Pre-publication draft of a paper which appeared in The Proceedings of the Second Australian Joint Artificial Intelligence Conference, Adelaide, 1988, Springer pp 225 - 239

# TECHNIQUES FOR EFFICIENT EMPIRICAL INDUCTION

Geoffrey I. Webb

*Division of Computing and Mathematics,*
*Deakin University, Victoria, 3217*

**Abstract**

This paper describes the LEI algorithm for empirical induction. The LEI algorithm provides efficient empirical induction for discrete attribute value data. It derives a classification procedure in the form of a set of predicate logic classification rules. This contrasts with the only other efficient approach to exhaustive empirical induction, the derivatives of the CLS algorithm, which present their classification procedures in the form of a decision tree. The LEI algorithm will always find the simplest non-disjunctive rule that correctly classifies all examples of a single class where such a rule exists.

## 1 Introduction

Empirical induction is the discovery of classification rules from examples. An empirical induction algorithm takes as input a set of example instances (the *example set)* and outputs a classification procedure. The aim is to produce a classification procedure that can correctly classify novel instances.

A classification procedure will generally be represented herein by a data structure (such as a decision tree or a set of classification rules). The reader should remember, however, that the complete classification procedure consists of that data structure and a method for applying that data structure to an instance to produce a classification.

A number of approaches to automated empirical induction have been developed. These can be divided into two major categories - *statistic based* and *logic based.*

The statistic based approaches have evolved from CLS (Hunt, Mann and Stone, 1966). ID3 (Quinlan, 1986) is a more recent version of this approach. These algorithms utilise *attribute-value* data for classification. That is, relevant details of instances are expressed solely in terms of values of specified attributes of the instances. Although the initial approaches only supported *binary classification* (classifying instances as either belonging or not belonging to a single class) more recent derivatives of the algorithm support classification into multiple mutually exclusive classes.

The statistical approaches express their classification procedures in the form of decision trees. The root decision node is formed by using a statistical test to determine which attribute best differentiates between classes. The instances are then partitioned on different values for that attribute and the appropriate sets of instances associated with each branch from the decision node. This process is repeated for the terminal branches of the tree until the set of instances associated with each terminal branch contains only instances of a single class. The node descending from each terminal branch is then labelled with the class of the instances that are associated therewith. The internal nodes of the decision tree now represent classification tests. The terminal nodes represent the classification to assign to any instance passing all tests that lead to that node.

By using windows to restrict the instances examined to a small subset of those available, the ID3 algorithm is able to provide very efficient empirical induction.

However, decision trees are difficult to read (Quinlan, 1987*b*). Further, they tend to include superfluous complexity (Quinlan, 1987*a*). As a consequence, recent research has developed methods for taking the decision trees developed by the ID3 algorithm and using them to develop logical classification rules (Quinlan, 1987*a*). This has been found to produce classification procedures that are both more comprehensible and more accurate.

By contrast to the statistical approaches, previous logic based approaches have not provided efficient implementations, except at the cost of the use of heuristics that severely limit the potential classification procedures that they can discover.

The logic based approaches use various predicate logic based notations to represent relevant details of the instances to be classified. The classification procedure is usually represented by a body of logical implications. An implication has the form **if** *condition* **then** *conclusion.* The condition is called an *antecedent* and the conclusion is called a *consequent.* The implications used in a classification system have consequents that represent the desired classifications and antecedents that specify the characteristics of the instances to which those classifications apply.

Logical implications provide a far richer and more expressive description language than the attribute-value analyses supported by the statistic based approaches.

One logic based method is to form a single implication as a hypothesis classification rule. This rule is then systematically varied to account for the body of available examples. Two basic manipulations may be performed - *generalisation* and *specialisation.* Generalisation involves weakening the antecedent so that the consequent (classification) will apply to more instances. Specialisation involves strengthening the antecedent so that the consequent will apply to fewer instances. Note that specialisations of a false implication may be true whereas specialisations of a true implication must be true. By contrast, generalisations of a true implication may be false and generalisations of a false implication must be false.

One approach to this method (see, for example, Winston, 1975) involves setting the antecedent of the initial hypothesis rule to a description of the first example instance encountered. Further example instances are then examined. Positive examples (instances of the class in question) are used to generalise the current hypothesis. Negative examples (instances that do not belong to the relevant class) are used to specialise the hypothesis. Thus, the system starts from a very specific hypothesis and works toward a more general hypothesis that is consistent with the available evidence.

By contrast, another approach (see, for example, Buchanan and Feigenbaum 1978) starts with the most general hypothesis, that everything belongs to the class in question, and utilises the example set to work toward a more specialised classification rule.

The version space approach, developed by Mitchell (1977), maintains a range of candidate rules delimited by a most general and a most specific form. Counter examples are used to specialise the most general form and positive examples are used to generalise the most specific form until they both converge.

The $A^q$ algorithm (Michalski, 1977) differs from the above logic based approaches in that instead of creating a single classification rule it creates a set of classification rules that classify instances from the domain. Further, instead of only supporting binary classification, $A^q$ may form rules for classifying instances into any number of classes.

$A^q$ finds a set of rules that correctly classifies all instances in the example set. It iterates through the following sequence of events until all instances in the example set are correctly classified.

1. Select a random example, *E*, from the instances in the example set that are not yet classified by a potential classification rule.
2. Generate a star for *E*. A star is a set of generalisations of a description of an example that are true with respect to the example set.
3. Add the star to the set of potential classification rules.

When no instance remains in the example set that is not classified by a potential rule, the set of potential rules must fully correctly classify all instances in the example set. As there is usually considerable redundancy in the resulting set of rules, a subset is selected that still correctly classifies all examples while maximising some function that evaluates the desirability of a set of classification rules.

Different variants of $A^q$ use different heuristics to search for and decide between potential members of this star. Most versions form the star by specialising from candidate rules formed from single attributes (see for example Michalski, 1980.) By contrast, one version, INDUCE1.2 (Dietterich and Michalski, 1981), forms the star by generalising from a candidate rule formed directly from the instance being examined.

In order to constrain the search space most variants consider only a small number of the generalisations of an example that are true with respect to the example set. As a result there is a corresponding risk that important generalisations will not be considered and consequently that the best set of classification rules will not be developed.

Several distinctions between the logic based approaches mentioned above are worth highlighting. Some start from specific hypotheses and generalise while others start from general hypotheses and specialise. A method may provide binary classification (either instances belong to a class or do not) or n-ary (instances may belong to any one of several disjoint classes.) Some provide a single rule for each class while others develop collections of classification rules.

All of the above approaches suffer from deficiencies. As already mentioned, the logic based approaches are not as efficient as the statistic based approaches. However, the decision trees that are produced as classification procedures by the statistic based approaches are often difficult to comprehend (Quinlan, 1987*b*). Further, they are not guaranteed to correspond to any specified measure of the best available classification for the example set.

This paper describes a variant of the $A^q$ algorithm that provides efficient logic based empirical induction. The LEI algorithm differs from other variants of the $A^q$ algorithm by conducting an exhaustive search that finds all potential members of the star. This is made efficient by a number of heuristics that constrain the search space without excluding any possible members of the star. Where a class can be described by a single rule, LEI will always find the simplest such rule. Where multiple rules are required for a class, a set of rules that is close to the best possible will always be selected although it is not possible to guarantee that the best such set will always be selected.

*red(a) & glossy(a) & round(a) & large(a) & positive(a)*

*red(b) & matte(b) & square(b) & large(b) & positive(b)*
*red(c) & matte(c) & triangular(c) & small(c) & positive(c)*
*blue(d) & glossy(d) & round(d) & small(d) & negative(d)*
*blue(e) & glossy(e) & square(e) & small(e) & negative(e)*

Note: *a, b,* c, *d* and e represent the names of individual examples

**Figure 1: A small example set**

## 2 Objectives

Induction is necessarily based on the assumption that the example set is representative of the population as a whole. That is, it must be assumed that it is possible to derive from the available instances a classification procedure that will correctly classify novel instances.

As the aim is to be able to classify novel instances, it is not satisfactory to simply derive a classification procedure that will correctly classify all of the example instances. Trivially, any classification procedure that simply associates the correct class with each of the example instances will correctly classify all such instances. For example, the example set in Figure 1 will be correctly classified into the two classes *positive* and *negative* by *positive(a), positive(b), positive(c), negative(d), negative(e).* However, this classification information will not classify any novel instances as their classes will not have been specified.

It can be seen from this that it is necessary for an induction system to make generalisations in the light of the available evidence. The more generalised the classification procedure the more instances that it will be able to classify. It also follows, that the more generalised the classification procedure the more likelihood of it incorrectly classifying an instance. Thus, the aim is to generalise just enough to correctly categorise any instance not in the example set without mis-classifying any such instance. As instances outside the example set are not available to the induction system, no a priori method can identify the correct level of generalisation.

The LEI algorithm is based on a. number of observations. First, the logic based approaches provide more comprehensible and efficient classification procedures. Witness the recent development of algorithms for converting decision trees to sets of logical implications.

Second, it is possible to form a classification rule by treating all known attributes of an example as an antecedent and its class as a consequent. All correct classification rules that only use attributes included in the example set must be generalisations of such rules.

Third, although the search space for correct rules is extremely large, searches through this space can be tightly constrained. If there are *n* binary attributes then the search space is $2^n$. However, as will be demonstrated, a number of heuristics enable a search through this search space to only examine a portion of this search space.

The LEI algorithm is built on *the assumption of example set adequacy* - that any rule that does not mis-classify any of the example instances will not mis-classify any novel instance. An assumption such as this is necessary if we are to select between different candidate classification rules on anything other than an ad hoc basis. The significance of this particular assumption is that it does not require us to consider either statistical or domain dependent considerations when evaluating the relative merits of potential rules. Rather, they can be evaluated solely on logical grounds.

It follows from the assumption of example set adequacy that the best classification rules to develop are the most general that correctly classify all instances in the example set. Being most general they will classify the most novel instances and, as long as they correctly classify the example instances, by the assumption of example set adequacy, they will not mis-classify any novel instances.

To clarify these notions it is necessary to introduce some terminology. Any classification rule that does not mis-classify any example instance will be called a *correct rule.* Any correct rule for which there is no generalisation that is a correct rule will be called a *most general correct rule* (MGCR).

The MGCR for an example set will be the best possible classification rule. However, a MGCR will often contain disjunctions. Rather than developing classification rules with disjunctive antecedents, the LEI algorithm (like most variants of the $A^q$ algorithm) restricts itself to the use of conjunctive antecedents. Disjunction is represented by multiple classification rules. The best non-disjunctive classification rules will be the *most general correct non-disjunctive rules* (MGCNDRs).

By not considering disjunctive rules, the LEI algorithm is able to avoid the use of the disjunction addition (or adding alternative) generalisation operation (Michalski, 1984). This greatly reduces the number of possible classification rules that must be considered. This is achieved without any reduction in the classificatory power of the algorithm.

However, any example set will have a large number of MGCNDRs. For example, the MGCNDRs of the example set in Figure 1 are

$\forall X\ (red(X) \Rightarrow positive(X)),$
$\forall X\ (large(X) \Rightarrow positive(X)),$
$\forall X\ (matte(X) \Rightarrow positive(X)),$
$\forall X\ (triangular(X) \Rightarrow positive(X)),$
$\forall X\ (blue(X) \Rightarrow negative(X)),$
$\forall X\ (round(X) \& \ small(X) \Rightarrow negative(X)),$
$\forall X\ (square(X) \& \ small(X) \Rightarrow negative(X)),$
$\forall X\ (glossy(X) \& \ square(X) \Rightarrow negative(X)),$

From just five very simple examples there are eight MGCNDRs. In general, the more predicates (or attributes) that are considered, the more MGCNDRs that will exist. Rather than seeking to capture all MGCNDRs in the classification procedure, it will be assumed that the best classification procedure will be the one that contains the smallest number of MGCNDRs and that classifies all example instances. Any set of MGCNDRs which correctly classifies all instances in the example set will be called a *complete MGCNDR set.* Any complete MGCNDR set for which there is no other complete MGCNDR set which contains fewer MGCNDRs will be called a *minimal MGCNDR set.*

Where more than one minimal MGCNDR set exists, the simpler will be preferred. A count of the total number of conjuncts in the antecedents of the rules will be used herein as a measure of classification procedure complexity, although it is recognised that superior measures may exist (for some alternatives see Michalski, 1984.)

A requirement that will be placed on the classification procedure is *the requirement of universal classification* - hat no classification rule may refer to specific instances. This requirement is based on the assumption that the purpose of developing the classification procedure is to classify novel instances. It ensures that the classification procedure will not be restricted to classifying only instances in the example set.

## 3 The LEI algorithm.

The LEI algorithm starts from the most specific rules that can be formed from the example set and searches toward the desired minimal MGCNDR set. This search is conducted in two stages. First, a set of candidate MGCNDRs (the star) is discovered. The minimal MGCNDR set is constructed from the set of candidate MGCNDRs.

### 3.1 DERIVING MSCURS

There will be exactly one *most specific correct rule* (MSCR) for each instance in the example set. This rule will have as an antecedent the conjunction of all known details of the instance (other than its classification). The MSCR's consequent will be the instance's classification. In general, the set of MSCRs will provide a convenient means of describing the example set to the induction system. Figure 2 presents the set of MSCRs for the example set in Figure 1.

Given the requirement of universal classification, no MSCR will be of interest as a final classification rule. The first step in the algorithm is to form the set of most specific rules that do satisfy this requirement. These are called the *most specific correct universal rules* (MSCURs). They are formed by replacing all references to an instance in each MSCR by a single variable and discarding any resulting rules that are not correct. (A potential MSCUR will be incorrect only if the attributes used do not fully differentiate the classes in question. In this case, different examples with the same attributes may belong to different classes. It will not be possible to form a correct MSCUR for any such rule.) Figure 3 shows the MSCURs for the example set in Figure 1.

$red(a)\ \&\ glossy(a)\ \&\ round(a)\ \&\ large(a) \Rightarrow positive(a)$

$red(b)\ \&\ matte(b)\ \&\ square(b)\ \&\ large(b) \Rightarrow positive(b)$

$red(c)\ \&\ matte(c)\ \&\ triangular(c)\ \&\ small(c) \Rightarrow positive(c)$

$blue(d)\ \&\ glossy(d)\ \&\ round(d)\ \&\ small(d) \Rightarrow negative(d)$

$blue(e)\ \&\ glossy(e)\ \&\ square(e)\ \&\ small(e) \Rightarrow negative(e)$

**Figure 2: Most specific correct rules for the example set in Figure 1**

$\forall X\ (red(X)\ \&\ glossy(X)\ \&\ round(X)\ \&\ large(X) \Rightarrow positive(X))$

$\forall X\ (red(X)\ \&\ matte(X)\ \&\ square(X)\ \&\ large\ (X \Rightarrow positive(X))$

$\forall X\ (red(X)\ St\ matte(X)\ \&\ triangular(X)\ a\ small(X) \Rightarrow positive(X))$

$\forall X\ (blue(X)\ \&\ glossy(X)\ \&\ round(X)\ \&\ small(X) \Rightarrow negative(X))$

$\forall X\ (blue(X)\ \&\ glossy(X)\ \&\ square(X)\ \&\ small(X) \Rightarrow negative(X))$

**Figure 3: Most specific correct universal rules for the example set in Figure 1**

Note that where the available predicates are not sufficient to correctly classify all instances, some MSCRs will not have corresponding MSCURs. That is, the rules formed by replacing the references to the instance by a variable will be incorrect. For instance, no universal rule will be able to correctly classify all of the instances in the example set

*red(a) & glossy(a) & round(a) & large(a) & positive(a)*
*red(b) & glossy(b) & round(b) & large(b) & negative(b)*

(where positive and negative are the two classes.)

In such a case LEI forms a rule with the known attributes of the instances as antecedent as normal but with the consequent classifying the instance as belonging to the class *unclassifiable*. The resulting rule for the example set above is

$\forall X$ *(red(X) & glossy(X) & round(X) & large(X)* $\Rightarrow$ *unclassifiable(X))*.

The development of the set of MSCURs is one of the substantive differences between the LEI algorithm and other variants of the $A^q$ algorithm. All other variants work directly from the MSCRs. As the set of MSCURs will generally be smaller than the set of MSCRs, this provides LEI with a corresponding increase in efficiency by decreasing the number of entities that must be manipulated at various stages during an analysis.

## 3.2 DERIVING MGCNDRS

Finding the set of MGCNDRs from the set of MSCURs requires search. All MGCNDRs can be found by a simple search from each MSCUR through all non-disjunctive generalisations thereof that are correct rules. However, even with very restricted generalisation rules, such as conjunct deletion, this search is subject to extreme combinatorial explosion. Fortunately, however, there are a number of heuristics that can drastically prune the search space with only very minimal risk that members of the minimal MGCNDR set will not be found.

The first heuristic is *success pruning*. Unfortunately, it is not possible to constrain the search so that every generalisation of the MSCUR is examined only once. Consider, for example, a search for the MCCNDRS that can be derived from

$\forall X$ *(red(X) & glossy(X) & round(X) & large(X)* $\Rightarrow$ *positive(X))*.

with respect to the example set in Figure 1. In order to evaluate whether

1. $\forall X$ *(red(X) & glossy(X) & round(X)* $\Rightarrow$ *positive(X))*.

is a MGCNDR it is necessary to determine whether

2. $\forall X$ *(red(X) & glossy(X)* $\Rightarrow$ *positive(X))*

is correct (which it is, ensuring that 1 is not a *most general* correct rule.) Further, in order to evaluate whether

3. $\forall X$ *(red(X) & glossy(X) & large(X)* $\Rightarrow$ *positive(X))*

is a MGCNDR it is also necessary to consider 2 which again serves to determine that the rule is not a MGCNDR. In general, it is only possible to determine that a rule is not a MGCNDR if a non-disjunctive generalisation of it is found that is also correct.

Further, even once it has been determined that a correct rule is not a MGCNDR, it is still necessary to examine all of its non-disjunctive generalisations to determine whether any of them, or of their non-disjunctive generalisations, are MGCNDRs. However, there is no need to examine generalisations of a correct rule to see if they are also MGCNDRs if they have already been examined during an earlier stage of the search.

As many correct rules will be examined many times during a full search from a single MSCUR, a tremendous saving results if each correct rule that is examined is recorded, and this record is examined for each rule encountered during a search to determine whether further search is required. This is only worth while for correct rules as so many incorrect rules will be encountered in a search that it will be more efficient to directly test against the example set whether a rule is correct.

Success pruning will greatly reduce the number of rules examined while not affecting the set of MGCNDRs discovered by the search.

*MSCUR pruning* is another heuristic employed by the LEI algorithm. This heuristic takes advantage of the observation that every MSCUR will have as a generalisation at least one of the MGCNDRs in the minimal MGCNDR set. This must be the case as every possible instance must be classified by a MGCNDR in the minimal MGCNDR set. As a result, once a MGCNDR has been found, it will only be necessary to examine for further MGCNDRs MSCURs which do not have the discovered MGCNDR as a generalisation. If, each time that a MGCNDR in the minimal MGCNDR set has been found, all MSCURs that have that MGCNDR as a generalisation are discarded, then the search from the MSCURs need only examine one MSCUR for each MGCNDR in the minimal MGCNDR set. However, the catch is that it is only possible to determine which MGCNDRs are in the minimal MGCNDR set once all of the candidate MGCNDRs have been discovered.

The MSCUR pruning heuristic takes a slight risk of failing to discover the minimal MGCNDR set by discarding for each MGCUR that is discovered every MSCUR of which the MGCNDR is a generalisation. This introduces the risk that a MGCNDR in the minimal MGCNDR set may fail to be discovered because other MGCNDRS will cause the deletion of all MSCURs from which it could be discovered. This can only happen, however, if there are three or more MGCNDRs for a single class in the minimal MGCNDR set. Even so, the chances of it occurring are minimal.

The MSCUR pruning heuristic reduces the number of MSCURs from which searches must be conducted to, at most, the number of MGCNDRs in the minimal MGCNDR set.

The MSCUR pruning heuristic is similar to the $A^q$ step of removing from the example set all instances that are covered by the existing classification rules. It benefits, however, from the formation of the MSCURs by directly pruning potential rules rather than removing instances from the example set as does $A^q$.

## 4 Finding the minimal MGCNDR set

The previous section has defined efficient techniques for generating all MGCNDRs that may participate in a minimal MGCNDR set. This leaves the problem of selecting the minimal MGCNDR set from those MGCNDRs. A complete search for the minimal set would be costly. However, a simple heuristic allows sequential selection of MGCNDRs that, in practice, will almost invariably select a minimal MGCNDR from those available. Even when it does not select a minimal MGCNDR set, it will always create a complete MGCNDR set that will rarely contain many more MGCNDRS than a minimal MGCNDR set.

This heuristic is based on the observation that the minimal MGCNDR sets usually contain a small number of MGCNDRs that each correctly classify a large number of instances from the example set.

Each available MGCNDR is examined to determine how many instances from the example set it classifies. The MGCNDR that classifies the most instances is selected. Where more than one such MGCNDR exists, the one with the fewest antecedents is selected in line with the stated preference for simpler minimal MGCNDR sets.

In general it is necessary to select several rules in order to correctly classify all instances in the example set. The above process is repeated until the selected rules correctly classify all examples for which a MSCUR could be formed. After the first selection, rules are evaluated by the number of instances that they classify that are not correctly classified by rules already selected.

This heuristic ensures that where a single rule will classify all examples for a given class it will be selected as a classification rule. Where several such rules exist, the one with the fewest conjuncts will be selected. If other measures of rule quality were desired, they could easily be incorporated.

## 5 Implementation

The LEI algorithm has been implemented in 'C' on a Gould 6031. This initial implementation is restricted to discrete attribute values.

Restricted to discrete attribute values, the only generalisation rules that are required are variablisation (or turning constraints into variables) and conjunct deletion (or dropping condition) (Michalski, 1984). Variablisation is required to form the most specific correct universal rules and conjunct deletion is required for forming the most general correct rules.

Figures 4 to 7 present the results of some experiments that were conducted on this implementation of LEI and, for comparison, on a public domain implementation of ID3 in Pascal. In these experiments, data was randomly generated in conformity to a number of rules. Due to limitation on the public domain version of ID3 that was used, the experiments were restricted to two class data - a positive class and a negative class. LE i was set to generate rules for only the positive class as the negative class simply consists of all instances not belonging to the positive class. ID3 was run with an initial window of 100 with an increment limit of 10 per iteration. These parameters appeared to provide near optimal performance for a wide range of conditions in the experiment.

The two programs were run on each set of data. Four factors were crossed to produce the 48 sets of data –

- example set size - the examples sets were composed of either 1000, 2000 or 5000 instances;
- no of attribute values - in half of the sets of data each attribute had two values, in the other half each attribute had three values;
- rule complexity - the rules that defined the positive class were composed of either 2, 3, 4 or 5 conjuncts; and
- number of rules - the number of non-disjunctive rules required to define the positive class was varied from 2 to 3.

The comparison between results for ID3 and for LEI should be treated with extreme caution due to differences in the comparative efficiency of the languages in which they are implemented. Of greater interest than the absolute comparisons in processing time (over all data sets LEI averaged 15.13 seconds while ID3 averaged 48.3 seconds) are the trends as the data sets are manipulated.

The increase in processing time resulting from an increase in the size of the data set appears to be linear in both cases. However, the rate of increase is lower for LEI than ID3. The reason that LEI is able to so effectively constrain this increase is its use of MSCURs. The number of most specialised correct *universal* rules that can be derived from a data set will rise at a lower rate than the increase in the size of the data set as some new instances will be covered by existing rules. Thus, after the stage of MSCUR generation, the impact of additional instances is minimised.

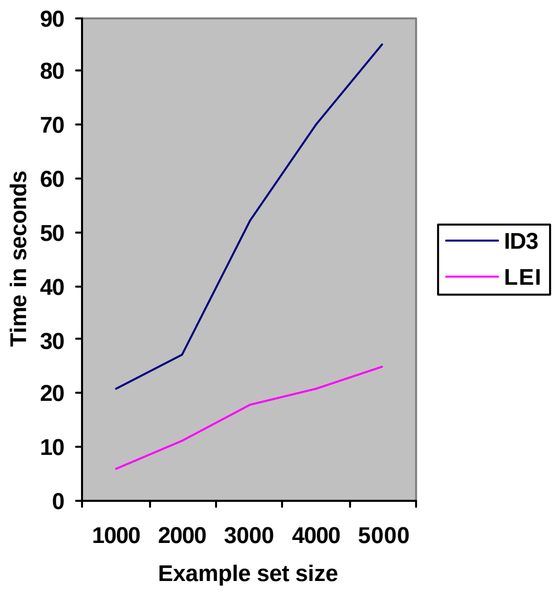

**Figure 4: Average computation time for differing example set sizes**

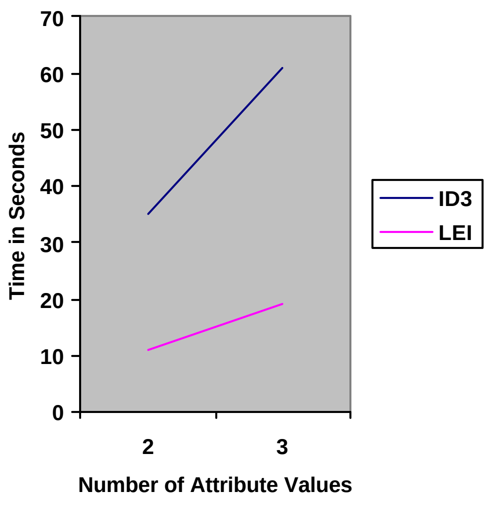

**Figure 5: Average computation time for differing numbers of attribute values**

Figure 5 shows that while both algorithms suffer from an increase in processing time as a result of an increase in the number of values for each attribute, the rate of increase is low for LEI in comparison to ID3.

Figure 6 indicates that both algorithms suffer from an increase in processing time as they encounter an increase in the number of non-disjunctive rules required to define the positive case. The increase for LEI is minimal, however. This is another consequence of the use of MSCURs. A large amount of LEI's processing time is devoted to generating the MSCUR. This time is well spent, as evidenced by LEI's overall efficiency and response to an increase in the size of the example set. However, an increase in the number of rules required to define a class does not affect the amount of processing involved in generating the MSCUR. The only increase in processing time is in the generation of the MGCRs and the minimal MCCR set. Both of these stages are made highly efficient by the use of MSCURs.

Figure 7 shows the two algorithms' reactions to an increase in the complexity of the rules used to define the positive case. It can be seen that there is a small but significant reduction in LEI's processing time as a result of an increase in the complexity of the rules that it is deriving. This is a result of a decrease in the amount of generalisation that is required to create the MGCRS when the rules have more conjuncts.

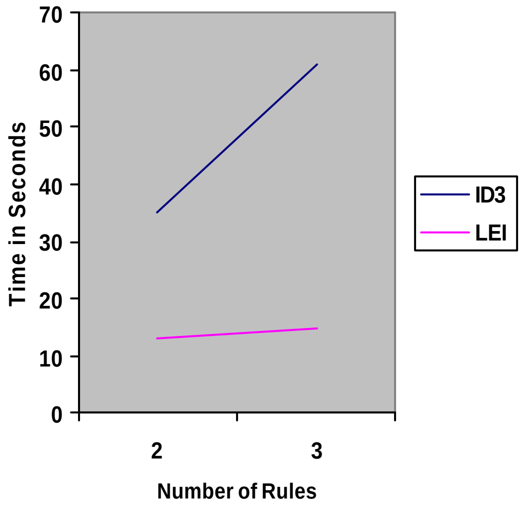

**Figure 6: Average computation time for differing numbers of positive rules**

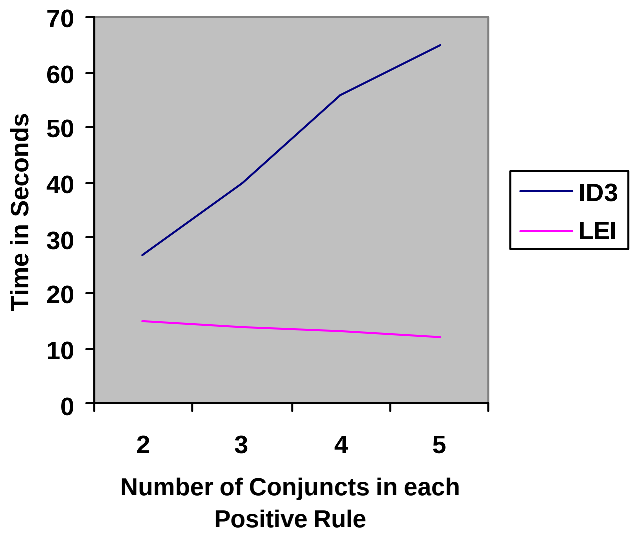

**Figure 7: Average computation time for differing rule complexities**

These results clearly demonstrate that with the advent of the LEI algorithm it can no longer be said that induc tion through exhaustive generalisation is inefficient.

## 6 Future research

There is a need to develop further heuristics to improve the efficiency of the LEI algorithm when applied to more complex predicates. Two types of extension are necessary, the use of continuous values, and the use of more complex logical formulae.

The LEI algorithm is also restricted in that it is highly susceptible to noise. Any error in the example set will result in an erroneous classification procedure. There is a need to develop a more sophisticated method for deriving MSCURs that is less susceptible to noise.

## 7 Summary

The LEI algorithm for empirical induction is based on a search from the set of most specific rules that describe the available evidence to the minimal set of rules that fully classify the example set. Positive evidence is used to initialise the search with the creation of the MSCURs. Negative evidence is used to constrain the search by limiting the rules that axe considered to those that are correct for all examples.

The success pruning and MSCUR pruning heuristics make this search computationally efficient for data equivalent to discrete attribute value data.

The LEI algorithm differs from previous logic based approaches to empirical induction in that it exhaustively examines all correct generalisations of the example set. This prevents it from pruning possibilities that are evaluated as undesirable by some heuristic while actually leading to the best classification rule.

The LEI algorithm provides its evaluation procedure in the form of a set of logic based implications, which is a fax clearer representation than the decision trees provided by the only other efficient approach to exhaustive empirical induction - the statistic based approach.

## Acknowledgements

Part of this research was conducted in the Griffith University School of Computing and Information Technology.

Many thanks to Ross Quinlan for providing access to the version of 103 that was used in the preparation of this paper.

## References

1. Buchanan, B. G. and Feigenbaum, E. A. (1978) “DENDRAL and Meta-DENDRAL: Their Applications Dimension”, *Artificial Intelligence*, Vol 11, pp. 5 - 24.
2. Dietterich, T. C. and Michalski, R. S. (1981) “Inductive Learning of Structural Descriptions: Evaluation Criteria and Comparative Review of Selected Methods”, *Artificial Intelligence*, Vol 16, pp. 257 - 294.
3. Hunt, E. B., Marin, J. and Stone, P. J. (1966). *Experiments in induction.* Academic Press, New York.
4. Michalski, R. S. (1977). “Synthesis of Optimal and Quasi-optimal Variable-valued Logic Formulas”, in *Proceedings of the 1975 International Symposium on Multiple Valued Logic,* Bloomington, Ind., pp. 76 - 87.
5. Michalski, R. S. (1984). “A Theory and Methodology of Inductive Learning”, in R. S. Michalski, J. G. Carbonell, and T. M. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach,* Springer-Verlag, Berlin, pp. 83 - 129.
6. Michalski, R. S. (1980) “Pattern recognition as rule-guided inductive inference”, *IEEE Trans on Pattern Analysis and Machine Intelligence*, Vol.PAMI-2, No 4 (July), pp 349 - 361.
7. Mitchell, T. M. (1977). “Version spaces: A candidate elimination approach to rule learning”, in *Proceedings of the Fifth International Joint Conference on Artificial Intelligence,* pp. 305 - 310.
8. Quinlan, .J. R. (1986) “Induction of Decision Trees”, *Machine Learning*, Vol 1, pp. 81 - 106.
9. Quinlan, J. R. (1987*a*) “Generating Production Rules From Decision Trees”, in *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, Morgan Kaufmann, Los Altos, pp. 304 - 307.
10. Quinlan, J. R. (1987*b*), “Induction, Knowledge and Expert Systems”, in *Proceedings of the Australian Joint Artificial Intelligence* Conference, Sydney, pp. 223 - 240.
11. Winston, P. H. (1975). “Learning Structural Descriptions from Examples. In P. H. Winston (Ed.) *The Psychology of Computer Vision*, McGraw-Hill, New York, pp. 157 - 209.