Journal of Intelligent Information Systems, 5, 211–228 (1995)

Induction of Ripple-Down Rules Applied to Modeling Large Databases

B.R. GAINES gaines@cpsc.ucalgary.ca
Knowledge Science Institute, University of Calgary, Calgary, Alberta, Canada T2N 1N4

P. COMPTON
Department of Computer Science, University of New South Wales, Sydney 2033, Australia

Abstract. A methodology for the modeling of large data sets is described which results in rule sets having minimal inter-rule interactions, and being simply maintained. An algorithm for developing such rule sets automatically is described and its efficacy shown with standard test data sets. Comparative studies of manual and automatic modeling of a data set of some nine thousand five hundred cases are reported. A study is reported in which ten years of patient data have been modeled on a month by month basis to determine how well a diagnostic system developed by automated induction would have performed had it been in use throughout the project.

Keywords: inductive database modeling, induction, machine learning, medical diagnosis, ripple-down rules, rules with exceptions, Induct, Garvan thyroid database

1. Introduction

Knowledge-based systems with large structures of concepts and rules are now in routine use in a number of applications. However, the problems of developing and maintaining large rule-based systems is a major impediment to knowledge-based system application (Li, 1991). Knowledge acquisition for these systems as new circumstances arise is an ongoing requirement that increases in difficulty as the system grows due to interaction between rules.

One approach to this problem has been to design knowledge-base structures which minimize interactions between rules. For example, Compton & Jansen have developed a structure of rules with two forms of exception for diagnostic applications that is very simple to maintain (Compton and Jansen, 1990a). Their 'ripple-down' rules are such that a case can fall under only one rule, and an error may be corrected by adding one exception rule taking into account only cases that previously fell under the rule to be corrected. This techniques has been successfully applied to restructuring the Garvan ES-1 thyroid diagnosis system that has been in routine use since 1984 producing some six thousand interpretations a year (Horn et al., 1985). The diagnosis system was developed in use, and the current version uses some six hundred and fifty rules covering sixty diagnoses, but has become difficult to maintain. When reconstructed with ripple-down rules, the complex rule set was reduced to some five hundred and fifty simpler rules, and the effective rate at which rules can be developed has been increased from about two rules a day to ten rules an hour (Compton and Jansen, 1990b). A successful application to the development of a new expert system for chemical pathology at St Vincent's hospital has also been reported (Compton et al., 1991).

Another approach to the development and maintenance of large rule-based systems has been the use of inductive modeling techniques to analyze large databases and develop knowledge-based systems (Piatetsky-Shapiro and Frawley, 1991). With the increasing power of modern workstations and improvements in inductive techniques it is possible to develop inductive models of databases with tens of thousands of cases in a few minutes.

It is possible to combine both approaches by developing an algorithm for the induction of ripple-down rules. This allows expertise transfer and machine learning approaches to knowledge base development to be combined in a variety of ways. At one extreme, where the relevant attributes of a database are well-defined, automatic induction may be applied to both development and maintenance with little human interaction. At the other extreme, expert editing may be primarily used to the develop the knowledge base with automatic induction used to suggest rules. In practice, a changing strategy is attractive from exploratory manual development in the initial stages when the relevant attributes are not clear and data is scarce, to automatic induction in the later stages of system maintenance when relevant attributes are well-defined and a large database is available.

This paper describes an algorithm for the induction of ripple-down rules based on a very fast and effective statistical approach to empirical induction already described (Gaines, 1989). It reports comparative studies of manual and automatic development for the Garvan ES-1 data, and a reconstructive analysis of the performance of automatic induction had it been used over ten years of data. The paper commences with a description of ripple-down rule structures, their manual development, their automatic induction, and results on standard datasets in the literature.

2. Ripple-down rules

Compton and Jansen introduced the ripple-down rule technique as a methodology for the acquisition and maintenance of large rule-based systems (Compton and Jansen, 1990a; Compton and Jansen, 1990b). The underlying concept is that people cope with the acquisition and maintenance of complex knowledge structures by making incremental changes to them within a well-defined *context* such that the effect of changes is locally contained in a well-defined manner. Standard production rule systems do not have this property. The modularity of the rules themselves is not reflected in a modularity of the consequences of changes to those rules. Small changes can lead, through complex interactions, to major effects, making the development and maintenance of rule-based systems far more difficult than it appears at first sight.

The ripple-down rule technique creates a two-way dependency relation between rules such that rule activation is investigated only in the context of other rule activation. If the premise of a parent rule is true for a particular individual then, if has no dependents, its conclusion will be asserted for that individual. If, however, it has an 'if-true' dependent then that rule, and its dependents, will be tested, and the original conclusion will only be asserted if the premises of none of them are true for the entity. Conversely, if the premise of a parent rule is false for a particular individual, then not only will its conclusion not be asserted but also, if it has an 'if-false' dependent then it, and its dependents, will be tested.

Thus, ripple down rules form a binary decision tree that differs from standard decision trees (Breiman et al., 1984) in that compound clauses are used to determine branching, and these clauses need not exhaustively cover all cases so that it is possible for a decision to be reached at an interior node. This contrasts with standard trees where all decisions are made at root nodes. However, the feature of standard decision trees is retained that one, and only one, decision node is activated for each case. This makes maintenance simple because if the decision reach is erroneous then only node, and the past cases that have fallen under it, need be considered.

2.1. *Manual development of ripple-down rules*

To illustrate the manual development of ripple-down rules as a knowledge acquisition methodology, Fig. 1 shows six of the top level rules in Compton's reconstruction of the Garvan ES-1 diagnostic system as a set of ripple-down rules. There are 61 possible diagnostic combinations, labeled 00 to 60. The top level rule, 0.00, is an overall default with no conditions stating that if no other rule is activated the diagnosis is 00, no thyroid disorder. This rule is always true so it has only an 'if-true' link to rule 1.46 which gives four conditions leading to diagnosis 46. If these conditions are not met, the 'if-false' link to rule 2.32 is followed. If the conditions are met then diagnosis 46 is not immediately indicated. The 'if-true' link to rule 497.48 is first tested and if its conditions are satisfied its diagnosis of 48 takes precedence over 46.

Logically, ripple-down rules have exactly the same representational and deductive capabilities as standard production rules. Where they differ is in the strong context provided by the 'if-true' and 'if-false' links that guarantees that the effect of adding a rule is strictly contained in that it can affect the conclusions applied to cases previously falling under at most one other rule. A new sibling rule joined to a parent by an 'if-true' link can only change the conclusion applied to its entities falling under its parent, and no others. A new

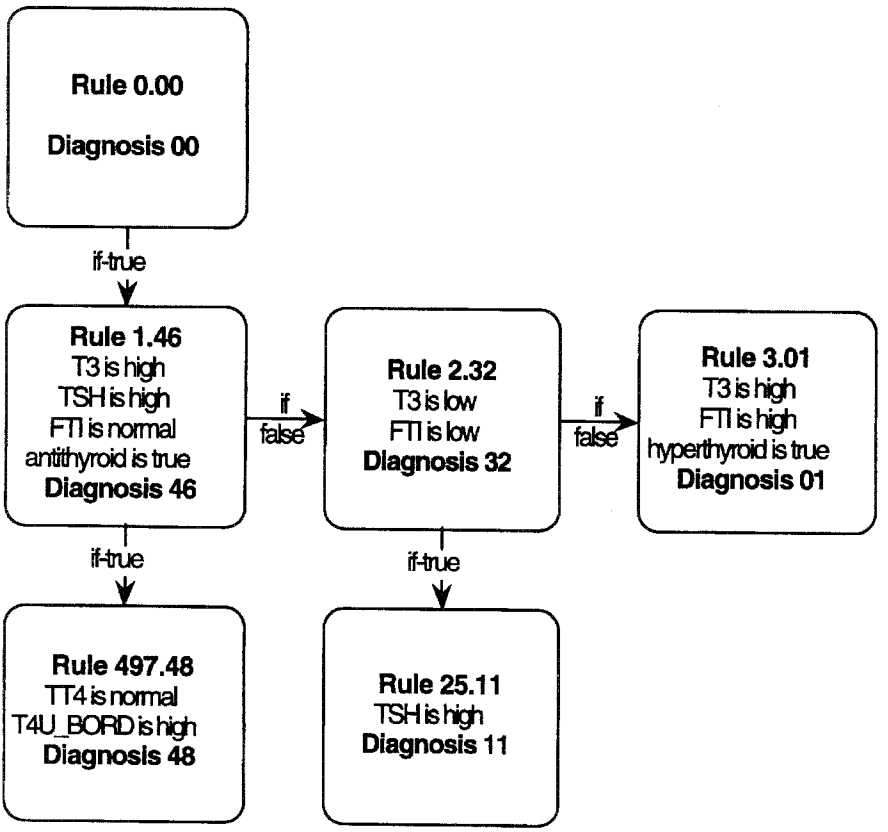

Figure 1. Some ripple-down rules.

sibling rule joined to a parent by an 'if-false' link can only change the conclusion applied to its entities falling under the rule with an 'if-true' link to the chain of which its parent is part, and no others. In general, a new rule can only affect cases falling under the rule responsible for the conclusion if the new rule were not present. This limits the comparisons and decisions required in knowledge acquisition and maintenance to the consideration of cases falling under only one rule.

For example, consider a situation in which a rule network has been built as shown in Fig. 1, and a new case is under consideration for which rule 0.00 succeeds, 1.46 fails, 2.32 succeeds, and 25.11 succeeds. The conclusion is diagnosis 11, but suppose the correct diagnosis is 15. Since 25.11 succeeds but its conclusion is wrong, it is necessary to add a new rule with an 'if-true' link to 25.11. How should its premise be specified? Figure 2 gives

Case Number	2	25	41	79
sex	female	female	female	female
on_t4	false	false	false	false
query_t4	false	false	false	false
antithyroid	false	false	false	false
sick	true	false	***true***	***true***
pregnant	false	false	false	false
i131	false	false	false	false
hypothyroid	false	false	false	**true**
hyperthyroid	false	false	false	false
lithium	false	false	false	false
tumour	false	false	false	false
goitre	false	false	false	false
hypopit	false	false	false	false
psych	false	false	false	false
antibs	false	false	false	false
surgery	false	false	false	false
treated	false	false	false	false
no_comment	true	true	false	false
screening	false	false	false	false
ovulatory	false	true	**false**	**true**
TSH	normal	*high*	*high*	***missing***
TSH_BORD	missing	missing	**high**	missing
T3	*low*	*low*	*low*	*low*
T3_BORD	low	missing	missing	**missing**
TT4	missing	missing	missing	missing
TT4_BORD	missing	missing	missing	missing
T4U	missing	missing	missing	missing
T4U_BORD	missing	missing	missing	missing
FTI	*low*	*low*	*low*	*low*
FTI_BORD	missing	missing	missing	**low**
TBG	missing	missing	missing	missing
TBG_BORD	missing	missing	missing	missing
diagnosis	32	11	15	14
RULE	2.32	25.11	41.15	79.14

Figure 2. Some cases relevant to the rules of Fig. 1.

the description of relevant cases available to the expert. Case 25 is the one case that has so far been diagnosed using rule 25.11, and case 41 is the new case. In this table, the values of attributes satisfying the premises of the relevant rules are italicized, and the values in which two comparative cases differ are in bold. In considering the premises for a new rule, one is aiming to create a rule such that case 41 is included so that it can be diagnosed as 15, but case 25 is excluded so that it remains diagnosed as 11.

Hence, the relevant differences are that *sick* is true for 41 and false for 25, *ovulatory* is false for 41 and true for 25, *TSH_BORD* is high for 41 and missing for 25. One could generate a very specific rule requiring all three differences to be present, or a more general rule requiring just one or two to be present. In this case the *sick* attribute was chosen as the differentiation resulting in the 'if-true' addition of rule 41.15 shown in Fig. 3.

Now consider case 79 for which rule 0.00 succeeds, 1.46 fails, 2.32 succeeds, and 25.11 fails. The conclusion is diagnosis 32 from the last successful rule, but suppose the correct diagnosis is 01. Since 2.32 succeeds but its conclusion is wrong, it is necessary to add a new rule with an 'if-false' link to 25.11. How should its premise be specified? In Fig. 2, Case 2 is the one case that has so far been diagnosed using rule 2.32, and case 79 is the new case. One is aiming to create one such that case 79 is included so that it can be diagnosed as 01, but case 2 is excluded so that it remains diagnosed as 32. Hence, the relevant differences are that *hypothyroid* is true for 79 and false for 2, *ovulatory* is true for 79 and false for 2,*TSH*

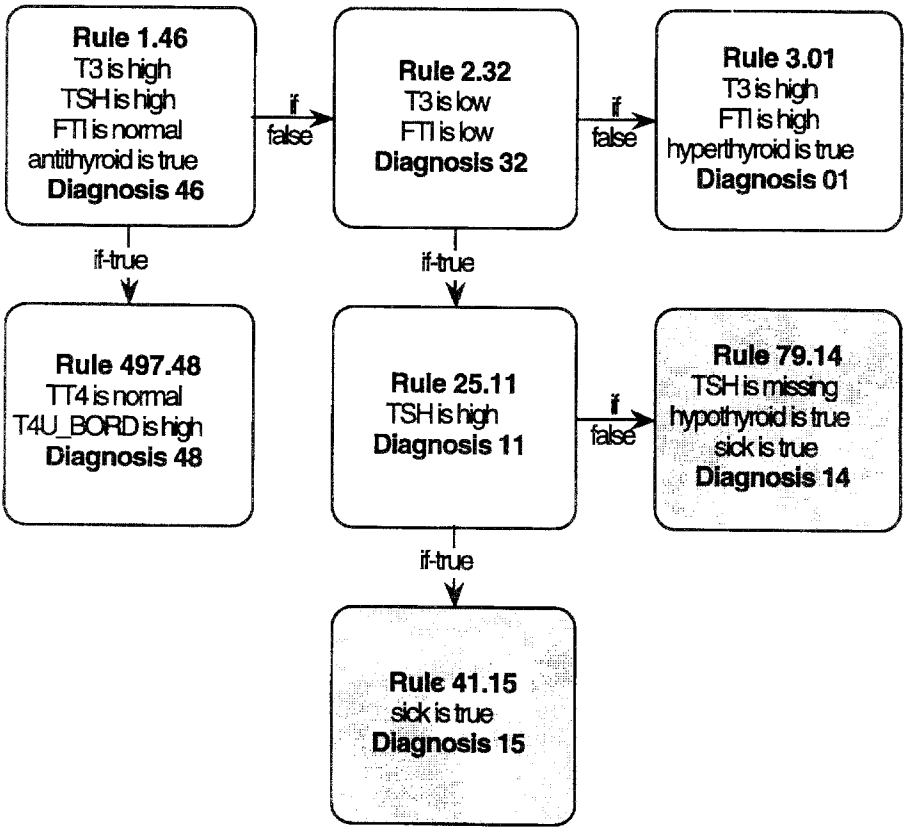

Figure 3. Ripple-down rules after incremental acquisition.

is missing for 79 and normal for 2, *T3_BORD* is missing for 79 and low for 2, *FTI_BORD* is low for 79 and missing for 2. In this case the *TSH*, *hypothyroid* attributes were chosen as the differentiation, and the *sick* attribute was also included resulting in the 'if-false' addition of rule 79.14 shown in Fig. 3.

This constrained but indeterminate choice of conditions for the premise of a new rule is the main feature of ripple down rule acquisition and maintenance. The expert has only to consider a well-defined set of cases for which conclusions have been drawn. The possible clauses for the premise of a new rule are constrained to cover the new case but not the previous cases. Within these constraints, the expert has the discretion to use other knowledge to design as specific or as general a rule as is deemed appropriate. Thus, the ripple down rule structure allows the computer to support the management of large rule bases while leaving the human designer the capability to use his or her knowledge effectively in a well-defined decision framework.

There are three extensions to the situation illustrated that require further discussion. First, the new cases may be identical in features to an old case but have a different diagnosis. The cases cannot be differentiated and the expert has to cope with the indeterminacy, perhaps by gathering more data if there are missing values in the new case, perhaps by accepting the indeterminacy and giving alternative diagnoses, possibly with probability estimates. The expert may also introduce new attributes that enable the cases to be differentiated. This incremental acquisition of relevant attrubutes is an important feature of manual ripple-down rule development that cannot be duplicated by automatic induction from a database with a fixed set of attributes.

Second, there are multiple cases falling under the existing rule, not just the single case so far exemplified. This causes no fundamental problems since, if the first situation does not apply, there will be at least one attribute differentiating the new case from any given previous case, and a combination of such differentiating attributes provides a rule differentiating it from all the previous cases. Hopefully, the situation simplifies, in that one attribute is sufficient to differentiate several previous cases, but even if it does not there is always a solution available.

Third, the expert may chose not to differentiate all previous cases in the multiple case situation. In this event an additional 'if-true' rule is required as a sibling of the new rule to cover the old cases that have been allowed to fall under the new rule. This would be done because the pair of rules generated in this way were perceived to be simpler than a single differentiating rule.

It is also possible in concept for the expert to edit the actual rule giving rise to the incorrect diagnosis rather than add a supplementary rule to it as an exception. This is what would be done with conventional production rule development. However, amending a rule would involve considering not only the cases that have fallen under that rule, but also those have fallen under the rule which leads to it as an exception. As more rules and cases have to be considered the task of the expert, and the scope for errors, increases. The essence of the simple procedure explained above is that only one rule and the cases falling under it has to be considered.

2.2. *Automatic induction of ripple-down rules*

The automatic induction of ripple-down rules is very simple using the statistical methodology for empirical induction of rules from entity-attribute-value data previously described

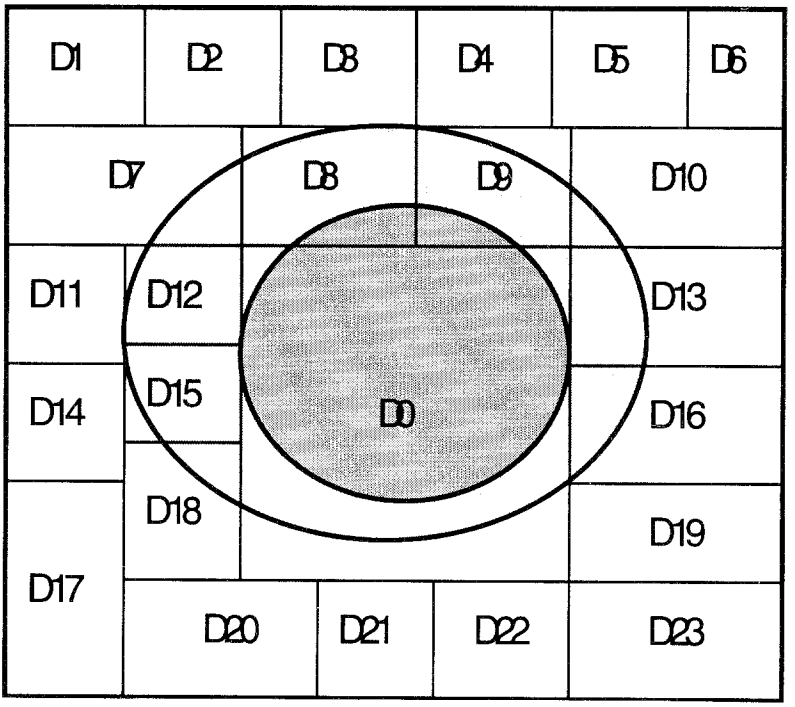

Figure 4.

for the Induct algorithm (Gaines, 1989). Induct is a system with performance similar to that of C4.5 (Quinlan, 1993) but deriving rules directly using an extension of Cendrowska's (1987) Prism algorithm to operate with noisy data. The system already derives rules with exceptions (Gaines, 1991b), and the extension to ripple-down rules was not only simple but also very natural in that the algorithm involves one very simple statistical decision procedure to generate a rule which is called recursively on residual data sets to generate 'if-true' and 'if-false' rules.

Figure 4 shows the error analysis that provides statistical control of the algorithm used in Induct. The space of cases in the part of the database under consideration is shown as a rectangle broken down into sub rectangles by diagnosis. The center rectangle with diagnosis D0 is that for which Induct is attempting to determine a rule. The cases covered by the premise of the currently proposed rule are indicated by the outer ellipse. The inner ellipse indicates the reduced set of cases covered when an additional clause is added to the premise. The topology of Fig. 4 is arbitrary but otherwise the information shown is precisely that available to the algorithm.

There are three steps in the generation of the premise. First, the most frequently occurring diagnosis in the part of the database under consideration is selected as the target conclusion. Second, a premise is initialized with no clauses. Third, iteratively, each available attribute-value combination is tested as a possible clause and the best selected according to a statistical test detailed below. Fourth, a decision is made using a similar test as to whether adding the selected clause improves the rule or not. If there is improvement the process iterates back to the third stage, and otherwise it terminates with the output of a rule if one has been generated.

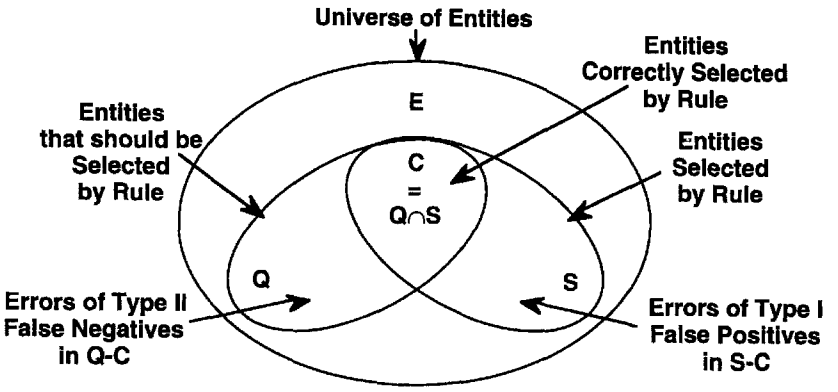

Figure 5. Error analysis for statistical control of empirical induction.

Figure 5 shows the basis of the statistical test used in Induct. The problem of empirical induction is, given a universe of entities, E, a target predicate, Q, and a set of possible test predicates of the form, S, on entities in E, to use them to construct a set of rules from which the target predicate may be inferred given the values of the test predicates. For the purposes of the statistical analysis the forms of S and Q do not matter. One may regard S as a *selector* choosing those e out of some subset of E for which to assert $Q(e)$, and compare the selection process of the rule with that of random selection, asking "what is the probability that random selection of the same degree of generality would achieve the same accuracy or greater."

This probability is easily calculated. Let Q be the relevant entities in E for which $Q(e)$ holds, S be the selected entities in E for which $S(e)$ holds, C be the correct entities in E for which both $S(e)$ and $Q(e)$ hold:

$$Q \equiv \{e: e \in E \wedge Q(e)\} \tag{1}$$

$$S \equiv \{e: e \in E \wedge S(e)\} \tag{2}$$

$$C \equiv \{e: e \in E \wedge S(e) \wedge Q(e)\} \tag{3}$$

Let the cardinalities of E, Q, S and C be e, q, s and c respectively. The probability of selecting an entity from E for which Q holds at random is:

$$p = q/e \tag{4}$$

The probability of selecting s and getting c or more correct at random is determined by summing the standard binomial distribution to get the incomplete beta function (Press, Flannery, Tekolsky and Vetterling, 1988, p. 182):

$$r = \sum_{i=c}^{s} \binom{s}{i} p^i (1-p)^{s-i} \tag{5}$$

The advantage of using r as a measure of the correctness of a rule is that is easily understood, as the probability that the rule could be this good at random, and that it involves no assumptions about the problem such as sampling distributions. It is interesting to

note that if $c = s$ (all correct) then $\log(r) = s\log(q/e)$ which seems to be the basis of 'information-theoretic' measures.

Missing values are taken into account by assuming that they might have any value. When the selection of an entity is tested a missing value is assumed to have the required value for selection. In the statistics a selection based on missing values is allowed to contribute to false positives but not to correct positives. This has important consequence in knowledge acquisition since it allows the expert to enter conjunctive rules as if they were entities with missing values. Induct then generates the same or an equivalent smaller rule set. It is reasonable to test the consistency of an inductive procedure by requiring the rule-set produced by it to be 'fixed-point' if re-entered as data. That is, if we treat a set of rules as if they were a set of cases with the unspecified attributes having the value "any", then any induction algorithm modeling this set of data should reproduce exactly the same rules that have been given it as data. If such a fixed-point is not achieved then what is the source of the additional information?

In terms of Fig. 4, a universe of entities, E, is the part of the database under consideration as shown, the target predicate, Q, is D0, and the selected entities, S, are those within the outer ellipse. The choice of the best clause at each stage is based on minimizing the probability that the results achieved by a premise could be achieved by random selection, the measure r in (5) as used in the original Induct. For ripple-down rules the interesting decision is whether to keep the more general rule corresponding to the outer ellipse, or add the additional clause corresponding to the inner ellipse. The outer ellipse covers more target cases having D0 as diagnosis, but also covers more D7, D8, D9, and so on, cases which will have to be treated as exceptions in the ripple-down rule structure. The decision is an interesting one because, given the capability to deal with exceptions, it does not necessarily affect the *accuracy* of the final knowledge base. Rather, it affects its *structure* in terms of preference for general rules with many exceptions rather than specific rules with few exceptions. It will affect quantitative measures of the knowledge base such as the numbers of rules and clauses involved, but it also affects its structure in a way that corresponds to *cognitive style* in the way a human expert might present the knowledge.

The algorithm currently used in Induct attempts to minimize the number of parameters involved in controlling induction by applying the same test and criterion to the relative improvement of a premise by adding an additional clause as it does to the absolute significance of that premise. That is, it determines the probability that the improvement to the rule by adding a clause might arise through chance, and does not add the clause if this is greater than the same significance level prescribed to determine whether the rule as a whole should be accepted. The empirical studies reported in the following sections show that this strategy is effective in generating minimal rule sets that are also reasonable knowledge structures. However, there is much scope for future research in determining what strategies best match human behavior in representing phenomena in the world through rules with exceptions.

The discussion so far has focused on generating a single rule from part of the database. The algorithm presented extends very simply to the generation of a complete ripple-down rule structure. Assume, in Fig. 4 that the premise corresponding to the outer ellipse has been selected to generate a rule. The algorithm then calls itself recursively twice: first, to generate 'if-true' exceptions on the part of the database within the ellipse that does not have the diagnosis D0; and, second, to generate 'if-false' exceptions on the part of the database outside the ellipse. For the 'if-true' case the diagnosis, D0, is passed on as the existing default diagnosis so that the next most frequently occurring diagnosis will be

chosen for rule generation. For the 'if-false' case the existing default diagnosis is similarly passed on.

The algorithm as described is sufficient to generate a complete ripple-down rule knowledge base from a database. There are some minor internal refinements to cope with known problems of empirical induction as described by Quinlan (1987) and Cendrowska (1987). For example, the lack of performance gain by 'exclusive-or' conditions is dealt with by proceeding with clause addition on a provisional basis even if there is no improvement, and rules generated are checked for pruning clause by clause in case the order of clause generation has led to redundant clauses. Overall, it is an extremely simple and efficient algorithm that is readily optimized for very high speed modeling of large datasets. The implementation used on the examples in the following sections is part of the knowledge acquisition system, KSSn (Gaines, 1991a), written as a class library in C++ (Gaines, 1994).

3. Results with some standard datasets

In presenting a new algorithm for empirical induction it is appropriate first to illustrate its performance on test datasets published by others. Three examples are presented here: Cendrowska's (1987) contact lens example; one of Quinlan's (1979) chess end-game examples also used by Cendrowska (1987); and Quinlan's (1987) prob-disj example. The first two cases allow the ripple-down-rule solutions to be compared with conventional decision trees and production rules for simple deterministic data, and the last case allows the comparison to be made for noisy data with irrelevant attributes. The results with the Garvan datasets in the next section illustrate the performance with large real-world datasets.

Cendrowska's (1987) contact lens data provides a simple example involving 24 cases with three 2-valued attributes, one 3-valued attribute, and a 3-valued conclusion. Figure 6 is a screen dump of Induct running on this data. The rear window at the top is that for Induct itself with various parameter settings in the upper part, and the results of evaluation in the lower part. The front window at the bottom of Fig. 6 is a graph of the ripple-down rules produced automatically by Induct. The "if-true" links are represented by simple arrows, and the "if-false" links by arrows with a bar across. The system is fully interactive in that the expert is able to edit the rules in the graphic browser if they wish, and copy the results back to Induct for evaluation. This is particularly useful in exploring the role of specific rules and experimenting with alternatives, but such post-editing was not used in the examples given in this article.

The ID3 tree representing the contact lens data has 15 nodes. The normal 9-rule solution for the contact lens data is shown at the top of Fig. 7, and may be compared with the 5-rule ripple-down rule solution at the bottom. The 25 clauses in the original solution have been reduced to 8 and the knowledge structure is simpler and easier to understand.

The fact that the ripple-down rule solution results in identical decisions to the production rule solution may be checked from Fig. 7 by treating each production rule as if it were a case with missing, or 'don't care', values. One can then trace the way in which a decision will be reached for each of the 9 such 'cases' corresponding to the production rules. For example, the rule "tear production reduced $\rightarrow$ lens = none" corresponds to a case that cannot proceed beyond the initial default ripple-down rule, "$\rightarrow$ lens = none", because the "if-true" branch and its attached "if-false" branches all require "tear production = normal".

A somewhat more complex deterministic example is one of Quinlan's (1979) chess datasets that Cendrowska (1987) also analyzed with Prism. The data consists of 647 cases

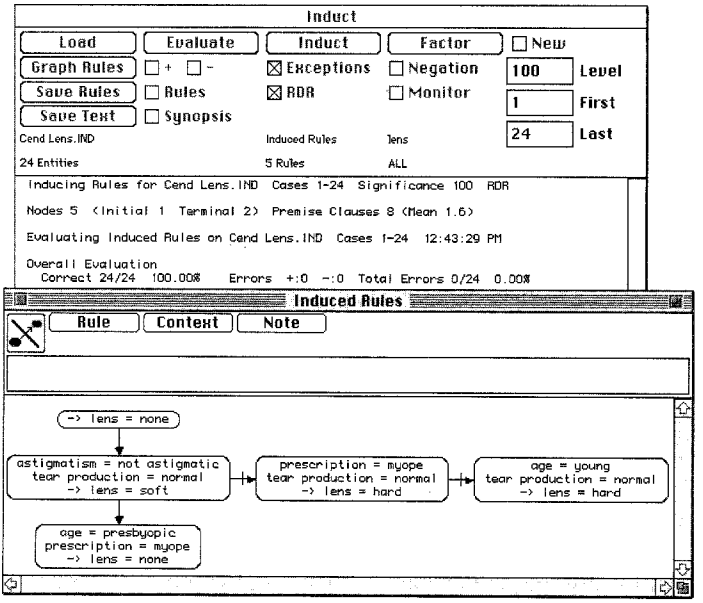

Figure 6. Induct inducing, evaluating and graphing rules for contact lens dataset.

```
prescription = myope & astigmatism = astigmatic & tear production = normal -> lens = hard
age = young & astigmatism = astigmatic & tear production = normal -> lens = hard
prescription = hypermetrope & astigmatism = not astigmatic & tear production = normal -> lens = soft
age = young & astigmatism = not astigmatic & tear production = normal -> lens = soft
age = pre-presbyopic & astigmatism = not astigmatic & tear production = normal -> lens = soft
tear production = reduced -> lens = none
age = presbyopic & prescription = myope & astigmatism = not astigmatic -> lens = none
age = pre-presbyopic & prescription = hypermetrope & astigmatism = astigmatic -> lens = none
age = presbyopic & prescription = hypermetrope & astigmatism = astigmatic -> lens = none
```

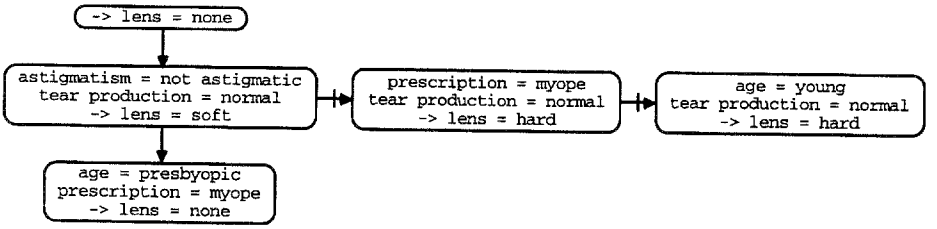

Figure 7. Normal and ripple-down rules induced for contact lens dataset.

of a rook versus knight end game situation described in terms of four 3-valued and three 2-valued attributes leading to one of two conclusions. The ID3 tree for this data has 30 nodes. Prism's 15-rule solution is shown at the top of Fig. 8, and at the bottom is the 9-rule ripple-down rule solution produced by Induct. The 48 clauses in the original solution have been reduced to 16 and the knowledge structure is again simpler and easier to understand.

The contact lens and chess examples have noise-free complete datasets, and ID3, Prism and Induct all generate 100% correct solutions. Induct was designed to cope well with noisy data and missing and irrelevant attributes (Gaines, 1989) and it is reasonable to expect the

```
e = f -> x = safe
f = f -> x = safe
g = f -> x = safe
b = 1 & d = 2 -> x = safe
b = 1 & d = 3 -> x = safe
a = 1 & c = 2 -> x = safe
a = 2 & c = 2 -> x = safe
a = 1 & c = 3 -> x = safe
a = 2 & c = 3 -> x = safe
a = 3 & b = 2 & e = t & f = t & g = t -> x = lost
b = 3 & c = 1 & e = t & f = t & g = t -> x = lost
a = 3 & b = 3 & e = t & f = t & g = t -> x = lost
b = 2 & c = 1 & e = t & f = t & g = t -> x = lost
a = 3 & b = 1 & d = 1 & e = t & f = t & g = t -> x = lost
a = 2 & b = 1 & c = 1 & d = 1 & e = t & f = t & g = t -> x = lost
```

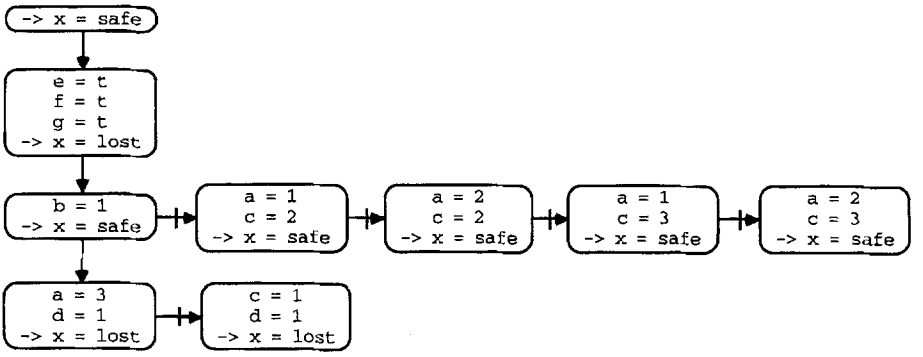

Figure 8. Normal and ripple-down rules induced for chess end-game dataset.

version that generates ripple down rules to continue to do so. A good test is Quinlan's (1987) prob-disj dataset which involves a 3-way disjunction of 3-attribute conjunctions over 9 binary attributes, with 1 additional irrelevant attribute and 10% errors in the data. This is a difficult problem for ID3 which generates an 187 node tree to capture the disjunction through replication of clauses. Figure 9 is a screen dump of Induct running on the original data and on the two test datasets provided. The significance level for r in Eq. (5) has been set to 1% so that the noise is filtered out by requiring a rule to have 1 chance in 100 of arising by chance. Induct achieves the best possible error rate of 10% on the test data, and generates the simple 4 rule solution shown which exactly captures the generating principle behind the data.

Thus, as previously reported, the Induct direct rule generation strategy is effective in performance and generates knowledge structures that are at least as good as the best previously reported for the test data analyzed. The ripple-down rule results reported here show that the knowledge structures of rules with exceptions are substantially smaller in most cases than conventional trees or production rules. In addition they satisfy the criterion of minimal interaction for ease of maintenance discussed above.

4. Results with some large medical datasets

Data from the Garvan ES-1 thyroid disorder diagnosis system have been used extensively in studies of empirical induction over the years (Horn et al., 1985). In the past few years a set

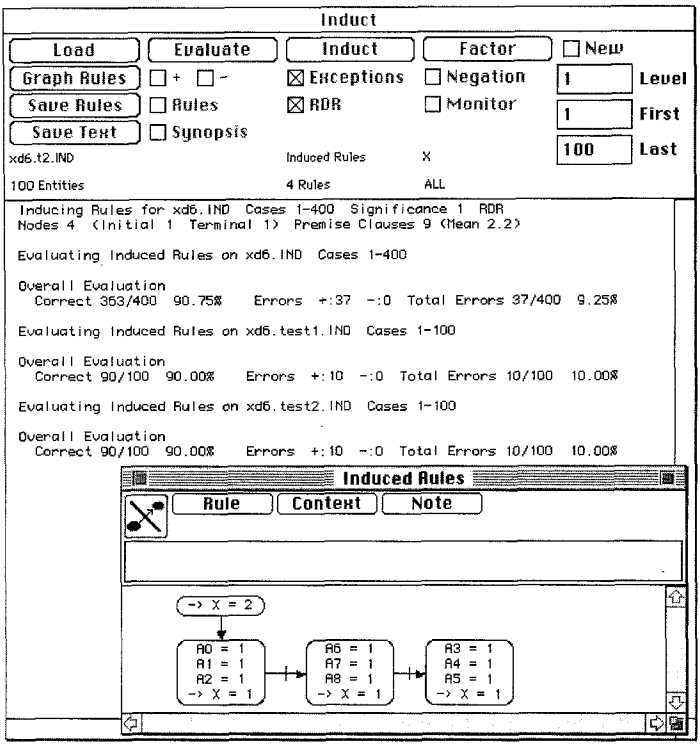

Figure 9. Ripple-down rules induced for prob-disj noisy, triple exclusive-or dataset.

of 9,514 cases has been used in a number of studies comparing manual ripple-down rules with the original rule set and with automatically induced rules. For example, Mansuri et al., (1991) split the dataset into 8,000 sample cases and 1,514 test cases, and report 550 rules, 2.8% errors for hand generated ripple down rules; and 502 rules, 2.71% errors for ID3. When Induct is run on the first 8000 cases and tested on the complete dataset it generates 195 ripple-down rules and has an error rate of 2.31% on the remaining test cases which is similar to the manual and ID3 results.

When Induct is run on all 9,514 cases, it generates 174 rules in 154 seconds accounting perfectly for the dataset. This compares favorably with the 550 ripple down rules generated manually in the reconstruction of the Garvan ES-1 using the same data. For ripple-down rules, a better measure of the complexity of an induced rule set is the total clauses involved since, as noted above, the structure may be varied without affecting the accuracy. The total clause count was 731 for the manual rules and 478 for the induced rules. The reduction is less apparent because the manual rules had only 1.3 clauses per rule on average compared with 2.5 clauses per rule for the induced rules. This seems to indicate that the expert tended to make the minimum differentiation necessary when adding a new rule.

It is also interesting to examine these results in the context of the usual approximation-complexity trade-off curve for an inductive process (Gaines, 1977). The lower plot in Fig. 10 was generated by varying the threshold used by Induct to determine whether the improvement achieved by adding a clause is due to chance from 5% to 100% in steps of

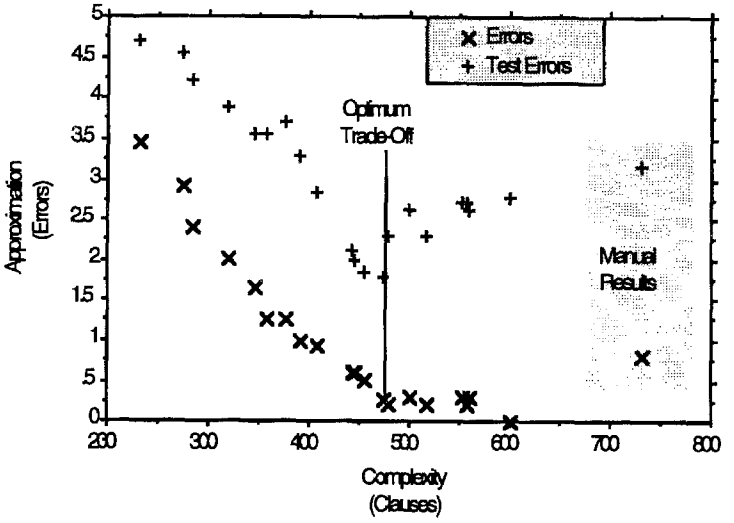

Figure 10. Complexity/approximation trade-off in modeling Garvan dataset.

2.5%. This generates sets of rules modeling the data that increase in complexity (number of clauses) and improve in approximation (error rate on data used in induction). The optimum trade-off was reached with a threshold of 37.5% corresponding to a 0.375 probability that improvements were due to chance. The upper plot shows the percentage errors for the same rule sets on the test data. It goes to a minimum close to that of the lower set indicating the sample for induction and the test data are drawn from the same population. The manual results plotted on the right can now be seen in the context of the complete trade-off curve.

A dataset of 43,472 cases is also available spanning the period from late 1979 to early 1990. This makes it possible to investigate the intrinsic predictability of the Garvan data, and to reconstruct what would have happened had Induct been available through the project. The 9,514 cases used previously as a test set are embedded in the new data but, due to an earlier conversion, they do not contain age information and hence are not a true subset. As already notes, the various models of the first 8,000 cases from the previous subset reported above when used on the residual 1,514 cases result in error rates of some 2.5%. This figure is important because it gives an estimate of the predictability of the Garvan data, that is not how well the model fits the data on which it was based, but how well the model predicts the following data on which it may be used as a diagnostic tool. Inductive modeling techniques are effective in so far as the database used is representative of future data, and this is dependent both on the size of the dataset available and the whether the data is stationary. Unfortunately, longitudinal studies in which these phenomena are investigated are very rare, and the availability of the large dataset over ten years provides an interesting opportunity to examine the predictability of a significant real-world medical database.

As a first experiment the database of 43,472 cases was split into eight successive datasets of 5,434 cases each, and Induct was run on the entire dataset and on the eight subsets. The result rules were then evaluated on every dataset to give the table shown in Fig. 11 in which the rows represent the evaluation of one set of rules on the full dataset and the eight successive subsets. The errors in the first row are exceptionally low because they are

	43472	1:5434	2:5434	3:5434	4:5434	5:5434	6:5434	7:5434	8:5434
43472	**0.46**	0.26	0.42	1.31	0.52	0.52	0.17	0.26	0.24
1:5434	9.38	**0.52**	4.34	6.81	10.38	14.13	13.75	12.68	12.27
2:5434	9.30	5.98	**0.61**	5.72	9.40	13.75	14.32	12.94	11.72
3:5434	9.87	14.08	5.13	**1.56**	8.15	12.86	12.75	12.62	11.80
4:5434	7.08	12.16	9.92	6.04	**0.77**	6.26	7.53	7.29	6.70
5:5434	10.96	21.16	26.13	19.80	7.99	**0.57**	3.85	4.32	3.86
6:5434	11.43	21.62	26.17	21.51	9.79	4.62	**0.52**	3.86	3.39
7:5434	12.37	22.23	30.31	21.77	10.20	5.65	4.38	**0.66**	3.79
8:5434	11.43	19.67	25.97	20.67	9.27	5.70	4.77	4.16	**1.21**

Figure 11. Analysis of Garvan data over ten years in eight successive datasets. (Each row shows the percentage of errors on rules developed on the datasets indicated in the left column on the complete dataset and the eight subsets).

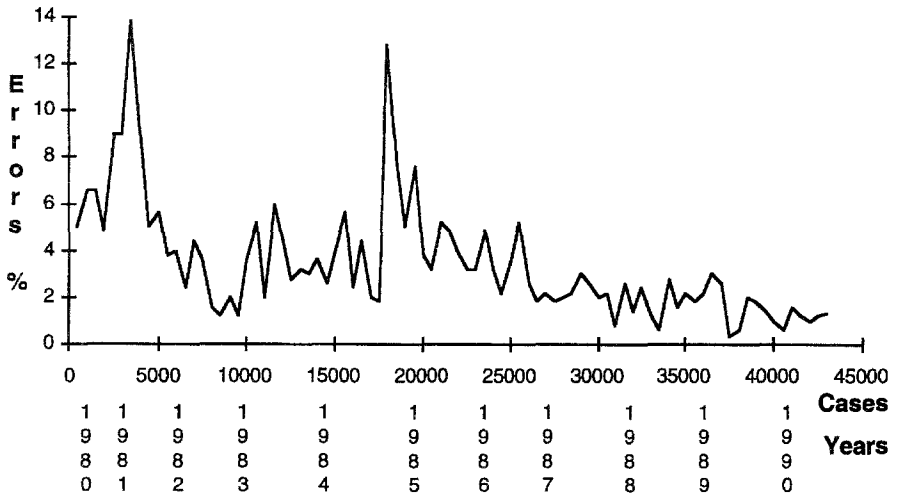

Figure 12. Error rates on next cases form models of Garvan database up to a certain time.

those of a model based on the complete data, and those along the diagonal are low because they are evaluated on the dataset on which the model is based. What is of primary interest is the performance of a model on the dataset to the immediate right of the diagonal since this indicates how well the model predicts the coming data. This error rate ranges through 4.34%, 5.72%, 8.15%, 6.26%, 3.85%, 3.86% and 3.79%. It also appears as one follows the error rate to the right that it tends to deteriorate substantially indicating that the database is not stationary, and hence that maintenance of the rule base to track the changing data is important.

To attempt to gain some insight into the dynamics of the Garvan data, and the way that this would have been tracked if an inductive model had been developed on a continuous basis, another analysis was undertaken in which the complete dataset was modeled for the first 500 cases, the first 1,000 cases, the first 1,500 cases, and so on, up to 43,000 cases. Each model was then tested on the next 500 cases (472 for the last model) and the error rate plotted as shown in Fig. 12. It can be seen that the 1979–80 data did not predict for 1981 very well. Then an improving model develops down a low error rate in late 1982

followed by a period of higher errors in 1983–84 with a major change in late 1984 followed by a steady improvement from 1985 through 1990. The major unpredictability in 1984 corresponds to changes in procedures in the hospital introduced at the start of the expert system development.

The table of Fig. 11 and the graph of Fig. 12 illustrate both the merits and the problems of the development of knowledge bases through the modeling of databases. A manually or mechanically derived model can be a valuable tool in supporting the diagnostic process. The recent history in Fig. 12 shows a period of error rates of less than 2% so that the system is providing a correct diagnosis automatically in 49 cases out of 50. However, there are much higher error rates at various times in the past history of the system, which are due to intrinsic unpredictability in the data, not to any defects of the modeling process. These indicate not only a need for continuous knowledge base maintenance, but also for effective system management in understanding and managing the impact of changes in procedures which, while they improve predictability in the long term, may greatly reduce it in the short term.

5. Conclusions

This paper has presented a knowledge representation schema based on rules with two forms of exceptions that is easy to understand and maintain manually. It has presented an empirical induction algorithm for modeling databases in terms of this schema that is simple, fast and effective, and has illustrated its application to standard datasets from the literature. It has used the schema and the algorithm to analyze a large clinical database, and shown how the model tracks non-stationarity in the database.

One conclusion that may be drawn from this paper is that in the clinical domain analyzed the automated induction process is at least as effective as knowledge transfer from experts. However, this does not imply that the support of transfer from human experts is not still important in this domain. It is reasonable to expect induction of ripple down rules to be substantially better than hand generation if we compare the circumstances of the human expert and the Induct algorithm. The expert and the algorithm are doing exactly the same thing at each node of the ripple down rule tree—that is adding additional rules to improve the performance. However, the expert is doing this in response to one erroneous case whereas the algorithm has available to it a complete analysis of the statistics of all relevant cases.

However, the decrease in the size of the rule set by no means detracts from the value of the manual acquisition of ripple down rules. At the start of a knowledge-based system development there may not be enough data available for induction, but there may be adequate human expertise available to build a working system. Later, when that system has been in use for some time, ripple down induction may be used to reduce the rule base without changing the basic procedures for use and maintenance to which those responsible for the system have become accustomed.

There are also more subtle reasons why manual acquisition remains very important. These show up well in the recent St Vincent's studies (Compton et al., 1991). The expert in entering new ripple down rules also enters new attributes. That is, the conceptual framework being applied to the cases is elaborated as part of the ripple down rule knowledge acquisition process. The contextual constraints already described continue to apply so that the use of new concepts can be managed locally and does not involve the expert in reanalysis of other

parts of the rule tree. Some of the new concepts entered fall completely outside those already entered. Others are intervals on existing numeric data. The latter could be induced with sufficient data but the former could not. Manual ripple down rule elicitation is an effective acquisition technique for conceptual schema.

In a practical system, the best features of the manual and inductive approaches can be combined. The evaluation of a new rule set has been designed to be fast enough for interactive use. It takes only a few seconds to evaluate a complete model on the Garvan cases. An expert can make arbitrary changes to the ripple down rule structure and have a complete account of the effects rapidly enough to maintain interactive development. In addition, the Induct algorithm may be used incrementally to advise the expert of an appropriate rule, which may then be accepted or manually amended. It takes less than one second to generate a new rule from the Garvan cases. Such a mixed initiative system allows the known trade-offs between knowledge acquisition techniques (Gaines, 1991b) to be optimized through a combination of expertise transfer, machine learning, visualization and rapid evaluation.

Acknowledgments

Financial assistance for this work has been made available by the Natural Sciences and Engineering Research Council of Canada and by the Australian Research Council.

References

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

Cendrowska, J. (1987). An algorithm for inducing modular rules. *International Journal of Man-Machine Studies*, 27(4), 349–370.

Compton, P., Edwards, G., Kang, B., Malor, R., Menzies, T., Preston, P., Srinivasan, A., and Sammut, S. (1991). Ripple down rules: possibilities and limitations. J.H. Boose and B.R. Gaines (Eds.), *Proceedings of the Sixth AAAI Knowledge Acquisition for Knowledge-Based Systems Workshop* (pp. 6-1–6-20). Calgary, Canada, University of Calgary.

Compton, P. and Jansen, R. (1990a). Knowledge in context: a strategy for expert system maintenance. C.J. Barter and M.J. Brooks (Eds.), *AI'88: 2nd Australian Joint Artificial Intelligence Conference, Adelaide Australia, November 1988, Proceedings* (pp. 292–306). Berlin, Springer.

Compton, P. and Jansen, R. (1990b). A philosophical basis for knowledge acquisition. *Knowledge Acquisition*, 2(3), 241–258.

Gaines, B.R. (1977). System identification, approximation and complexity. *International Journal of General Systems*, 2(3), 241–258.

Gaines, B.R. (1989). An ounce of knowledge is worth a ton of data: quantitative studies of the trade-off between expertise and data based on statistically well-founded empirical induction. *Proceedings of the Sixth International Workshop on Machine Learning* (pp. 156–159). San Mateo, California, Morgan Kaufmann.

Gaines, B.R. (1991a). Empirical investigations of knowledge representation servers: Design issues and applications experience with KRS. *ACM SIGART Bulletin*, 2(3), 45–56.

Gaines, B.R. (1991b). The tradeoff between knowledge and data in data acquisition. G. Piatetsky-Shapiro and W. Frawley (Ed.), *Knowledge Discovery in Databases* (pp. 491–505). Cambridge, Massachusetts, AAAI/MIT Press.

Gaines, B.R. (1994). Class library implementation of an open architecture knowledge support system. *International Journal Human-Computer Studies*, 41(1/2), 59–107.

Horn, P.J., Compton, P.J., Lazarus, L., and Quinlan, J.R. (1985). An expert system for the interpretation of thyroid assays in a clinical laboratory. *Australian Computer Journal*, 17, 7–11.

Li, X. (1991). What's so bad about rule-based programming? *IEEE Software*, 8(5), 103–105.

Mansuri, Y., Kim, J.G., Compton, P., and Sammut, C. (1991). A comparison of a manual knowledge acquisition method and an inductive learning method. *Proceedings of the First Australian Workshop on Knowledge Acquisition for Knowledge-Based Systems* (pp. 114–132). Sydney, University of Sydney.

Piatetsky-Shapiro, G. and Frawley, W. (Ed.) (1991). *Knowledge Discovery in Databases*. Cambridge, Massachusetts, MIT Press.

Press, W.H., Flannery, B.P., Teukolsky, S.A., and Vetterling, W.T. (1988). *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge, UK, Cambridge University Press.

Quinlan, J.R. (1979). Discovering rules by induction from large collections of examples. D. Michie (Ed.), *Expert Systems in the Micro Electronic Age* (pp. 168–201). Edinburgh, Edinburgh University Press.

Quinlan, J.R. (1987). Simplifying decision trees. *International Journal of Man-Machine Studies*, 27(3), 221–234.

Quinlan, J.R. (Ed.) (1993). *C4.5: Programs for Machine Learning*. San Mateo, California Morgan-Kaufman.