than *all* other methods. The ANOVA tests only showed that at least the accuracy of one of the methods significantly differs from one of the other methods.

Table 27: Data table for the two-factor ANOVA [rows = rule set type, columns = data set]: Average predictive accuracy [in %].

Algorithm: \ Data set:	Br-cancer-loi	Br-cancer-wi	Lymph	Iris	Credit	Vote 1984	Row Mean
AQ15c ordered rules	66.10	95.57	76.35	89.94	60.59	61.40	74.99
AQ15c unordered rules	60.50	91.28	71.62	84.62	16.83	0.00	54.14
C4.5 decision tree	74.10	93.85	79.05	94.03	85.07	95.86	86.99
C4.5 ordered rules	71.68	94.14	76.35	94.03	85.21	95.86	86.21
CN2 ordered rules	66.09	94.14	79.05	96.68	80.14	93.10	84.87
CN2 unordered rules	72.02	93.99	79.73	96.68	81.59	93.56	86.26
GIL unordered rules	64.32	94.42	66.89	89.37	84.06	92.88	81.99
Column Mean	67.83	93.91	75.58	92.19	70.50	76.09	79.35

Table 28: Two-factor ANOVA result table (for the 95% confidence level).

Source of Variation	Sum of Squares	Degrees of Freedom	Mean Squares	Computed F Value	Critical F Value
Learning Algorithm:	5070.79	6	845.13	3.73	2.42
Data Set:	4290.36	5	858.07	3.78	2.53
Error:	6804.89	30	226.83		
Totals:	16166.04	41			

Significance test for the difference of the means of two small samples

In order to obtain a more precise answer to the question of interest, the average accuracy values of each of the different rule set types has to be compared to that of the C4.5 decision trees, the rule set type with the highest average accuracy, separately. The comparison of the learning algorithms in this study is only based on six data sets and hence a significance test for small samples (sample size $N<30$) has to be applied. Here *"Student's" t distribution* has been used (Spiegel, 1961; Hoel, 1984).

The goal of the significance test for a particular pair of learning algorithms is to determine whether or not the difference of the means of the corresponding two samples is significant, i.e. whether one algorithm is significantly more accurate than the other one. Assuming that the two random samples of sizes N_1 and N_2 are drawn form normal distributions whose

standard deviations are equal ($\sigma_1 = \sigma_2$) and that these two samples have means and standard deviations given by $\overline{X}_1$, $\overline{X}_2$ and s_1, s_2 respectively, the hypothesis H_0 that the samples come from the same population (i.e. $\mu_1 = \mu_2$ as well as $\sigma_1 = \sigma_2$) can be tested using the t score given by

$$t = \frac{\overline{X}_1 - \overline{X}_2}{\sigma \sqrt{1/N_1 + 1/N_2}} \qquad where \qquad \sigma = \sqrt{\frac{N_1^2 s_1^2 + N_2^2 s_2^2}{N_1 + N_2 - 2}}$$

The distribution of t is Student's distribution with $v = N_1 + N_2 - 2$ degrees of freedom. Here the sample sizes are determined by the number of data sets, i.e. $N_1 = N_2 = 6$. In order to decide between the hypotheses

H_0: $\mu_1 = \mu_2$, and the difference in the average accuracies is due to chance.

H_1: $\mu_1 > \mu_2$, and C4.5 decision trees (μ_1) are significantly more accurate than the other rule set type (μ_2).

on the basis of a one-tailed significance test at a 5% level of significance (= 95% level of confidence), H_0 can be rejected, if t computed as above is greater than the critical value $t_{.95}$, which for $v = N_1 + N_2 - 2 = 6 + 6 - 2 = 10$ degrees of freedom is $t_{.95} = 1.81$.

Table 29: Results of the pair-wise significance tests based on Student's *t* distribution.

Algorithms	$\overline{X}_1$	s_1	$\overline{X}_2$	s_2	t	Result
1: C4.5 decision tree **2: AQ15c ordered rules**	86.99	8.25	74.99	13.66	1.68	1.68 < 1.81 => H_0 cannot be rejected. **The sample means do not differ significantly.**
1: C4.5 decision tree **2: AQ15c unordered rules**	86.99	8.25	54.14	34.10	2.09	2.09 > 1.81 => H_0 can be rejected. **The sample means differ significantly.**
1: C4.5 decision tree **2: C4.5 ordered rules**	86.99	8.25	86.21	9.37	0.14	0.14 < 1.81 => H_0 cannot be rejected. **The sample means do not differ significantly.**
1: C4.5 decision tree **2: CN2 ordered rules**	86.99	8.25	84.87	10.82	0.35	0.35 < 1.81 => H_0 cannot be rejected. **The sample means do not differ significantly.**
1: C4.5 decision tree **2: CN2 unordered rules**	86.99	8.25	86.26	9.03	0.13	0.13 < 1.81 => H_0 cannot be rejected. **The sample means do not differ significantly.**
1: C4.5 decision tree **2: GIL unordered rules**	86.99	8.25	81.99	12.05	0.77	0.77 < 1.81 => H_0 cannot be rejected. **The sample means do not differ significantly.**

Results of the series of significance tests

The result of the comparison of the accuracy values of all rule set types with C4.5 decision trees are listed in Table 29. These results can be summarized to the statement, that C4.5 decision trees are significantly more accurate than the AQ15c unordered rules, but they are *not* significantly more accurate than *all* other rule set types, because the improvement of C4.5 decision trees over AQ15c ordered rules, C4.5 ordered rules, CN2 ordered rules, CN2 unordered rules, and GIL unordered rules is not significant. Hence the answer to the question posed in this appendix is that *C4.5 decision trees are not significantly more accurate than the other rule set types.* The improvement is only significant for one of the six other rule set types at the 5% significance level (= 95% confidence level)..

The sign test for testing the difference of two medians

The sign test is a *non-parametric or distribution-free method*, i.e. it does not require any knowledge about the distributions of the underlying basic variables. For non-parametric problems related to continuous variables, the *median* is a more natural measure of location for a distribution than the mean. The median has the desirable property that the probability is $\frac{1}{2}$ that a sample value will exceed the population median regardless of the nature of the distribution. The sign test can be used for testing the difference of two medians. Let f_1 and f_2 be the two continuous density functions under discussion and let $X_1, X_2, \ldots, X_n$ and $Y_1, Y_2, \ldots, Y_n$ represent samples of size n taken from f_1 and $_{f.}$, respectively. For the purpose of testing the hypothesis

$$H_0: \; f_1(x) = f_2(x)$$

the differences $X_i - Y_i$ for $I=1,2,\ldots,n$ are considered. When H_0 is true, X_i and Y_i constitute a random sample of size two from the same population. Since the probability that the first of two sample values will exceed the second is the same as the probability that the second will exceed the first and since, theoretically, the probability of a tie is zero, it follows that the probability that $X_i - Y_i$ will be positive is $\frac{1}{2}$. Thus, if only the signs of the differences are considered, a non-parametric test for H_0 can be constructed based on the variable

$$Z_i = 1, \quad if \quad X_i - Y_i \geq 0$$
$$Z_i = 0, \quad if \quad X_i - Y_i < 0$$

The variable Z_i is a binomial variable corresponding to a single trial of an experiment for which $p = \frac{1}{2}$. Since the Z_i are independent, their sum $U = \sum_{i=1}^{n} Z_i$ will be a binomial variable corresponding to n independent trials of an experiment for which $p = \frac{1}{2}$.

In order to use this fact for testing H_0, consider as an alternative hypothesis to H_0 the hypothesis

$$H_1: \ f_1(x) = f_2(x - c)$$

where c is some positive constant. H_1 states that the second density function is merely the first density function shifted to the left a distance of c units. Under H_1, the X_i will tend to be larger than the Y_i and the variable U will tend to exceed its expected value of $\frac{n}{2}$. One would therefore choose as critical region the right tail of the binomial distribution. If c was negative, the left tail would be chosen. If H_1 was the alternative that a translation of unknown direction had occurred, both tails would be used.

For great values of n, the binomial distribution can be well approximated by its normal approximation, but for very small values of n it is necessary to calculate the right tail probabilities until a total probability of approximately α, the desired significance level, has been obtained to obtain the critical region for the test. The right tail probability for given sample size n and probability p of a success in a single trial is computed as

$$P(X \leq j) = \sum_{k=0}^{j} P(X = k) \quad \text{where} \quad P(X = k) = \frac{n!}{k!(n-k)!} \, p^k \, q^{n-k} \quad \text{and} \quad q = 1 - p.$$

X denotes the number of successful trials in n trials, $P(X=k)$ is the probability of k successful trials in n trials, and $P(X \leq j)$ is the probability of j or less successful trials in n trials.

Results of the sign test

Table 30 lists the result of a pair-wise comparison of the predictive accuracy of C4.5 decision trees with all other rule set types. The results are based on the sign test described in the

Table 30: Results of the pair-wise comparison of C4.5 decision trees with the other rule set types based on the sign tests.

Algorithms	*U*	*P(X≤U)*	Result
AQ15c ordered rules	1	0.1094	H_0 cannot be rejected, i.e. the sample medians do not differ significantly.
AQ15c unordered rules	0	0.0156	H_0 can be rejected, i.e. the sample medians differ significantly.
C4.5 ordered rules	4	0.8906	H_0 cannot be rejected, i.e. the sample medians do not differ significantly.
CN2 ordered rules	3	0.6563	H_0 cannot be rejected, i.e. the sample medians do not differ significantly.
CN2 unordered rules	3	0.6563	H_0 cannot be rejected, i.e. the sample medians do not differ significantly.
GIL unordered rules	1	0.1094	H_0 cannot be rejected, i.e. the sample medians do not differ significantly.

previous section. The left tail critical value for a significance level of 5%, $U_{0.05}$, is between 0 and 1, i.e. for $U=0$ H_0 can be rejected, while for $U \geq 1$ H_0 cannot be rejected. According to the sign test, the C4.5 decision trees are only significantly more accurate than the unordered AQ15c rules at the 5% significance level (= 95% confidence level), but they are not significantly more accurate than ordered AQ15c rules, ordered C4.5 rules, ordered and unordered CN2 rules and unordered GIL rules**.**

VITA

Ralf Helmut Klinkenberg was born on May 30, 1971 in Troisdorf, Germany. He received his primary education at Gemeinschaftsgrundschule Troisdorf-Eschmar in Troisdorf-Eschmar, Germany, from 1978 to 1982 and his secondary education from Städtisches Gymnasium Troisdorf "Zum Altenforst", Troisdorf, Germany, from 1982 to 1988 and Städtisches Ruhr-Gymnasium, Witten, Germany from 1988 to 1991. He received the Vordiplom in computer science from University of Dortmund, Dortmund, Germany, in September 1993.

Since September 1993, Mr. Klinkenberg has been a graduate student at University of Dortmund and since August 1994 a graduate student at University of Missouri-Rolla in Rolla, Missouri.